សៀវភៅ Python

រៀនសរសេរកម្មវិធីកុំព្យូទ័រដោយប្រើភាសា Python

ដោយៈ ក្សរស្រួច

លំនាំដើម 13

កម្មវិធី 13

ផ្នែកទន់ចាំបាច់ 15

វត្ថុនិងឈ្មោះ 19

វត្ថុ 19

ឈ្មោះរបស់វត្ថុ 20

ប្រភេទនៃវត្ថុ 25

ចំនួនគត់ 25

ប្រមាណវិធីនព្វន្ត *26*

អាទិភាពនៃប្រមាណសញ្ញា *29*

ចំនួនពិត 30

ប្រមាណវិធីនព្វន្ត *31*

តក្កវត្ថុ 32

ប្រមាណវិធីតក្កវិទ្យា *33*

ប្រមាណវិធីប្រៀបធៀប *36*

ប្រមាណវិធីអត្តសញ្ញាណ *38*

មោឃៈវត្ថុ 39

កម្រងអក្សរ 39

ប្រមាណវិធីបូកបន្ត *43*

ប្រមាណវិធីគុនបន្ត *43*

ប្រមាណវិធីលេខរៀង *44*

ប្រមាណវិធីកាត់ចម្លង *46*

ប្រមាណវិធីតក្កវិទ្យា *48*

ប្រមាណវិធីប្រៀបធៀប *49*

ប្រមាណវិធីអត្តសញ្ញាណ *51*

ប្រមាណវិធីរកធាតុ *52*

កំណាត់ពន្យល់ *53*

កម្រងថេរ 54

ប្រមាណវិធីបូកបន្ត *56*

ប្រមាណវិធីគុនបន្ត *56*

ប្រមាណវិធីតក្កវិទ្យា *56*

ប្រមាណវិធីប្រៀបធៀប *58*

ប្រមាណវិធីរកធាតុ *60*

ប្រមាណវិធីអត្តសញ្ញាណ *61*

ប្រមាណវិធីលេខរៀង *61*

ប្រមាណវិធីកាត់ចម្លង *62*

គម្រូងអថេរ 63

ប្រមាណវិធីបូកបន្ត *64*

ប្រមាណវិធីគុនបន្ត *64*

ប្រមាណវិធីតក្កវិទ្យា *64*

ប្រមាណវិធីប្រៀបធៀប *66*

ប្រមាណវិធីរកធាតុ *68*

ប្រមាណវិធីអត្តសញ្ញាណ *68*

ប្រមាណវិធីលេខរៀង *70*

ប្រមាណវិធីកាត់ចម្លង *71*

វចនានុក្រម 73

ប្រមាណវិធីលេខរៀង *73*

ប្រមាណវិធីតក្កវិទ្យា *75*

ប្រមាណវិធីប្រៀបធៀប *76*

ប្រមាណវិធីអត្តសញ្ញាណ *77*

ប្រមាណវិធីរកធាតុ *78*

សំណុំ 78

ប្រមាណវិធីដកធាតុ *79*

ប្រមាណវិធីប្រជុំ *80*

ប្រមាណវិធីប្រេសញ្ញ......82

ប្រមាណវិធីប្រជុំធាតុខុសគ្នា......83

ប្រមាណវិធីតក្កវិទ្យា......84

ប្រមាណវិធីប្រៀបធៀប......85

ប្រមាណវិធីរកធាតុ......87

ប្រមាណវិធីអត្តសញ្ញាណ......88

សមាសវត្ថុនៃសមាសវត្ថុ...... 89

ប្រភេទនៃបញ្ជា......92

បញ្ជាចាត់តាំង...... 92

ស្វ័យប្រមាណវិធី......95

កន្សោមប្រមាណវិធី...... 102

បញ្ជា if...... 103

បញ្ជា if/else...... 104

បញ្ជា if/elif/else...... 105

កន្សោមប្រមាណវិធីមានជម្រើស......107

បញ្ជា while...... 108

បញ្ជា for...... 110

បញ្ជា break 111

បញ្ជា continue 112

បញ្ជា pass 113

បញ្ជា while/else 114

បញ្ជា for/else 116

វដ្តកម្មក្នុងវដ្តកម្ម 117

ក្បួន 119

ការបង្កើតក្បួន 119

ការយកក្បួនមកប្រើ 122

ដំណឹងនិងដំណាង 127

បញ្ជា return 129

ដំណឹងតាមលេខរៀង 130

ដំណឹងតាមដំណាង 131

ដំណឹងមានស្រាប់ 132

ការបំបែកដំណឹង 135

ការប្រមូលផ្តុំដំណឹង 136

លំដាប់ថ្នាក់នៃដំណាងនិងដំណើង 138

ដែនកំណត់ 140

ដែនកំណត់សកល........ 140

ដែនកំណត់ដោយឡែក........ 141

ដែនកំណត់ចារឹកក្នុងនិងដែនកំណត់ចារឹកក្រៅ........ 142

ដែនកំណត់ទូទៅ........ 143

ការស្វែងរកវត្ថុ........ 145

ក្បួនមានស្រាប់ 153

ថ្នាក់........ 158

ការបង្កើតថ្នាក់........ 158

ការយកថ្នាក់មកប្រើ 161

ការយកសម្បត្តិថ្នាក់មកប្រើ 163

ការបន្ថែមវត្ថុចូលទៅក្នុងថ្នាក់........ 165

ស្ថាបនិក........ 166

វិធី 167

និន្នន័យគម្រូ 173

សិស្ស........ 175

ការបន្តថ្នាក់ 185

សម្បត្តិឈ្មោះដូចគ្នា 195

ពហុបន្តថ្នាក់ 202

ការបន្តថ្នាក់រាងចតុកោណស្មើ 210

ថ្នាក់មានស្រាប់ 217

ថ្នាក់ int 219

ថ្នាក់ float 221

ថ្នាក់ bool 223

ថ្នាក់ str 224

ថ្នាក់ tuple 229

ថ្នាក់ list 231

ថ្នាក់ dict 236

ថ្នាក់ set 240

សាស្ត្រា 244

ការបង្កើតសាស្ត្រា 244

ការយកសាស្ត្រាមកប្រើ 245

ការយកសម្បត្តិសាស្ត្រាមកប្រើ 247

កញ្ចប់ 251

ការបង្កើតកញ្ចប់ 251

ការយកសាស្ត្រាក្នុងកញ្ចប់មកប្រើ 253

បណ្ណាល័យមជ្ឈឹម 256

ការស្វែងរកសាស្ត្រា 256

ភាពមិនប្រក្រតី 258

ប្រភេទនៃភាពមិនប្រក្រតី 258

បញ្ជា try/except 260

បញ្ជា try/except/else 265

បញ្ជា try/except/finally 266

បញ្ជា try/except/else/finally 266

បញ្ជា raise 267

បញ្ជា assert 268

ការបង្កើតថ្នាក់នៃភាពមិនប្រក្រតី 268

បច្ចេកទេសជាន់ខ្ពស់ 271

ក្បួនអនាមិក 271

ទឡ្ហនករ 272

ក្បួនផលិតករនិងផលិតករ 276

កម្រងអថេររូបមន្ត 283

កន្សោមផលិតករ 286

វិធីពិសេស 287

វិធីពិសេសសម្រាប់ប្រមាណវិធីនព្វន្ត *292*

វិធីពិសេសសម្រាប់ប្រមាណវិធីប្រៀបធៀប *296*

វិធីពិសេសសម្រាប់សមាសវត្ថុ *299*

វិធីពិសេសសម្រាប់សិស្ស *301*

សម្បត្តិពិសេស 304

សម្បត្តិពិសេសឈ្មោះ __dict__ *305*

សម្បត្តិពិសេសឈ្មោះ __doc__ *307*

សម្បត្តិពិសេសឈ្មោះ __slots__ *308*

សម្បត្តិឯកជន 309

លក្ខណៈសម្បត្តិ 311

វិធីឯកោ 313

វិធីប្រចាំថ្នាក់ 315

រចនាករ 316

ថ្នាក់មេអរូបី 318

ស្វ័យសេវា 319

ការចម្លងសមាសវត្ថុ 321

ការចម្លងសើលើ *321*

ការចម្លងទាំងស្រុង *325*

អាណាព្យាបាលនិងបញ្ជា with 326

អាណាព្យាបាល *326*

បញ្ជា with *328*

បញ្ជា with/as *331*

ពាក្យបច្ចេកទេស 333

ឯកសារស្រាវជ្រាវ 336

លំនាំដើម

កម្មវិធី

នៅក្នុងជីវភាពរស់នៅរបស់យើងសព្វថ្ងៃ យើងតែងឮគេនិយាយពីកម្មវិធីផ្សេងៗមានដូចជា កម្មវិធីបុណ្យ កម្មវិធីអភិវឌ្ឍន៍ កម្មវិធីទូរទស្សន៍ជាដើម។ តើពាក្យថា "កម្មវិធី" នោះមានន័យដូចម្ដេចដែរ?

បើយើងលើកយកកម្មវិធីបុណ្យមកពិនិត្យមើលយើងនឹងឃើញថា កម្មវិធីបុណ្យគឺជាអត្ថបទរៀបរាប់ពីធ្វើពិធីផ្សេងៗដែលត្រូវធ្វើឡើងតាំងពីពេលចាប់ផ្ដើមបុណ្យរហូតដល់ពេលចប់បុណ្យក្នុងគោលបំណងប្រារព្ធពិធីបុណ្យទាំងមូល។ ដូចនេះកម្មវិធីបុណ្យគឺជាការរៀបចំការងារដែលត្រូវធ្វើតាមលំដាប់លំដោយដើម្បីធ្វើពិធីបុណ្យទាំងមូល។

មួយវិញទៀត បើសិនជាយើងលើកយកកម្មវិធីអភិវឌ្ឍន៍មកពិនិត្យមើលវិញម្ដង យើងនឹងឃើញថា កម្មវិធីអភិវឌ្ឍន៍គឺជាអត្ថបទរៀបរាប់ពីការងារទាំងឡាយណាដែលត្រូវធ្វើតាមលំដាប់លំដោយនៅក្នុងរយៈពេលមានកម្រឹតណាមួយដើម្បីកាត់បន្ថយភាពក្រីក្រ។

ចំណែកឯកម្មវិធីទូរទស្សន៍វិញ វាគឺជាអត្ថបទរៀបរាប់ពីការបញ្ចាំងនាទីផ្សេងៗសម្រាប់រយៈពេលពេញមួយថ្ងៃ។

សរុបមក ទាំងកម្មវិធីបុណ្យ ទាំងកម្មវិធីអភិវឌ្ឍន៍ ទាំងកម្មវិធីទូរទស្សន៍ គឺសុទ្ធតែស្ដែងចេញពីការងារដែលត្រូវធ្វើតាមលំដាប់លំដោយក្នុងរយៈពេលមានកម្រឹតណាមួយដើម្បីដោះស្រាយបញ្ហាមួយចំនួន។

ចំពោះកម្មវិធីកំព្យូទ័រវិញក៏មានលក្ខណៈស្រដៀងនឹងកម្មវិធីទាំងអស់ខាងលើនេះដែរ ពោលគឺវាក៏ជាអត្ថបទរៀបរាប់ពីការងារមួយចំនួនដែលតម្រូវឲ្យកំព្យូទ័រយកទៅធ្វើក្នុងគោលបំណងដោះស្រាយបញ្ហាមួយចំនួនដែរ។ តែដោយហេតុថា កំព្យូទ័រអាចធ្វើការងារជាច្រើនមានចំនួនរាប់ម៉ឺនសែនតែក្នុងរយៈពេលតែមួយប៉ប្រិចភ្នែកតែប៉ុណ្ណោះ ដូចនេះការងារទាំងអស់នៅក្នុងកម្មវិធីកំព្យូទ័រ មិនទាមទារឲ្យមានរយៈពេលកំណត់ឡើយ។ បច្ចុប្បន្ននេះ កំព្យូទ័រទំនើបៗអាចធ្វើការងារចំនួនរាប់លានតែក្នុងមួយវិនាទីតែប៉ុណ្ណោះ។

និយាយឲ្យខ្លី នៅពេលណាដែលយើងសរសេររៀបរាប់ពីការងារមួយចំនួនតម្រូវឲ្យកំព្យូទ័រធ្វើតាមលំដាប់លំដោយក្នុងគោលបំណងដោះស្រាយបញ្ហាណាមួយ គឺនៅពេលនោះហើយដែលយើងសរសេរកម្មវិធីកំព្យូទ័រ។ ជាក់ស្តែង ដើម្បីរៀបចំកម្មវិធីតម្រូវឲ្យកំព្យូទ័រដោះស្រាយបញ្ហារកប្រាក់ចំណេញ យើងអាចធ្វើដូចខាងក្រោមនេះ៖

គណនារកថ្លៃទិញ
គណនារកថ្លៃលក់
ប្រាក់ចំណេញ = ថ្លៃលក់ - ថ្លៃទិញ

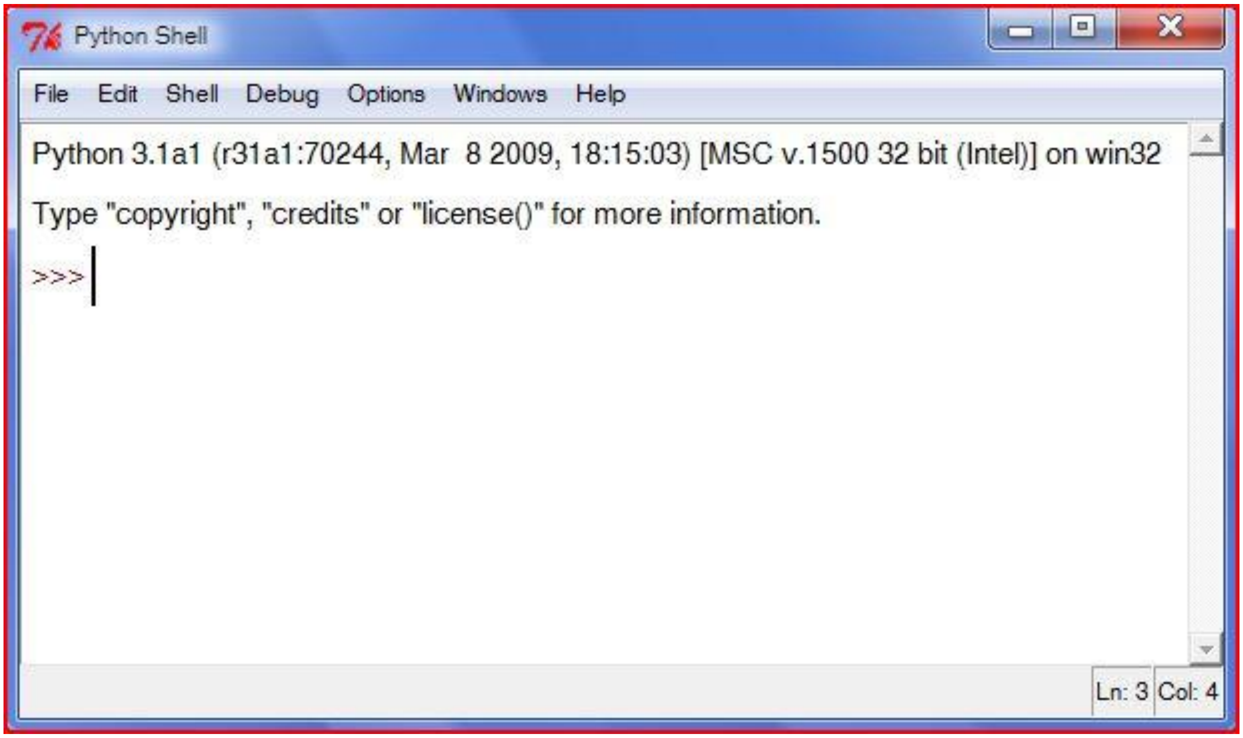
ការធ្វើដូចខាងលើនេះហៅថាការសរសេរ *ដំណោះស្រាយ* (algorithm) ឬការរៀបចំផែនការដែលជាកម្មវិធីជាភាសាខ្មែរយើងធម្មតា។ ក៏ប៉ុន្តែ កំព្យូទ័រមិនអាចយល់កម្មវិធីជាភាសាខ្មែរដូចខាងលើនេះបានឡើយ គឺយើងត្រូវសរសេរកម្មវិធីខាងលើនេះជាភាសាណាមួយដែលកំព្យូទ័រអាចយល់បាន និងអាចយកទៅ *អនុវត្ត* (execute) ដើម្បីដោះស្រាយបញ្ហារកប្រាក់ចំណេញនេះ។

កន្លងមក យើងទាំងអស់គ្នាសុទ្ធតែបានស្គាល់រួចមកហើយនូវអ្វីដែលហៅថាដំណោះស្រាយនោះ។ ពីព្រោះយើងទាំងអស់គ្នាសុទ្ធតែធ្លាប់បានធ្វើការស្រាយបំភ្លឺឬធ្វើការដោះស្រាយលំហាត់ធរណីមាត្រឬគណិតវិទ្យាផ្សេងៗរួចមកហើយដែរ។ ម្យ៉ាងទៀតការសរសេរ

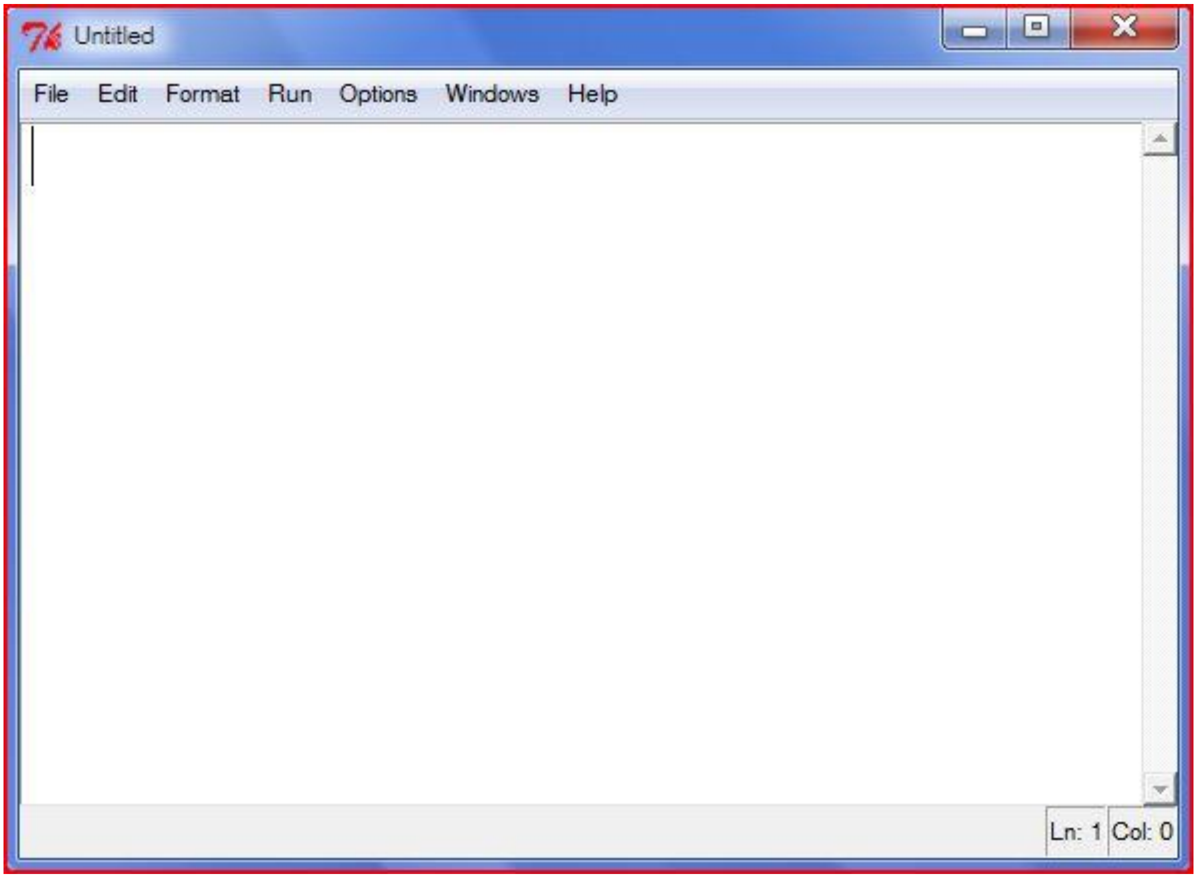
ដំណោះស្រាយសម្រាប់កម្មវិធីកំព្យូទ័រ គឺគ្មានអ្វីខុសប្លែកពីការសរសេរដំណោះស្រាយនៅក្នុង លំហាត់គណិតវិទ្យាឡើយ។ ពេលគឺយើងត្រូវសេសេរចម្លើយផ្សេងដោយរៀបចំទៅតាម លំដាប់លំដោយនៃការដោះស្រាយបញ្ហា។

ផ្នែកទន់ ចាំបាច់

នៅក្នុងចំណោមភាសាប្រើសម្រាប់សរសេរកម្មវិធីកំព្យូទ័រ Python គឺជាភាសាមួយដែលស្រួល រៀនជាងគេ ព្រោះវាមានលក្ខណៈដ៏សាមញ្ញបំផុត។ ដើម្បីអាចសរសេរកម្មវិធីជាភាសា Python បាន យើងចាំបាច់ត្រូវតែទាញយកផ្នែកទន់ Python 3 ឬថ្មីជាងនេះពីគេហទំព័រ www.python.org ។ នៅក្នុងប្រព័ន្ធប្រតិបត្តិការ Windows យើងអាចប្រើប្រាស់ IDLE ជា *រោងជាង* (IDE) ដែលជាកន្លែងសម្រាប់សរសេររៀបចំកម្មវិធីជាភាសា Python ។ ដើម្បី ដំណើរការរោងជាង IDLE យើងត្រូវចុច Start > All Programs > Python 3 > IDLE ។



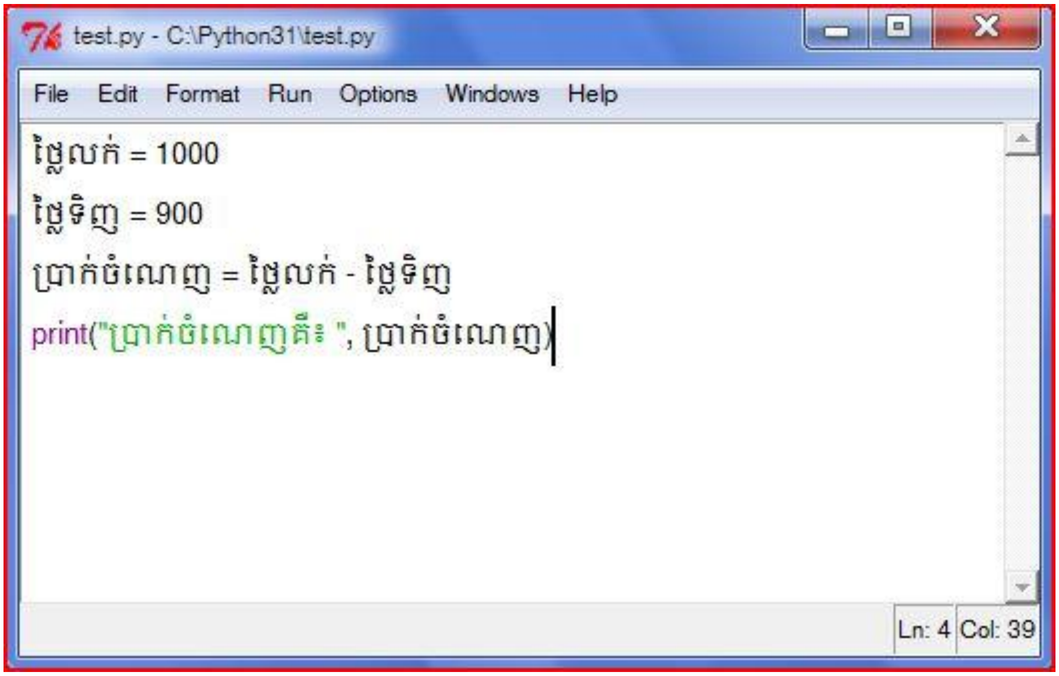
ដើម្បីបង្កើតឯកសារដែលជាកម្មវិធីជាភាសា Python យើងត្រូវចុច File > New Window ។ យើងនឹងឃើញមានបង្អួចមួយទៀតដូចនៅក្នុងរូបខាងក្រោមនេះ៖



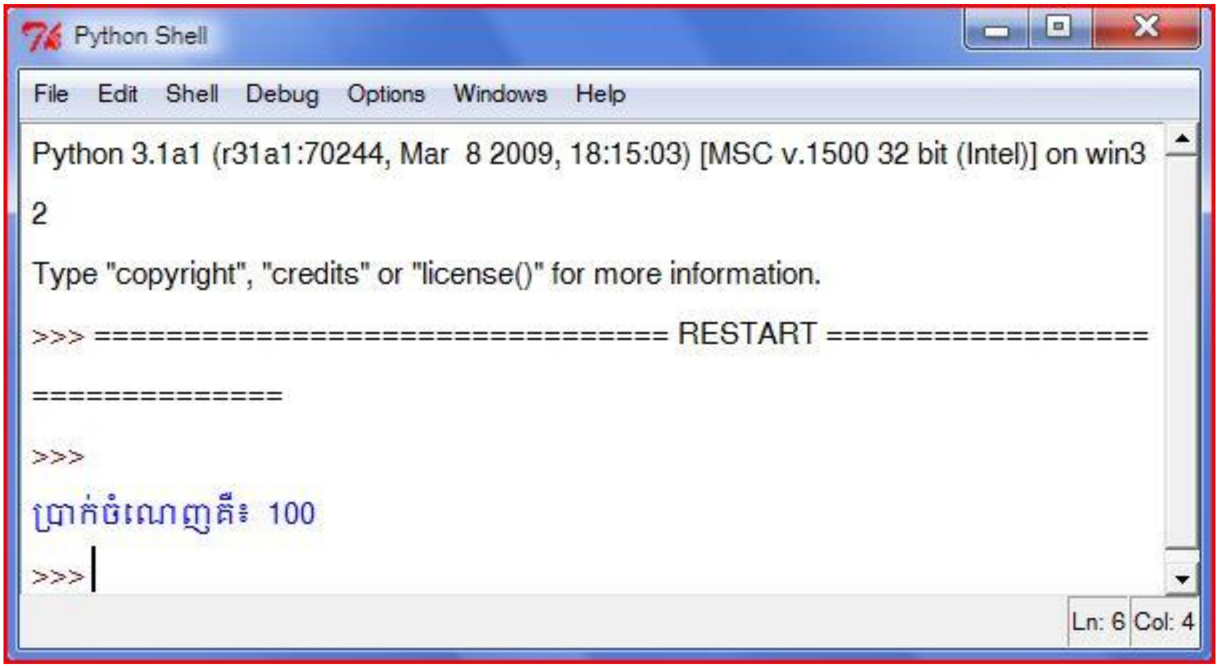
គឺនៅលើបង្អួចចុងក្រោយនេះហើយដែលយើងនឹងសរសេរកម្មវិធីជាភាសា Python 3 ទាំងឡាយ។ ក៏ប៉ុន្តែគួរឲ្យស្តាយដែរ បង្អួច IDLE មិនអនុញ្ញាតឲ្យយើងសរសេរអក្សរខ្មែរនៅលើនោះដោយផ្ទាល់បានឡើយ យើងចាំបាច់ត្រូវតែសរសេរអក្សរខ្មែរនៅក្នុងកម្មវិធីណាមួយរួចសឹមចម្លងយកមកដាក់នៅលើបង្អួច IDLE នេះ។ ដើម្បីសរសេរអក្សរខ្មែរ យើងអាចប្រើប្រាស់កម្មវិធីមួយចំនួនមានដូចជា Notepad ជាដើម។

ជាកិច្ចចាប់ផ្តើម ចូលយើងសរសេរកម្មវិធីដូចខាងក្រោមនេះ៖

```
ថ្លៃលក់ = 1000
ថ្លៃទិញ = 900
ប្រាក់ចំណេញ = ថ្លៃលក់ - ថ្លៃទិញ
print("ប្រាក់ចំណេញគឺ៖ ", ប្រាក់ចំណេញ)
```



បន្ទាប់មកទៀត យើងត្រូវរក្សាកម្មវិធីខាងលើនេះទុកដោយដាក់ឈ្មោះថាអ្វីមួយ មានដូចជា test.py ជាដើម។ ហើយដើម្បី *ដំណើរការ* (run) កម្មវិធីនេះ យើងត្រូវចុច F5 ។ យើងនឹងឃើញលទ្ធផលដូចនៅក្នុងរូបខាងក្រោមនេះ៖



នៅពេលដែលកម្មវិធីខាងលើនេះដំណើរការ *ផ្នែកទន់បកប្រែ* (interpreter) នឹងបកប្រែអ្វីៗទាំងអស់ដែលមាននៅក្នុងកម្មវិធីនោះ ពីភាសា Python ទៅជាភាសាម៉ាស៊ីន ដើម្បីពន្យល់ប្រាប់កំព្យូទ័រឲ្យធ្វើការងារផ្សេងៗដូចមានចែងនៅក្នុងកម្មវិធីនោះ។ ហើយការបកប្រែគឺត្រូវធ្វើឡើងពីលើចុះក្រោមនិងពីឆ្វេងទៅស្តាំមួយបន្ទាត់ម្តងៗ។

មួយវិញទៀត ដោយហេតុថាការងារទាំងឡាយនៅក្នុងកម្មវិធីកំព្យូទ័រ គឺជាការងារដែលតម្រូវឲ្យកំព្យូទ័រយកទៅធ្វើ ដូចនេះការងារទាំងនោះមានលក្ខណៈជា *បញ្ជា* (statement) តម្រូវឲ្យកំព្យូទ័រយកទៅអនុវត្តដើម្បីដោះស្រាយបញ្ហាផ្សេងៗ។

សរុបមក ការសរសេរកម្មវិធីកំព្យូទ័រ គឺជាការសរសេររៀបចំបញ្ជាមួយចំនួនតាមលំដាប់លំដោយតម្រូវឲ្យកំព្យូទ័រយកទៅអនុវត្តក្នុងគោលបំណងដោះស្រាយបញ្ហាមួយចំនួន។

វត្ថុនិងឈ្មោះ

វត្ថុ

បើយើងលើកយកបញ្ហាដោះស្រាយរកប្រាក់ចំណេញមកវិភាគមើលម្តងទៀត យើងនឹងឃើញថា ដើម្បីរកប្រាក់ចំណេញ យើងត្រូវមានព័ត៌មានពីរគឺ ថ្លៃលក់ និង ថ្លៃទិញ ពីព្រោះប្រាក់ចំណេញ គឺជាផលដករវាង ថ្លៃលក់ និង ថ្លៃទិញ ។ នៅក្នុងវិស័យព័ត៌មានវិទ្យា ព័ត៌មានចាំបាច់សម្រាប់យកមកដោះស្រាយបញ្ហាទាំងនោះហៅថា *ទិន្នន័យ* (data) ដូចនេះការដោះស្រាយបញ្ហារកប្រាក់ចំណេញ គឺជាការយកទិន្នន័យដែលជា ថ្លៃលក់ និងទិន្នន័យដែលជា ថ្លៃទិញ មកធ្វើប្រមាណវិធីដក។

នៅក្នុងការសរសេរកម្មវិធីជាភាសា Python ដើម្បីឲ្យមានទិន្នន័យចាំបាច់ទាំងនោះ យើងចាំបាច់ត្រូវតែសរសេរចេញបញ្ជាតម្រូវឲ្យកំព្យូទ័របង្កើតទិន្នន័យទាំងនោះទុកនៅក្នុង *សតិ* (memory) របស់វាផ្ទាល់តែម្តង។ ហើយការសរសេរកម្មវិធីបង្កើតទិន្នន័យផ្សេងៗនៅក្នុងភាសា Python គឺត្រូវធ្វើឡើងដូចខាងក្រោមនេះ៖

```
1000
```

ខាងលើនេះគឺជាកម្មវិធីជាភាសា Python *ជាអប្បបរិមា* (minimal) ដែលនៅក្នុងនោះមានបញ្ជាតម្រូវឲ្យបង្កើតទិន្នន័យលេខ1000 មួយទុកនៅក្នុងសតិរបស់កំព្យូទ័រ។

ដោយហេតុថានៅក្នុងភាសា Python ផ្នែកនៃសតិរបស់កំព្យូទ័រដែលមានព័ត៌មាននៅក្នុងនោះត្រូវហៅថា *វត្ថុ* (object) ដូចនេះការសរសេរ 1000 គឺជាការបង្កើតវត្ថុលេខ 1000 មួយ។ មួយវិញទៀត យើងអាចប្រៀបប្រដូចវត្ថុទៅនឹងប្រអប់មួយដែលមានទិន្នន័យនៅក្នុងនោះ។

ឈ្មោះរបស់វត្ថុ

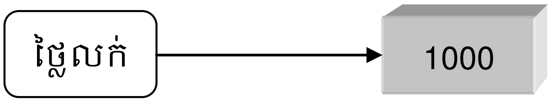
វត្ថុដែលត្រូវបានបង្កើតដូចនៅក្នុងកម្មវិធីខាងលើនេះ ត្រូវបានលុបចេញវិញជាបន្ទាន់ពីក្នុងសតិរបស់កំព្យូទ័រ គឺយើងមិនអាចយកវាទៅប្រើការធ្វើការអ្វីផ្សេងទៀតបានឡើយ។ ដើម្បីកុំឲ្យវត្ថុដែលត្រូវបានបង្កើតរួចហើយត្រូវលុបចេញវិញ យើងត្រូវដាក់ឈ្មោះឲ្យវត្ថុនោះដោយត្រូវសរសេរកម្មវិធីដូចខាងក្រោមនេះ៖

```
ថ្លៃលក់ = 1000
```

ថ្លៃលក់ = 1000 គឺជាបញ្ជាតម្រូវឲ្យបង្កើតវត្ថុលេខ 1000 ដែលមានឈ្មោះថា ថ្លៃលក់ ។

យើងត្រូវធ្វើការកត់សំគាល់ថា សញ្ញាស្មើ (=) នៅក្នុងភាសា Python គឺជាសញ្ញាប្រើសម្រាប់ *ភ្ជាប់* (bind) *ឈ្មោះ* (identifier) ទៅនឹងវត្ថុ គឺមិនមែនជាសញ្ញាប្រើសម្រាប់បញ្ជាក់លទ្ធផលបានមកពីការធ្វើប្រមាណវិធីឡើយ។

ការភ្ជាប់ឈ្មោះ ថ្លៃលក់ ទៅនឹងវត្ថុលេខ 1000 អាចតាងដោយគំនូសបំព្រួញដូចខាងក្រោមនេះ



ឈ្មោះរបស់វត្ថុអាចជាពាក្យជាភាសាណាមួយក៏បានដែរ ឲ្យតែឈ្មោះទាំងនោះត្រូវចាប់ផ្តើមដោយតួអក្សរមិនមែនជាលេខ។ តែយើងអាចប្រើសញ្ញា _ ជាតួអក្សរទីមួយបាន។ ម៉្យាងទៀតឈ្មោះរបស់វត្ថុមិនត្រូវមានអក្សរដកឃ្លា (space) នៅក្នុងនោះបានឡើយ ហើយឈ្មោះទាំងនោះត្រូវតែខុសពី *ពាក្យពិសេស* (keyword) មួយចំនួននៅក្នុងភាសា Python ។ ពាក្យពិសេសទាំងនោះមានដូចខាងក្រោមនេះ៖

```
False       class       finally     is          return
None        continue    for         lambda      try
True        def         from        nonlocal    while
and         del         global      not         with
as          elif        if          or          yield
assert      else        import      pass
break       except      in          raise
```

ចំពោះឈ្មោះរបស់វត្ថុដែលជាអក្សរឡាតាំង ឈ្មោះជាអក្សរធំខុសពីឈ្មោះជាអក្សរតូច ដូចជា NUMBER, Number, number គឺជាឈ្មោះខុសៗគ្នា។ ឈ្មោះរបស់វត្ថុគួរតែជាពាក្យមានន័យអ្វីម៉្យាងដែលស្តែងចេញពីតួនាទីរបស់វត្ថុនោះ។

តាមការសង្កេតជាក់ស្តែង ការភ្ជាប់ឈ្មោះណាមួយទៅនឹងវត្ថុណាមួយ គឺជាការដាក់ឈ្មោះឲ្យវត្ថុនោះ ដែលជាទង្វើមួយដូចជាការតាងអញ្ញាតនៅក្នុងគណិតវិទ្យាដែរ។ ម៉្យាងទៀត ក្រោយពីវត្ថុមានឈ្មោះមួយរួចហើយ គ្រប់ការយកឈ្មោះរបស់វត្ថុនោះទៅប្រើនៅពេលក្រោយៗទៀត គឺជាការយកវត្ថុនោះផ្ទាល់ទៅប្រើតែម្តង។

នៅក្នុងកម្មវិធីខាងលើ ក្រោយពីឈ្មោះ ថ្លៃលក់ ត្រូវបានភ្ជាប់ទៅនឹងវត្ថុលេខ 1000 រួចមក វត្ថុចុងក្រោយនេះអាចស្ថិតនៅក្នុងសតិរបស់កំព្យូទ័ររហូតដល់ចប់កម្មវិធី។ ក៏ប៉ុន្តែទោះជាយ៉ាងណាក៏ដោយ ក៏យើងនៅតែមិនអាចមើលឃើញវត្ថុលេខ 1000 នោះដែរ គឺវាមាននៅតែក្នុងសតិរបស់កំព្យូទ័រតែប៉ុណ្ណោះ។ ហើយបើយើងចង់មើលឃើញវត្ថុនោះ យើងត្រូវសរសេរកម្មវិធីដូចខាងក្រោមនេះ៖

```
ថ្លៃលក់ = 1000
print(ថ្លៃលក់)
```

print(*ថ្លៃលក់*) គឺជាបញ្ជាតម្រូវឲ្យសរសេរវត្ថុដែលមានឈ្មោះថា ថ្លៃលក់ នៅលើ *បង្អួចបឋម* (prompt window) ។ បង្អួចបឋមគឺជាបង្អួចម៉្យាងដែលមានតួនាទីជាអ្នកបង្ហាញព័ត៌មានផ្សេង

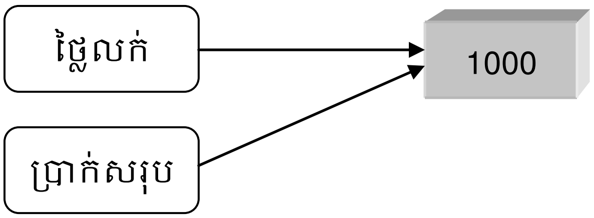
ៗដែលជាលទ្ធផលបានមកពីការប្រើបញ្ជា print នេះ។ ហើយយើងនឹងបានស្គាល់ច្បាស់ពីបញ្ជា print នេះនៅពេលណាដែលយើងធ្វើការសិក្សាពីវត្ថុដែលជាក្បួននៅក្នុងភាសា Python ។ នៅពេលនេះយើងគ្រាន់តែដឹងថា ការសរសេរពាក្យថា print គឺជាបញ្ជាតម្រូវឲ្យបង្ហាញវត្ថុផ្សេងៗនៅលើបង្អួចបឋម។

យើងឃើញថាការដាក់ឈ្មោះឲ្យវត្ថុ គឺជាប្រការដែលធ្វើឲ្យវត្ថុអាចស្ថិតនៅក្នុងសតិរបស់កំព្យូទ័ររហូតដល់ចប់កម្មវិធី ហើយវាជាប្រការដែលធ្វើឲ្យយើងអាចយកវត្ថុនោះទៅប្រើការធ្វើអ្វីផ្សេងៗទៀតបានដែរ។ លើសពីនេះទៀត ក្រៅពីការដាក់ឈ្មោះតែមួយឲ្យទៅវត្ថុណាមួយ យើងក៏អាចដាក់ឈ្មោះជាច្រើនឲ្យទៅវត្ថុនោះដោយសរសេរកម្មវិធីដូចខាងក្រោមនេះ៖

```
ថ្លៃលក់ = 1000
ប្រាក់សរុប = ថ្លៃលក់
print(ថ្លៃលក់)
print(ប្រាក់សរុប)
```

ប្រាក់សរុប = ថ្លៃលក់ គឺជាបញ្ជាតម្រូវឲ្យភ្ជាប់ឈ្មោះ ប្រាក់សរុប មួយទៀតទៅនឹងវត្ថុដែលមានឈ្មោះថា ថ្លៃលក់ មួយរួចទៅហើយ។ ជាលទ្ធផល វត្ថុលេខ 1000 មានឈ្មោះរហូតដល់ទៅពីរ គឺឈ្មោះមួយជា ថ្លៃលក់ និងឈ្មោះមួយទៀតជា ប្រាក់សរុប ។

ការភ្ជាប់ឈ្មោះ ប្រាក់សរុប ទៅនឹងវត្ថុដែលមានឈ្មោះថា ថ្លៃលក់ អាចតាងដោយគំនូសបំព្រួញដូចខាងក្រោមនេះ៖

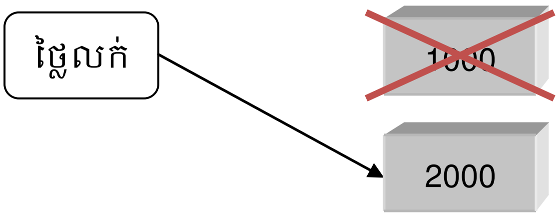


ឈ្មោះរបស់វត្ថុមានតួនាទីសំខាន់សម្រាប់សំគាល់ទីតាំងរបស់វត្ថុនៅក្នុងសតិរបស់កំព្យូទ័រ ហើយបើគ្មានឈ្មោះនោះទេ យើងនឹងគ្មានមធ្យោបាយណាមួយដែលអនុញ្ញាតឲ្យយើងអាចយកវត្ថុទៅប្រើប្រាស់នៅទីកន្លែងផ្សេងៗទៀតបានឡើយ។ ក៏ប៉ុន្តែផ្ទុយមកវិញ យើងអាចដកយកឈ្មោះរបស់វត្ថុណាមួយទៅភ្ជាប់ជាមួយវត្ថុមួយផ្សេងទៀតបានដោយសរសេរកម្មវិធីដូចខាងក្រោមនេះ៖

```
ថ្លៃលក់ = 1000
ថ្លៃលក់ = 2000
print(ថ្លៃលក់)
```

ថ្លៃលក់ = 2000 គឺជាបញ្ជាតម្រូវឲ្យយកឈ្មោះ ថ្លៃលក់ ដែលជាឈ្មោះរបស់លេខ 1000 ទៅភ្ជាប់នឹងវត្ថុលេខ 2000 វិញម្តង។ ជាលទ្ធផល វត្ថុលេខ 1000 ត្រូវបាត់ឈ្មោះ និងត្រូវលុបចេញពីក្នុងសតិរបស់កំព្យូទ័រដោយស្វ័យប្រវត្តិ។ ការលុបវត្ថុរបៀបនេះហៅថា *យន្តការបោសសម្អាត* (garbage collection) ដែលតែងតែកើតមានឡើងនៅពេលដែលវត្ថុត្រូវបាត់ឈ្មោះ។ ម៉្យាងទៀតឈ្មោះ ថ្លៃលក់ បានក្លាយទៅជាឈ្មោះរបស់វត្ថុលេខ 2000 វិញម្តង ដូចនេះគ្រប់ការយកវត្ថុឈ្មោះ ថ្លៃលក់ ទៅប្រើ គឺជាការយកលេខ 2000 ទៅប្រើ មិនមែនជាការយកលេខ 1000 ទៅប្រើឡើយ ពីព្រោះ ថ្លៃលក់ លែងជាឈ្មោះរបស់វត្ថុលេខ 1000 ទៀតហើយ។

ការយកឈ្មោះ ថ្លៃលក់ របស់វត្ថុលេខ 1000 ទៅភ្ជាប់នឹងវត្ថុលេខ 2000 អាចតាងដោយគំនូសបំព្រួញដូចខាងក្រោមនេះ៖

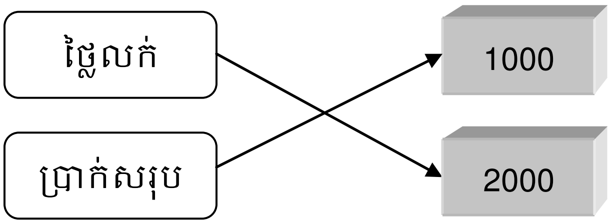


សរុបមក ការយកឈ្មោះរបស់វត្ថុមួយទៅភ្ជាប់នឹងវត្ថុមួយផ្សេងទៀត បើនិយាយឲ្យស្រួលស្តាប់ទៅ គឺជាការយកវត្ថុមកផ្លាស់ប្តូរ។ នៅក្នុងកម្មវិធីខាងលើនេះ មុនដំបូងវត្ថុឈ្មោះ ថ្លៃលក់ គឺជាលេខ 1000 ហើយក្រោយពីការយកឈ្មោះ ថ្លៃលក់ ទៅភ្ជាប់នឹងវត្ថុលេខ 2000 វិញម្តងមក វត្ថុឈ្មោះថ្លៃលក់ គឺជាលេខ 2000 វិញម្តង។ ដូចនេះការយកឈ្មោះ ថ្លៃលក់ ទៅភ្ជាប់នឹងវត្ថុថ្មី គឺជាការផ្លាស់ប្តូរវត្ថុឈ្មោះ ថ្លៃលក់ ពីលេខ 1000 ឲ្យក្លាយទៅជាលេខ 2000 វិញម្តង។ ការធ្វើដូច្នេះអាចនិយាយបានម៉្យាងទៀតថា គឺជាការយកវត្ថុឈ្មោះ ថ្លៃលក់ មកដោះដូរតម្លៃរបស់វា។

មួយវិញទៀត យើងត្រូវធ្វើការកត់សំគាល់ផងដែរថា វត្ថុមួយត្រូវលុបចេញពីក្នុងសតិរបស់កំព្យូទ័រ តែក្នុងករណីដែលវត្ថុនោះត្រូវបាត់ឈ្មោះអស់តែប៉ុណ្ណោះ តែបើវត្ថុនោះមានឈ្មោះលើសពីមួយ ការយកឈ្មោះណាមួយទៅភ្ជាប់នឹងវត្ថុផ្សេងទៀតនឹងមិនបណ្តាលឲ្យវត្ថុនោះត្រូវលុបចេញពីក្នុងសតិរបស់កំព្យូទ័រឡើយ ព្រោះនៅមានឈ្មោះជាច្រើនទៀតនៅជាប់នឹងវត្ថុនោះ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
ថ្លៃលក់ = 1000
ប្រាក់សរុប = ថ្លៃលក់
ថ្លៃលក់ = 2000
print(ថ្លៃលក់)
print(ប្រាក់សរុប)
```

ថ្លៃលក់ = 2000 គឺជាបញ្ជាតម្រូវឲ្យយកឈ្មោះ ថ្លៃលក់ ដែលជាឈ្មោះរបស់វត្ថុលេខ 1000 ទៅភ្ជាប់នឹងវត្ថុលេខ 2000 វិញម្តង។ ប្រការនេះមិនបានធ្វើឲ្យវត្ថុលេខ 1000 ត្រូវលុបចេញពីក្នុងសតិរបស់កំព្យូទ័រឡើយ ពីព្រោះនៅមានឈ្មោះ ប្រាក់សរុប មួយទៀតនៅជាប់នឹងវត្ថុនោះ។



ប្រភេទនៃវត្ថុ

នៅក្នុងភាសា Python ក្រៅពីវត្ថុដែលយើងបានឃើញកន្លងមក នៅមានវត្ថុជាច្រើន *ប្រភេទ* (type) ទៀតដែលយើងអាចបង្កើតឡើងដើម្បីឆ្លើយតបទៅនឹងតម្រូវការនៃការដោះស្រាយបញ្ហាផ្សេងៗ។ វត្ថុទាំងនោះមាន៖

ចំនួនគត់

ចំនួនគត់ (integer) គឺជាលេខទាំងឡាយណាដែលជាលេខគ្មានក្បៀស។ ដូចនេះការបង្កើតវត្ថុនៅក្នុងកម្មវិធីកន្លងមក គឺជាការបង្កើតវត្ថុដែលមានប្រភេទជាចំនួនគត់ ពីព្រោះវត្ថុទាំងនោះគឺជាលេខគ្មានក្បៀស។ យើងគួររំលឹកឡើងវិញថា ដើម្បីបង្កើតចំនួនគត់ យើងត្រូវសរសេរកម្មវិធីដូចខាងក្រោមនេះ៖

```
ថ្លៃលក់ = 1000
ថ្លៃទិញ = 900
print(ថ្លៃលក់)
print(ថ្លៃទិញ)
```

ថ្លៃលក់ = 1000 គឺជាបញ្ជាតម្រូវឲ្យបង្កើតចំនួនគត់មួយមានឈ្មោះថា ថ្លៃលក់ ដែលជាលេខ 1000 ។

ថ្លៃទិញ = 900 គឺជាបញ្ជាតម្រូវឲ្យបង្កើតចំនួនគត់មួយទៀតមានឈ្មោះថា ថ្លៃទិញ ដែលជាលេខ 900 ។

ចាប់ពីភាសា Python 3 ឡើងទៅ ចំនួនគត់គឺជាលេខគ្មានព្រំដែន ពោលគឺវាអាចជាលេខមានប៉ុន្មានខ្ទង់ក៏បានដែរ។

ក្រោយពីចំនួនគត់ផ្សេងៗត្រូវបានបង្កើតឡើងនៅក្នុងសតិរបស់កំព្យូទ័ររួចមក យើងអាចយកវត្ថុទាំងនោះមកធ្វើប្រមាណវិធីផ្សេងៗក្នុងគោលបំណងបង្កើតវត្ថុថ្មីៗទៀតដែលឆ្លើយតបទៅនឹងតម្រូវការរបស់យើងនៅក្នុងការដោះស្រាយបញ្ហា។ ចំនួនគត់អាចត្រូវយកមកធ្វើប្រមាណវិធីមួយចំនួនដូចតទៅនេះ៖

ប្រមាណវិធីនព្វន្ត

ប្រមាណវិធីនព្វន្ត (arithmetic operation) គឺជា ប្រមាណវិធីទាំងឡាយណាដែលនៅក្នុងនោះមានការប្រើប្រាស់ *ប្រមាណសញ្ញា* (operator) +, -, *, /, //, %, ** ។

យើងអាចយកចំនួនគត់ផ្សេងៗមកធ្វើប្រមាណវិធីនព្វន្តដូចខាងក្រោមនេះ៖

```
ថ្លៃលក់ = 1000
ថ្លៃទិញ = 900
print(ថ្លៃលក់ + ថ្លៃទិញ)
print(ថ្លៃលក់ - ថ្លៃទិញ)
print(ថ្លៃលក់ * ថ្លៃទិញ)
print(ថ្លៃលក់ / ថ្លៃទិញ)
print(ថ្លៃលក់ // ថ្លៃទិញ)
print(ថ្លៃលក់ % ថ្លៃទិញ)
print(ថ្លៃលក់ ** ថ្លៃទិញ)
```

ថ្លៃលក់ + ថ្លៃទិញ គឺជាបញ្ជាតម្រូវឲ្យធ្វើ *ប្រមាណវិធីបូក* (addition) រវាងចំនួនគត់ឈ្មោះ ថ្លៃលក់ និងចំនួនគត់ឈ្មោះ ថ្លៃទិញ ។ ការសរសេរ *ថ្លៃលក់ + ថ្លៃទិញ* ហៅថា *កន្សោមប្រមាណវិធី* (expression) ដែលនៅក្នុងនោះមានវត្ថុឈ្មោះ ថ្លៃលក់ និង ថ្លៃទិញ ជា *ប្រមាណអង្គ* (operand) និងសញ្ញាបូក (+) គឺជាប្រមាណសញ្ញា។

នៅក្នុងភាសា Python ការធ្វើប្រមាណវិធីផ្សេងៗមិនទាមទារឲ្យមានសញ្ញាស្មើ (=) ដើម្បីបញ្ជាក់ពីលទ្ធផលបានមកពីការធ្វើប្រមាណវិធីនោះទេ គឺយើងគ្រាន់តែសរសេរកន្សោមប្រមាណវិធី យើងនឹងបានលទ្ធផលជាវត្ថុណាមួយដោយស្វ័យប្រវត្តិ។ យើងគួររំលឹកឡើងវិញថាសញ្ញាស្មើនៅក្នុងភាសា Python ត្រូវប្រើសម្រាប់តែភ្ជាប់ឈ្មោះទៅនឹងវត្ថុតែប៉ុណ្ណោះ។

មួយវិញទៀត ដែលហៅថាកន្សោមប្រមាណវិធី គឺគ្រប់ការសរសេរទាំងឡាយណាដែលនាំឲ្យបានលទ្ធផលជាវត្ថុអ្វីមួយ។ ជាក់ស្តែង ការសរសេរ ថ្លៃលក់ + ថ្លៃទិញ នាំឲ្យបានលទ្ធផលជាចំនួនគត់មួយផ្សេងទៀត ដូចនេះការសរសេររបៀបនេះហៅថាកន្សោមប្រមាណវិធីព្រោះវានាំឲ្យបានលទ្ធផលជាវត្ថុមួយថ្មីទៀត។

ថ្លៃលក់ - ថ្លៃទិញ គឺជាបញ្ជាតម្រូវឲ្យធ្វើ *ប្រមាណវិធីដក* (subtraction) រវាងចំនួនគត់ឈ្មោះ ថ្លៃលក់ និងចំនួនគត់ឈ្មោះ ថ្លៃទិញ ដែលនាំឲ្យបានលទ្ធផលជាចំនួនគត់មួយទៀតត្រូវបានសរសេរនៅលើបង្អួចបឋម។

*ថ្លៃលក់ * ថ្លៃទិញ* គឺជាបញ្ជាតម្រូវឲ្យធ្វើ *ប្រមាណវិធីគុន* (multiplication) រវាងចំនួនគត់ឈ្មោះ ថ្លៃលក់ និងចំនួនគត់ឈ្មោះ ថ្លៃទិញ ដែលនាំឲ្យបានលទ្ធផលជាចំនួនគត់មួយទៀតត្រូវបានសរសេរនៅលើបង្អួចបឋម។

ថ្លៃលក់ / ថ្លៃទិញ គឺជាបញ្ជាតម្រូវឲ្យធ្វើ *ប្រមាណវិធីចែក* (division) រវាងចំនួនគត់ឈ្មោះ ថ្លៃលក់ និងចំនួនគត់ឈ្មោះ ថ្លៃទិញ ដែលនាំឲ្យបានលទ្ធផលជាលេខមានក្បៀស។

នៅក្នុងភាសា Python គ្រប់ការធ្វើប្រមាណវិធីចែកទាំងឡាយផ្តល់លទ្ធផលជាលេខមានក្បៀស។ យើងនឹងធ្វើការសិក្សាលម្អិតពីលេខមានក្បៀសនៅពេលបន្តិចទៀតនេះ។

ថ្លៃលក់ // ថ្លៃទិញ គឺជាបញ្ជាតម្រូវឲ្យធ្វើ *ប្រមាណវិធីចែកបន្ថយ* (floor division) រវាងចំនួនគត់ឈ្មោះ ថ្លៃលក់ និងចំនួនគត់ឈ្មោះ ថ្លៃទិញ ដែលនាំឲ្យបានលទ្ធផលជាចំនួនគត់មួយទៀតត្រូវបានសរសេរនៅលើបង្គួចបបម។

ដែលហៅថាប្រមាណវិធីចែកបន្ថយ គឺជាប្រមាណវិធីទាំងឡាយណាដែលផ្តល់លទ្ធផលជាលេខគត់នៅខាងមុខក្បៀស លេខនៅខាងក្រោយក្បៀសត្រូវលុបចោលបើសិនជាមាន។

ថ្លៃលក់ % ថ្លៃទិញ គឺជាបញ្ជាតម្រូវឲ្យធ្វើ *ប្រមាណវិធីចែកយកសំណល់* (modulus) រវាងចំនួនគត់ឈ្មោះ ថ្លៃលក់ និងចំនួនគត់ឈ្មោះ ថ្លៃទិញ ដែលនាំឲ្យបានលទ្ធផលជាចំនួនគត់មួយទៀតត្រូវបានសរសេរនៅលើបង្គួចបបម។

ដែលហៅថាប្រមាណវិធីចែកយកសំណល់ គឺជាប្រមាណវិធីចែកដែលផ្តល់លទ្ធផលជាសំណល់។

*ថ្លៃលក់ ** ថ្លៃទិញ* គឺជាបញ្ជាតម្រូវឲ្យធ្វើ *ប្រមាណវិធីស្វ័យគុន* (exponentiation) រវាងចំនួនគត់ឈ្មោះ ថ្លៃលក់ និងចំនួនគត់ឈ្មោះ ថ្លៃទិញ ដែលនាំឲ្យបានលទ្ធផលជាចំនួនគត់មួយទៀតត្រូវបានសរសេរនៅលើបង្គួចបបម។ កន្សោមប្រមាណវិធី *ថ្លៃលក់ ** ថ្លៃទិញ* មានន័យថា ថ្លៃលក់ស្វ័យគុន ថ្លៃទិញ ។

អាទិភាពនៃប្រមាណសញ្ញា

ក្រៅពីការយកចំនួនគត់ពីរមកធ្វើប្រមាណវិធី យើងអាចយកចំនួនគត់មានចំនួនប៉ុន្មានក៏បានដែរមកធ្វើប្រមាណវិធីចំរុះគ្នាដើម្បីឲ្យបានលទ្ធផលជាវត្ថុដែលយើងត្រូវការ។ ក្នុងករណីនេះ យើងបានកន្សោមប្រមាណវិធីមួយដែលមានប្រមាណសញ្ញាជាច្រើននៅក្នុងនោះ។ ហើយបើសិនជាប្រមាណសញ្ញាទាំងនោះជាប្រមាណសញ្ញាខុសៗគ្នា ប្រមាណវិធីនឹងត្រូវធ្វើឡើងដោយអនុលោមទៅតាម *អាទិភាពនៃប្រមាណសញ្ញា* (operator precedence) ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
ថ្លៃលក់ = 1000
ថ្លៃទិញ = 900
សោហ៊ុយ = 300
print(ថ្លៃលក់ + ថ្លៃទិញ * សោហ៊ុយ)
```

*ថ្លៃលក់ + ថ្លៃទិញ * សោហ៊ុយ* គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីរវាងចំនួនគត់ឈ្មោះ ថ្លៃលក់ ចំនួនគត់ឈ្មោះ ថ្លៃទិញ និងចំនួនគត់ឈ្មោះ សោហ៊ុយ ។ នៅក្នុងករណីនេះ យើងបានកន្សោមប្រមាណវិធីមួយដែលនៅក្នុងនោះមានប្រមាណអង្គចំនួនបី និងប្រមាណសញ្ញាចំនួនពីរខុសៗគ្នា គឺប្រមាណសញ្ញាបូក (+) និងប្រមាណសញ្ញាគុន (*) ។ អាស្រ័យទៅតាមច្បាប់នៅក្នុងភាសា Python ប្រមាណវិធីត្រូវធ្វើឡើងដោយគោរពទៅតាមអាទិភាពនៃប្រមាណសញ្ញា។ ពោលគឺប្រមាណវិធីគុនត្រូវធ្វើឡើងមុនប្រមាណវិធីបូក។ ដូចនេះប្រមាណវិធីគុនរវាងចំនួនគត់ឈ្មោះ ថ្លៃទិញ និងចំនួនគត់ឈ្មោះ សោហ៊ុយ ត្រូវធ្វើឡើងមុន រួចបានយកលទ្ធផលមកធ្វើប្រមាណវិធីបូកជាមួយនឹងចំនួនគត់ឈ្មោះ ថ្លៃលក់ ។

អាទិភាពនៃប្រមាណសញ្ញានៅក្នុងកន្សោមប្រមាណវិធី មិនមែនជារឿងដែលយើងមិនដែលជួបប្រទះនោះទេ យើងបានដឹងតាំងតែនៅវិទ្យាល័យមកម្លេះថា នៅក្នុងកន្សោមពីជគណិត

ប្រមាណសញ្ញាគុនមានអាទិភាពជាងប្រមាណសញ្ញាបូក។ ម៉្យាងទៀតនៅក្នុងកន្សោមពីជគណិត បើសិនណាជាយើងចង់ផ្តល់អាទិភាពឲ្យទៅប្រមាណសញ្ញាណាមួយ យើងត្រូវប្រើសញ្ញាវង្ងង់ក្រចកនៅអមសងខាងកន្សោមប្រមាណវិធីដែលមានប្រមាណសញ្ញានោះ។ ដូចគ្នាដែរ នៅក្នុងភាសា Python បើសិនជាយើងចង់ផ្តល់អាទិភាពឲ្យទៅប្រមាណសញ្ញាណាមួយនោះ យើងក៏ត្រូវប្រើសញ្ញាវង្ងង់ក្រចកនៅអមសងខាងកន្សោមប្រមាណវិធីដែលមានប្រមាណសញ្ញានោះដែរ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
ថ្លៃលក់ = 1000
ថ្លៃទិញ = 900
សោហ៊ុយ = 300
print((ថ្លៃលក់ + ថ្លៃទិញ) * សោហ៊ុយ)
```

*(ថ្លៃលក់ + ថ្លៃទិញ) * សោហ៊ុយ* គឺជាបញ្ញាតម្រូវឲ្យធ្វើប្រមាណវិធីរវាងចំនួនគត់ឈ្មោះ ថ្លៃលក់ ចំនួនគត់ឈ្មោះ ថ្លៃទិញ និងចំនួនគត់ឈ្មោះ សោហ៊ុយ ។ ដោយសារមានសញ្ញាវង្ងង់ក្រចកនៅអមសងខាងកន្សោមប្រមាណវិធី *ថ្លៃលក់ + ថ្លៃទិញ* ដូចនេះប្រមាណវិធីបូកត្រូវធ្វើឡើងមុនប្រមាណវិធីគុន។

មួយវិញទៀត យើងសង្កេតឃើញថា គ្រប់ការយកចំនួនគត់ផ្សេងៗមកធ្វើប្រមាណវិធី គឺជាការបង្កើតវត្ថុថ្មីផ្សេងទៀត វត្ថុដើមដែលត្រូវបានយកមកធ្វើប្រមាណវិធីទាំងប៉ុន្មាន មិនត្រូវបានកែប្រែឬបាត់បង់ឡើយ។

ចំនួនពិត

ចំនួនពិត (floating-point number) គឺជាលេខទាំងឡាយណាដែលជាលេខមានក្បៀស។ ដើម្បីបង្កើតចំនួនពិត យើងត្រូវសរសេរកម្មវិធីដូចខាងក្រោមនេះ៖

```
ថ្លៃលក់ = 1000.75
ថ្លៃទិញ = 900.33
print(ថ្លៃលក់)
print(ថ្លៃទិញ)
```

ថ្លៃលក់ = 1000.75 គឺជាបញ្ជាតម្រូវឲ្យបង្កើតចំនួនពិតឈ្មោះ ថ្លៃលក់ មួយដែលជាលេខ 1000.75 ។

ថ្លៃទិញ = 900.33 គឺជាបញ្ជាតម្រូវឲ្យបង្កើតចំនួនពិតមួយទៀតមានឈ្មោះថា ថ្លៃទិញ ដែលជាលេខ 900.33 ។

ប្រមាណវិធីនព្វន្ត

ដូចគ្នានឹងចំនួនគត់ដែរ យើងអាចយកចំនួនពិតផ្សេងៗមកធ្វើប្រមាណវិធីនព្វន្តដូចខាងក្រោមនេះ៖

```
ថ្លៃលក់ = 100.75
ថ្លៃទិញ = 90.33
print(ថ្លៃលក់ + ថ្លៃទិញ)
print(ថ្លៃលក់ - ថ្លៃទិញ)
print(ថ្លៃលក់ * ថ្លៃទិញ)
print(ថ្លៃលក់ / ថ្លៃទិញ)
print(ថ្លៃលក់ // ថ្លៃទិញ)
```

```
print(ថ្ងៃលក់ % ថ្ងៃទិញ)
print(ថ្ងៃលក់ ** ថ្ងៃទិញ)
```

ក្រៅពីការធ្វើប្រមាណវិធីនព្វន្តរវាងចំនួនគត់និងចំនួនគត់ ឬរវាងចំនួនពិតនិងចំនួនពិត យើងអាចយកចំនួនគត់ផ្សេងៗមកធ្វើប្រមាណវិធីនព្វន្តជាមួយនឹងចំនួនពិតដែលនាំឲ្យបានលទ្ធផលជាចំនួនពិត។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
ថ្ងៃលក់ = 100
ថ្ងៃទិញ = 90.33
print(ថ្ងៃលក់ + ថ្ងៃទិញ)
print(ថ្ងៃលក់ - ថ្ងៃទិញ)
print(ថ្ងៃលក់ * ថ្ងៃទិញ)
print(ថ្ងៃលក់ / ថ្ងៃទិញ)
print(ថ្ងៃលក់ // ថ្ងៃទិញ)
print(ថ្ងៃលក់ % ថ្ងៃទិញ)
print(ថ្ងៃលក់ ** ថ្ងៃទិញ)
```

តក្កវត្ថុ

តក្កវត្ថុ (Boolean) គឺជាវត្ថុពីរយ៉ាងគឺ True និង False ។ ដើម្បីបង្កើតតក្កវត្ថុ យើងត្រូវធ្វើដូចខាងក្រោមនេះ៖

```
ត្រូវ = True
ខុស = False
print(ត្រូវ)
```

print(*ខុស*)

ត្រូវ = True គឺជាបញ្ជាតម្រូវឲ្យបង្កើតតក្កវត្ថុមានឈ្មោះថា ត្រូវ ដែលជា True ។

ខុស = False គឺជាបញ្ជាតម្រូវឲ្យបង្កើតតក្កវត្ថុមួយទៀតមានឈ្មោះថា ខុស ដែលជា False ។

ជាទូទៅតក្កវត្ថុមានប្រយោជន៍នៅក្នុងការធ្វើប្រមាណវិធីមួយចំនួនមានដូចជាប្រមាណវិធីតក្កវិទ្យាជាដើម។ លើសពីនេះទៀត តក្កវត្ថុអាចជាវត្ថុដែលជាលទ្ធផលបានមកពីការធ្វើប្រមាណវិធីមួយចំនួនទៀតដែលយើងនឹងបានឃើញនៅពេលបន្តិចទៀតនេះ។

ប្រមាណវិធីតក្កវិទ្យា

ប្រមាណវិធីតក្កវិទ្យា (logical operation) គឺជាប្រមាណវិធីទាំងឡាយណាដែលនៅក្នុងនោះមានការប្រើប្រាស់ប្រមាណសញ្ញា and, or, not ។

យើងអាចយកតក្កវត្ថុផ្សេងៗមកធ្វើប្រមាណវិធីតក្កវិទ្យាដូចខាងក្រោមនេះ៖

```
ត្រូវ = True
ខុស = False
print(ត្រូវ and ខុស)
print(ត្រូវ or ខុស)
print(not ត្រូវ)
print(not ខុស)
```

ត្រូវ and *ខុស* គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីតក្កវិទ្យាដោយប្រើប្រមាណសញ្ញា and រវាងតក្កវត្ថុឈ្មោះ ត្រូវ និងតក្កវត្ថុឈ្មោះ ខុស ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ថុ False ។

នៅក្នុងការធ្វើប្រមាណវិធីតក្កវិទ្យាដោយប្រើប្រមាណសញ្ញា and បើប្រមាណអង្គនៅខាងឆ្វេងជាតក្កវត្ថុ True លទ្ធផលគឺជាប្រមាណអង្គនៅខាងស្តាំ។ ផ្ទុយមកវិញ បើប្រមាណអង្គនៅខាងឆ្វេងជាតក្កវត្ថុ False លទ្ធផលគឺជាប្រមាណអង្គនោះតែម្តង។

ត្រូវ or *ខុស* គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីតក្កវិទ្យាដោយប្រើប្រមាណសញ្ញា or រវាងតក្កវត្ថុឈ្មោះ ត្រូវ និងតក្កវត្ថុឈ្មោះ ខុស ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ថុ True ។

នៅក្នុងការធ្វើប្រមាណវិធីតក្កវិទ្យាដោយប្រើប្រមាណសញ្ញា or បើប្រមាណអង្គនៅខាងឆ្វេងជាតក្កវត្ថុ True លទ្ធផលគឺជាប្រមាណអង្គនោះតែម្តង។ ផ្ទុយមកវិញ បើប្រមាណអង្គនៅខាងឆ្វេងជាតក្កវត្ថុ False លទ្ធផលគឺជាប្រមាណអង្គនៅខាងស្តាំ។

not *ត្រូវ* គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីតក្កវិទ្យាដោយប្រើប្រមាណសញ្ញា not ជាមួយនឹងតក្កវត្ថុឈ្មោះ ត្រូវ ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ថុ False ។

នៅក្នុងការធ្វើប្រមាណវិធីតក្កវិទ្យាដោយប្រើប្រមាណសញ្ញា not បើប្រមាណអង្គជាតក្កវត្ថុ True លទ្ធផលគឺជាតក្កវត្ថុ False ។ ផ្ទុយមកវិញ បើប្រមាណអង្គគឺជាតក្កវត្ថុ False លទ្ធផលគឺជាតក្កវត្ថុ True ។

not *ខុស* គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីតក្កវិទ្យាដោយប្រើប្រមាណសញ្ញា not ជាមួយនឹងតក្កវត្ថុឈ្មោះ ខុស ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ថុ True ព្រោះ ខុស គឺជាតក្កវត្ថុ False ។

ដូចគ្នាដែរ យើងអាចយកលេខផ្សេងៗមកធ្វើប្រមាណវិធីតក្កវិទ្យាដូចខាងក្រោមនេះ៖

```
បណ្តោយ = 176
ទទឹង = 0.0
```

```
print(បណ្ដោយ and ទទឹង)
print(បណ្ដោយ or ទទឹង)
print(not បណ្ដោយ)
print(not ទទឹង)
```

បណ្ដោយ and *ទទឹង* គឺជាបញ្ហាតម្រូវឲ្យធ្វើប្រមាណវិធីតក្កវិទ្យាដោយប្រើប្រមាណសញ្ញា and រវាងចំនួនគត់ឈ្មោះ បណ្ដោយ និងចំនួនពិតឈ្មោះ ទទឹង។ ដោយវត្ថុឈ្មោះ បណ្ដោយ ជាលេខខុសពីសូន្យ ដូចនេះវា *សមមូល* (equivalent) នឹងតក្កវត្ត True ពីព្រោះគ្រប់លេខទាំងឡាយណាដែលខុសពីសូន្យសមមូលនឹងតក្កវត្ត True ។ ប្រការនេះធ្វើឲ្យលទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាចំនួនគត់ឈ្មោះ ទទឹង ព្រោះវាជាប្រមាណអង្គនៅខាងស្ដាំ។

បណ្ដោយ or *ទទឹង* គឺជាបញ្ហាតម្រូវឲ្យធ្វើប្រមាណវិធីតក្កវិទ្យាដោយប្រើប្រមាណសញ្ញា or រវាងចំនួនគត់ឈ្មោះ បណ្ដោយ និងចំនួនពិតឈ្មោះ ទទឹង។ ដោយវត្ថុឈ្មោះ បណ្ដោយ សមមូលនឹងតក្កវត្ត True ប្រការនេះធ្វើឲ្យលទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាចំនួនគត់នេះតែម្ដង។

not *បណ្ដោយ* គឺជាបញ្ហាតម្រូវឲ្យធ្វើប្រមាណវិធីតក្កវិទ្យាដោយប្រើប្រមាណសញ្ញា not ជាមួយនឹងចំនួនគត់ឈ្មោះ បណ្ដោយ ។ ដោយ បណ្ដោយ គឺជាលេខសមមូលនឹងតក្កវត្ត True ដូចនេះលទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ត False ។

not *ទទឹង* គឺជាបញ្ហាតម្រូវឲ្យធ្វើប្រមាណវិធីតក្កវិទ្យាដោយប្រើប្រមាណសញ្ញា not ជាមួយនឹងចំនួនពិតឈ្មោះ ទទឹង ។ ដោយ ទទឹង គឺជាលេខសូន្យ ដូចនេះវាសមមូលនឹងតក្កវត្ត False ។ ដូចនេះលទ្ធផលបានមកគឺជាតក្កវត្ត True ។

ប្រមាណវិធីប្រៀបធៀប

ប្រមាណវិធីប្រៀបធៀប (comparison operation) គឺជាប្រមាណវិធីទាំងឡាយណាដែលនៅក្នុងនោះមានការប្រើប្រាស់ប្រមាណសញ្ញា ==, !=, >, <, >=, <= ក្នុងគោលបំណងយកវត្ថុផ្សេងៗមកប្រៀបធៀបគ្នា។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ថុ។

យើងអាចយកលេខផ្សេងៗមកធ្វើប្រមាណវិធីប្រៀបធៀបដូចខាងក្រោមនេះ៖

```
បណ្តោយ = 176
ទទឹង = 23.31
print(បណ្តោយ == ទទឹង)
print(បណ្តោយ != ទទឹង)
print(បណ្តោយ > ទទឹង)
print(បណ្តោយ < ទទឹង)
print(បណ្តោយ >= ទទឹង)
print(បណ្តោយ <= ទទឹង)
```

បណ្តោយ == ទទឹង គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណប្រៀបធៀបដោយប្រើប្រមាណសញ្ញា == ដើម្បីពិនិត្យមើលថាតើ បណ្តោយ ពិតជាស្មើនឹង ទទឹង មែនឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ថុ False បញ្ជាក់ប្រាប់ថា បណ្តោយ មិនស្មើនឹង ទទឹង ទេ។

ជាថ្មីម្តងទៀត យើងត្រូវធ្វើការកត់សំគាល់ថាប្រមាណសញ្ញា == នៅក្នុងភាសា Python មិនមែនជាសញ្ញាប្រើសម្រាប់បញ្ជាក់លទ្ធផលបានមកពីការធ្វើប្រមាណវិធីឡើយ ប្រមាណសញ្ញានេះត្រូវប្រើសម្រាប់តែធ្វើប្រមាណវិធីដើម្បីពិនិត្យមើលថា តើវត្ថុពីរពិតជាមានតម្លៃស្មើគ្នាដែរឬទេ។ ហើយបើសិនជាវត្ថុទាំងពីរនោះមានតម្លៃស្មើគ្នា លទ្ធផលគឺជាតក្កវត្ថុ True បើពុំនោះសោតទេ លទ្ធផលគឺជាកតក្កវត្ថុ False ។

បណ្តោយ != ទទឹង គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណប្រៀបធៀបដោយប្រើប្រមាណសញ្ញា != ដើម្បីពិនិត្យមើលថាតើ បណ្តោយ ពិតជាមិនស្មើនឹង ទទឹង មែនឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ត True បញ្ជាក់ប្រាប់ថា បណ្តោយ ពិតជាមិនស្មើនឹង ទទឹង មែន។

បណ្តោយ > ទទឹង គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណប្រៀបធៀបដោយប្រើប្រមាណសញ្ញា > ដើម្បីពិនិត្យមើលថាតើ បណ្តោយ ពិតជាធំជាង ទទឹង មែនឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ត True បញ្ជាក់ប្រាប់ថា បណ្តោយ ពិតជាធំជាង ទទឹង មែន។

បណ្តោយ < ទទឹង គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណប្រៀបធៀបដោយប្រើប្រមាណសញ្ញា < ដើម្បីពិនិត្យមើលថាតើ បណ្តោយ ពិតជាតូចជាង ទទឹង មែនឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ត False បញ្ជាក់ប្រាប់ថា បណ្តោយ មិនតូចជាង ទទឹង ទេ។

បណ្តោយ >= ទទឹង គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណប្រៀបធៀបដោយប្រើប្រមាណសញ្ញា >= ដើម្បីពិនិត្យមើលថាតើ បណ្តោយ ពិតជាធំជាងឬស្មើនឹង ទទឹង មែនឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ត True បញ្ជាក់ប្រាប់ថា បណ្តោយ ពិតជាធំជាងឬស្មើនឹង ទទឹង មែន ។

បណ្តោយ <= ទទឹង គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណប្រៀបធៀបដោយប្រើប្រមាណសញ្ញា <= ដើម្បីពិនិត្យមើលថាតើ បណ្តោយ ពិតជាតូចជាងឬស្មើនឹង ទទឹង មែនឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ត False បញ្ជាក់ប្រាប់ថា បណ្តោយ ពិតជាតូចជាងឬស្មើនឹង ទទឹង មែន។

ប្រមាណវិធីអត្តសញ្ញាណ

ប្រមាណវិធីអត្តសញ្ញាណ (identification) គឺជាប្រមាណវិធីទាំងឡាយណាដែលនៅក្នុងនោះមានការប្រើប្រាស់ប្រមាណសញ្ញា is និង is not ក្នុងគោលបំណងយកវត្ថុពីរមកពិនិត្យមើលថាតើវាពិតជាវត្ថុតែមួយមែនដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ថុ។

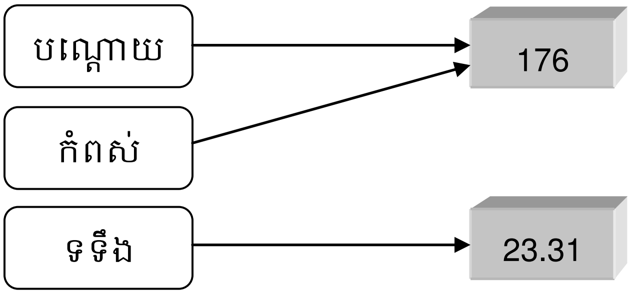
យើងអាចយកវត្ថុផ្សេងៗមកធ្វើប្រមាណវិធីអត្តសញ្ញាណដូចខាងក្រោមនេះ៖

```
បណ្ដោយ = 176
ទទឹង = 23.31
កំពស់ = បណ្ដោយ
print(បណ្ដោយ is ទទឹង)
print(បណ្ដោយ is កំពស់)
print(បណ្ដោយ is not ទទឹង)
```

បណ្ដោយ is ទទឹង គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីអត្តសញ្ញាណដោយប្រើប្រមាណសញ្ញា is ដើម្បីពិនិត្យមើលថាតើ បណ្ដោយ និង ទទឹង គឺជាវត្ថុតែមួយមែនដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ថុ False បញ្ជាក់ប្រាប់ថា បណ្ដោយ និង ទទឹង មិនមែនជាវត្ថុតែមួយនោះទេ។

បណ្ដោយ is កំពស់ គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីអត្តសញ្ញាណដោយប្រើប្រមាណសញ្ញា is ដើម្បីពិនិត្យមើលថាតើ បណ្ដោយ និង កំពស់ គឺជាវត្ថុតែមួយមែនដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ថុ True បញ្ជាក់ប្រាប់ថា បណ្ដោយ និង កំពស់ ពិតជាវត្ថុតែមួយមែន។

បណ្តោយ is not *ទទឹង* គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីអត្តសញ្ញាណដោយប្រើប្រមាណសញ្ញា is not ដើម្បីពិនិត្យមើលថាតើ បណ្តោយ និង ទទឹង មិនមែនជាវត្ថុតែមួយមែនដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ថុ True បញ្ជាក់ប្រាប់ថា បណ្តោយ និង ទទឹង ពិតជាមិនមែនជាវត្ថុតែមួយមែន។



មោឃៈវត្ថុ

None គឺជា *មោឃៈវត្ថុ* ។ ដើម្បីបង្កើតមោឃៈវត្ថុ យើងត្រូវសរសេរកម្មវិធីដូចខាងក្រោមនេះ៖

```
អត់តម្លៃ = None
print(អត់តម្លៃ)
```

អត់តម្លៃ = None គឺជាបញ្ជាតម្រូវឲ្យបង្កើតមោឃៈវត្ថុមួយមានឈ្មោះថា អត់តម្លៃ ។

មោឃៈវត្ថុត្រូវប្រើសម្រាប់តំណាងឲ្យវត្ថុទាំងឡាយណាដែលគ្មានតម្លៃអ្វីទាំងអស់។ ហើយជួនកាលមោឃៈវត្ថុអាចជាលទ្ធផលបានមកពីការធ្វើប្រមាណវិធីផ្សេងៗមានដូចជាការយកក្បួនមកប្រើជាដើម។ យើងនឹងបានស្គាល់ពីការយកក្បួនមកប្រើនៅពេលខាងមុខនេះ។

ក្បួនអក្សរ

កម្រងអក្សរ (string) គឺជាឃ្លាប្រយោគទាំងឡាយណាដែលមានន័យសេចក្តីអ្វីម៉្យាង។ ដើម្បីបង្កើតកម្រងអក្សរ យើងត្រូវធ្វើដូចខាងក្រោមនេះ៖

```
ឃ្លា = "កម្មវិធីជាភាសា Python "
print(ឃ្លា)
```

ឃ្លា = "កម្មវិធីជាភាសា Python " គឺជាបញ្ជាតម្រូវឲ្យបង្កើតកម្រងអក្សរមួយមានឈ្មោះថា ឃ្លា ដែលជាប្រយោគមានន័យថា "កម្មវិធីជាភាសា Python " ។

ដូចនេះយើងឃើញថាដើម្បីបង្កើតកម្រងអក្សរ យើងត្រូវប្រើសញ្ញា " " នេះនៅអមសងខាងឃ្លាប្រយោគណាមួយ។ តែលើសពីនេះទៀត យើងក៏អាចប្រើសញ្ញា ' ' នេះដើម្បីបង្កើតកម្រងអក្សរបានដែរ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
ឃ្លា១ = "អ្នកប្រាជ្ញរក្សាខ្លៅ"
ឃ្លា២ = 'អ្នកមានរក្សាក្សត់'
print(ឃ្លា១)
print(ឃ្លា២)
```

ឃ្លា១ = "អ្នកប្រាជ្ញរក្សាខ្លៅ" គឺជាបញ្ជាតម្រូវឲ្យបង្កើតកម្រងអក្សរមួយមានឈ្មោះថា ឃ្លា១ ដោយប្រើសញ្ញា " " នៅអមសងខាងប្រយោគមួយដែលមានន័យថា "អ្នកប្រាជ្ញរក្សាខ្លៅ" ។

ឃ្លា២ = 'អ្នកមានរក្សាក្សត់' គឺជាបញ្ជាតម្រូវឲ្យបង្កើតកម្រងអក្សរមួយមានឈ្មោះថា ឃ្លា២ ដោយប្រើសញ្ញា ' ' នៅអមសងខាងប្រយោគមួយដែលមានន័យថា 'អ្នកមានរក្សាក្សត់' ។

ដោយហេតុថាយើងអាចប្រើសញ្ញារហូតដល់ទៅពីរបែបនៅក្នុងការបង្កើតកម្រងអក្សរ ដូចនេះ យើងអាចប្រើសញ្ញាណាមួយជាតួអក្សរនៅក្នុងឃ្លានៅពេលដែលយើងប្រើសញ្ញាមួយទៀតនៅអមសងខាងឃ្លានោះ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
ប្រយោគ = 'សរសេរកម្មវិធីគឺជា "ការដោះស្រាយ" បញ្ហាទាំងពួង'
print(ប្រយោគ)
```

ប្រយោគ = 'សរសេរកម្មវិធីគឺជា "ការដោះស្រាយ" បញ្ហាទាំងពួង' គឺជាបញ្ជាតម្រូវឲ្យបង្កើតកម្រងអក្សរមួយមានឈ្មោះថា ប្រយោគ ដែលនៅក្នុងនោះមានសញ្ញា " " ជាតួអក្សរ។

ដើម្បីបង្កើតកម្រងអក្សរដែលមានច្រើនបន្ទាត់ យើងត្រូវប្រើសញ្ញា ''' ''' ឬសញ្ញា """ """ នៅអមសងខាងអត្ថបទណាមួយ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
ពាក្យចាស់ = """អ្នកមានរក្សាក្បត់ ដូចសំពត់ព័ទ្ធពីកក្រៅ
អ្នកប្រាជ្ញរក្សាខ្ញៅ ដូចសំពៅនៅសំប៉ាន។"""
កថា = '''ការសរសេរកម្មវិធីកំព្យូទ័រគឺជាការសរសេររៀបចំ
បញ្ជាមួយចំនួនតាមលំដាប់លំដោយ។'''
print(ពាក្យចាស់)
print(កថា)
```

នៅក្នុងកម្មវិធីខាងលើនេះ មានការបង្កើតកម្រងអក្សរចំនួនពីរគឺកម្រងអក្សរមានឈ្មោះថា ពាក្យចាស់ និងកម្រងអក្សរមានឈ្មោះថា កថា ។ ការបង្កើតកម្រងអក្សរមានឈ្មោះថា ពាក្យចាស់ ត្រូវបានធ្វើឡើងដោយប្រើសញ្ញា """ """ នៅអមសងខាងអត្ថបទមួយ ចំណែកការបង្កើតកម្រងអក្សរមានឈ្មោះថា កថា វិញត្រូវបានធ្វើឡើងដោយប្រើសញ្ញា ''' ''' នៅអមសងខាងអត្ថបទមួយទៀត។

ចំពោះអក្សរពិសេសមួយចំនួនដែលមាននៅលើខ្ទង់កំព្យូទ័រដូចជា៖ Tab, Backspace, Enter ជាដើម យើងត្រូវប្រើសញ្ញា \ រួមផ្សំជាមួយនឹងអក្សរណាមួយទៀតដើម្បីបង្កើតអក្សរពិសេសទាំងនោះ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
ពាក្យចាស់ = """អ្នកមានរក្សាក្សត់\tដូចសំពត់ព័ទ្ធពីកក្រៅ\nអ្នកប្រាជ្ញរក្សាខ្ញៅ\tដូចសំពៅនៅសំប៉ាន។"""
print(ពាក្យចាស់)
```

ពាក្យចាស់ = """អ្នកមានរក្សាក្សត់\tដូចសំពត់ព័ទ្ធពីកក្រៅ\nអ្នកប្រាជ្ញរក្សាខ្ញៅ\tដូចសំពៅនៅសំប៉ាន។""" គឺជាបញ្ជាតម្រូវឲ្យបង្កើតកម្រងអក្សរមួយមានឈ្មោះថា ពាក្យចាស់ ដែលនៅក្នុងនោះមាន \t ជាអក្សរសម្រាប់ដកឃ្លាវែង (Tab) និង \n ជាអក្សរសម្រាប់ចុះបន្ទាត់។

ក្រៅពីអក្សរ \t និង \n នេះ នៅមានអក្សរពិសេសមួយចំនួនទៀតដែលយើងអាចយកមកប្រើដោយសរសេរតាមលំនាំដូចគ្នានេះដែរ។ អក្សរពិសេសទាំងនោះមាន៖

\b Backspace

\r Enter

\" "

\' '

\b សម្លេងជួង

ផ្ទុយមកវិញ បើសិនជាយើងចង់ប្រើប្រាស់សញ្ញា \ នេះជាតួអក្សរធម្មតានៅក្នុងកម្រងអក្សរណាមួយ យើងត្រូវតែសរសេរអក្សរ R ឬ r នៅខាងមុខកម្រងអក្សរនោះ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
ឃ្លា = R"ឯកសារសំខាន់ស្ថិតនៅក្នុងថត c:\n\ឯកសារសំងាត់"
```

```
print(ឃ្លា)
```

ឃ្លា = R"ឯកសារសំខាន់ស្ថិតនៅក្នុងថត c:\n\ឯកសារសំងាត់" គឺជាបញ្ជាតម្រូវឲ្យបង្កើតកម្រង អក្សរមួយមានឈ្មោះថា ឃ្លា ដែលនៅក្នុងនោះមានសញ្ញា \ គឺជាអក្សរធម្មតា។ ហើយគឺដោយសារអក្សរ R នៅខាងដើមកម្រងអក្សរនោះហើយ ដែលធ្វើឲ្យសញ្ញា \n ក្លាយទៅជាអក្សរធម្មតា បើគ្មានអក្សរ R នេះទេ \n នឹងក្លាយទៅជាអក្សរសម្រាប់ចុះបន្ទាត់។

ប្រមាណវិធីបូកបន្ត

ប្រមាណវិធីបូកបន្ត (concatenation) គឺជាប្រមាណវិធីទាំងឡាយណាដែលនៅក្នុងនោះមាន ការប្រើប្រាស់ប្រមាណសញ្ញា + ដើម្បីចម្លងយកវត្ថុផ្សេងៗមកដាក់បន្តគ្នាបង្កើតជាវត្ថុថ្មីមួយទៀត។

យើងអាចយកកម្រងអក្សរផ្សេងៗមកធ្វើប្រមាណវិធីបូកបន្តដូចខាងក្រោមនេះ៖

```
ឃ្លា១ = "អ្នកប្រាជ្ញវក្សាខ្ញៅ\t"
ឃ្លា២ = "ដូចសំពៅនៅសំប៉ាន។"
print(ឃ្លា១ + ឃ្លា២)
```

ឃ្លា១ + ឃ្លា២ គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីបូកបន្តដោយចម្លងយកម្រងអក្សរឈ្មោះ ឃ្លា១ និង ឃ្លា២ មកដាក់បន្តគ្នាបង្កើតជាកម្រងអក្សរថ្មីមួយទៀត។

ប្រមាណវិធីគុនបន្ត

ប្រមាណវិធីគុនបន្ត (repetition) គឺជាប្រមាណវិធីទាំងឡាយណាដែលនៅក្នុងនោះមាន ការប្រើប្រាស់ប្រមាណសញ្ញា * ដើម្បីចម្លងយកវត្ថុណាមួយមកគុនឲ្យបានច្រើនរួចដាក់បន្តគ្នា បង្កើតជាវត្ថុថ្មីមួយទៀត។

យើងអាចយកកម្រងអក្សរផ្សេងៗមកធ្វើប្រមាណគុនបន្តដូចខាងក្រោននេះ៖

```
ប្រយោគដើម = "តក់ៗពេញបំពង់\t"
print(ប្រយោគដើម * 3)
```

ប្រយោគដើម * 3 គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីគុនបន្តដោយចម្លងយកកម្រងអក្សរមាន ឈ្មោះថា ប្រយោគដើម មកគុនបង្កើតជាកម្រងអក្សរដូចគ្នាចំនួនបី រួចដាក់បន្តគ្នាបង្កើតជា កម្រងអក្សរថ្មីមួយទៀត។

ប្រមាណវិធីលេខរៀង

កម្រងអក្សរគឺជា *សមាសវត្ថុ* (container) បានន័យថាកម្រងអក្សរគឺជាវត្ថុមួយដែលនៅក្នុងនោះ មានវត្ថុជាច្រើនទៀត។ វត្ថុនៅក្នុងកម្រងអក្សរគឺជាតួអក្សរផ្សេងៗ។ តួអក្សរទាំងនោះត្រូវចាត់ ទុកថាជា *ធាតុ* (element) នៃកម្រងអក្សរ។ តួអក្សរនៅក្នុងកម្រងអក្សរត្រូវដាក់តម្រៀបជួរគ្នា តាម *លេខរៀង* (index) ចាប់ពីសូន្យឡើងទៅ បើរាប់ពីដើមកម្រងអក្សរឡើងទៅ តែបើយើង រាប់ចាប់ពីចុងកម្រងអក្សរចុះមកវិញ តួអក្សរនៅខាងចុងគេមានលេខរៀង -1 ។ ជាក់ស្តែង នៅ ក្នុងកម្រងអក្សរ Python តួអក្សរ P, y, t, h, o, n តម្រៀបជួរគ្នាចាប់សូន្យឡើងទៅរហូតដល់ 5 បើយើងរាប់ចាប់ពីអក្សរ P ឡើងទៅ តែបើយើងរាប់ចាប់អក្សរ n ចុះមកវិញ តួអក្សរទាំងនោះ តម្រៀបជួរគ្នាចាប់ពី -1 ចុះមករហូតដល់ -6 ។

0	1	2	3	4	5
P	y	t	h	o	n
-6	-5	-4	-3	-2	-1

ម៉្យាងទៀតកម្រងអក្សរក៏មានលក្ខណៈដូចជាកម្រងនៃវត្ថុផ្សេងៗទៀតដែរ ពោលគឺវាជា *កម្រង* (sequence) មួយដែលនៅក្នុងនោះអាចមានវត្ថុផ្សេងៗទៀតជាច្រើនដែលត្រូវក្រងភ្ជាប់គ្នាពីមួយទៅមួយតម្រៀបគ្នាជាជួរត្រង់ដូចបន្ទាត់។ យើងអាចប្រៀបប្រដូចកម្រងអក្សរទៅនឹងត្រណោតនៃវត្ថុផ្សេងៗ។

ប្រមាណវិធីលេខរៀង (indexing) គឺជាប្រមាណវិធីទាំងឡាយណាដែលនៅក្នុងនោះមានការប្រើប្រាស់ប្រមាណសញ្ញា [ ] ដើម្បីចម្លងយកធាតុណាមួយនៅក្នុងសមាសវត្ថុណាមួយមកប្រើការ។

យើងអាចយកកម្រងអក្សរផ្សេងៗមកធ្វើប្រមាណវិធីលេខរៀងដូចខាងក្រោមនេះៈ

```
ឃ្លាដើម = "តក់ៗពេញបំពង់"
print(ឃ្លាដើម[5])
print(ឃ្លាដើម[-3])
print(ឃ្លាដើម[0])
```

ឃ្លាដើម[5] គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីលេខរៀងដើម្បីចម្លងយកតួអក្សរមានលេខរៀង 5 នៅក្នុងកម្រងអក្សរឈ្មោះ ឃ្លាដើម ។

ឃ្លាដើម[-3] គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីលេខរៀងដើម្បីចម្លងយកតួអក្សរមានលេខរៀង -3 នៅក្នុងកម្រងអក្សរឈ្មោះ ឃ្លាដើម ។

ឃ្លាដើម[0] គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីលេខរៀងដើម្បីចម្លងយកតួអក្សរមានលេខរៀង 0 នៅក្នុងកម្រងអក្សរឈ្មោះ ឃ្លាដើម ។

ប្រមាណវិធីកាត់ចម្លង

ប្រមាណវិធីកាត់ចម្លង (slicing) គឺជាប្រមាណវិធីទាំងឡាយណាដែលនៅក្នុងនោះមានការប្រើប្រាស់ប្រមាណសញ្ញា [:] ឬ [::] ដើម្បីចម្លងយកធាតុមួយចំនួនពីក្នុងកម្រងណាមួយមកបង្កើតជាកម្រងថ្មីមួយទៀត។

យើងអាចយកកម្រងអក្សរផ្សេងៗមកធ្វើប្រមាណវិធីកាត់ចម្លងដូចខាងក្រោមនេះ៖

```
ពាក្យចាស់ = "អ្នកប្រាជ្ញវក្សាខ្មៅដូចសំពៅនៅសំប៉ាន។"
print(ពាក្យចាស់[3:30])
print(ពាក្យចាស់[:30])
print(ពាក្យចាស់[3:])
print(ពាក្យចាស់[:])
print(ពាក្យចាស់[3:30:2])
```

ពាក្យចាស់[3:30] គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីកាត់ចម្លងដើម្បីចម្លងយកតួអក្សរមួយចំនួនមានលេខរៀងចាប់ពី 3 រហូតដល់ 29 នៅក្នុងកម្រងអក្សរឈ្មោះ ពាក្យចាស់ មកបង្កើតជាកម្រងអក្សរមួយថ្មីទៀត។

ពាក្យចាស់[:30] គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីកាត់ចម្លងដើម្បីចម្លងយកតួអក្សរមួយចំនួនមានលេខរៀងចាប់ពី 0 រហូតដល់ 29 នៅក្នុងកម្រងអក្សរឈ្មោះ ពាក្យចាស់ មកបង្កើតជាកម្រងអក្សរមួយថ្មីទៀត។

ពាក្យចាស់[3:] គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីកាត់ចម្លងដើម្បីចម្លងយកតួអក្សរមួយចំនួនមានលេខរៀងចាប់ពី 3 រហូតដល់អក្សរនៅខាងចុងគេបំផុតនៅក្នុងកម្រងអក្សរឈ្មោះ ពាក្យចាស់ មកបង្កើតជាកម្រងអក្សរមួយថ្មីទៀត។

ពាក្យចាស់[:] គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីកាត់ចម្លងដើម្បីចម្លងយកតួអក្សរទាំងអស់នៅក្នុងកម្រងអក្សរឈ្មោះ ពាក្យចាស់ មកបង្កើតជាកម្រងអក្សរមួយថ្មីទៀត។

ពាក្យចាស់[3:30:2] គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីកាត់ចម្លងដើម្បីចម្លងយកតួអក្សរមួយចំនួនមានលេខរៀងចាប់ពី 3 រហូតដល់ 29 ដោយរំលងអក្សរចំនួនមួយចោលរហូត នៅក្នុងកម្រងអក្សរឈ្មោះ ពាក្យចាស់ មកបង្កើតជាកម្រងអក្សរមួយថ្មីទៀត។

ដូចនេះបើ ក ខ គ ជាចំនួនគត់ និង ឃ្លា ជាឈ្មោះរបស់កម្រងអក្សរណាមួយ ការសរសេរៈ

- ឃ្លា[ក:ខ] គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីកាត់ចម្លង ដើម្បីចម្លងយកអក្សរមួយចំនួនមានលេខរៀងចាប់ពី ក រហូតដល់ ខ-1 នៅក្នុងកម្រងអក្សរឈ្មោះ ឃ្លា មកបង្កើតជាកម្រងអក្សរថ្មីមួយទៀត។
- ឃ្លា[:ខ] គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីកាត់ចម្លង ដើម្បីចម្លងយកអក្សរមួយចំនួនមានលេខរៀងចាប់ពី 0 រហូតដល់ ខ-1 នៅក្នុងកម្រងអក្សរឈ្មោះ ឃ្លា មកបង្កើតជាកម្រងអក្សរថ្មីមួយទៀត។
- ឃ្លា[ក:] គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីកាត់ចម្លង ដើម្បីចម្លងយកអក្សរមួយចំនួនមានលេខរៀងចាប់ពី ក រហូតដល់អក្សរចុងគេបំផុតនៅក្នុងកម្រងអក្សរឈ្មោះ ឃ្លា មកបង្កើតជាកម្រងអក្សរថ្មីមួយទៀត។
- ឃ្លា[:] គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីកាត់ចម្លង ដើម្បីចម្លងយកអក្សរទាំងអស់នៅក្នុងកម្រងអក្សរឈ្មោះ ឃ្លា មកបង្កើតជាកម្រងអក្សរថ្មីមួយទៀត។
- ឃ្លា[ក:ខ:គ] គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីកាត់ចម្លង ដើម្បីចម្លងយកអក្សរមានលេខរៀងចាប់ពី ក រហូតដល់ ខ-1 ដោយរំលងអក្សរចំនួន គ-1 ចោលរហូត នៅក្នុងកម្រងអក្សរឈ្មោះ ឃ្លា មកបង្កើតជាកម្រងអក្សរថ្មីមួយទៀត។

ប្រមាណវិធីតក្កវិទ្យា

យើងអាចយកកម្រងអក្សរផ្សេងៗមកធ្វើប្រមាណវិធីតក្កវិទ្យាដូចខាងក្រោមនេះៈ

```
ឃ្លាទទេ = ""
ពាក្យចាស់ = "តក់ៗពេញបំពង់"
print(ឃ្លាទទេ and ពាក្យចាស់)
print(ឃ្លាទទេ or ពាក្យចាស់)
print(not ឃ្លាទទេ)
print(not ពាក្យចាស់)
```

ឃ្លាទទេ = "" គឺជាបញ្ជាតម្រូវឲ្យបង្កើតកម្រងអក្សរឈ្មោះ ឃ្លាទទេ មួយដែលជាកម្រងអក្សរគ្មានអក្សរនៅក្នុងនោះ។

យើងត្រូវធ្វើការកត់សំគាល់ថា អក្សរដកឃ្លា (space) ក៏ជាតួអក្សរមួយដូចជាតួអក្សរដទៃទៀតដែរ។ ពោលគឺវាមិនមែនជាអក្សរទទេនោះទេ។ នៅក្នុងកម្រងអក្សរ អក្សរដកឃ្លាមាននាទីជាអ្នកធ្វើឲ្យមានគំលាតរវាងតួអក្សរផ្សេងៗទៀត។

ឃ្លាទទេ and *ពាក្យចាស់* គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីតក្កវិទ្យាដោយប្រើប្រមាណសញ្ញា and រវាងកម្រងអក្សរឈ្មោះ ឃ្លាទទេ និងកម្រងអក្សរឈ្មោះ ពាក្យចាស់ ។ ដោយកម្រងអក្សរឈ្មោះ ឃ្លាទទេ ជាកម្រងអក្សរទទេ ដូចនេះវាសមមូលនឹងតក្កវត្ត False ដែលនាំឲ្យលទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាកម្រងអក្សរឈ្មោះ ឃ្លាទទេ នោះតែម្តង។

ឃ្លាទទេ or *ពាក្យចាស់* គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីតក្កវិទ្យាដោយប្រើប្រមាណសញ្ញា or រវាងកម្រងអក្សរឈ្មោះ ឃ្លាទទេ និងកម្រងអក្សរឈ្មោះ ពាក្យចាស់ ។ ដោយកម្រងអក្សរឈ្មោះ ឃ្លាទទេ ជាកម្រងអក្សរទទេ ដូចនេះវាសមមូលនឹងតក្កវត្ត False ដែលនាំឲ្យលទ្ធផល

បានមកពីប្រមាណវិធីនេះគឺជាកម្រងអក្សរឈ្មោះ ពាក្យចាស់ ព្រោះវាជាប្រមាណអង្គនៅខាងស្ដាំ។

not *ឃ្លាទទេ* គឺជាបញ្ហាតម្រូវឲ្យធ្វើប្រមាណវិធីតក្កវិទ្យាដោយប្រើប្រមាណសញ្ញា not ជាមួយនឹងកម្រងអក្សរឈ្មោះ ឃ្លាទទេ ។ ដោយកម្រងអក្សរឈ្មោះ ឃ្លាទទេ ជាកម្រងអក្សរទទេ ដូចនេះវាសមមូលនឹងតក្កវត្ត False ដែលនាំឲ្យលទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ត True ។

not *ពាក្យចាស់* គឺជាបញ្ហាតម្រូវឲ្យធ្វើប្រមាណវិធីតក្កវិទ្យាដោយប្រើប្រមាណសញ្ញា not ជាមួយនឹងកម្រងអក្សរឈ្មោះ ពាក្យចាស់ ។ ដោយកម្រងអក្សរឈ្មោះ ពាក្យចាស់ មិនមែនជាកម្រងអក្សរទទេ ដូចនេះវាសមមូលនឹងតក្កវត្ត True ដែលនាំឲ្យលទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ត False ។

ប្រមាណវិធីប្រៀបធៀប

យើងអាចយកកម្រងអក្សរផ្សេងៗមកធ្វើប្រមាណវិធីប្រៀបធៀបដូចខាងក្រោមនេះ៖

```
នាម១ = "លឹម កុសល"
នាម២ = "កែវ សំណាង"
print(នាម១ == នាម២)
print(នាម១ != នាម២)
print(នាម១ > នាម២)
print(នាម១ < នាម២)
print(នាម១ >= នាម២)
print(នាម១ <= នាម២)
```

នាម១ == នាម២ គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីប្រៀបធៀបដើម្បីពិនិត្យមើលថាតើកម្រង អក្សរឈ្មោះ នាម១ ពិតជាដូចទៅនឹងកម្រងអក្សរឈ្មោះ នាម២ មែនដែរឬទេ។ លទ្ធផលបាន មកពីប្រមាណវិធីនេះគឺជាតក្កវត្ត False បញ្ជាក់ប្រាប់ថាកម្រងអក្សរទាំងពីរនោះមិនដូចគ្នាទេ។ កម្រងអក្សរពីរដូចគ្នាគឺជាកម្រងអក្សរដែលមានអក្សរដូចគ្នាទាំងអស់។

នាម១ != នាម២ គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីប្រៀបធៀបដើម្បីពិនិត្យមើលថាតើកម្រង អក្សរឈ្មោះ នាម១ ពិតជាមិនដូចទៅនឹងកម្រងអក្សរឈ្មោះ នាម២ មែនដែរឬទេ។ លទ្ធផល បានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ត True បញ្ជាក់ប្រាប់ថាកម្រងអក្សរទាំងពីរនោះពិតជាមិន ដូចគ្នាមែន។

នាម១ > នាម២ គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីប្រៀបធៀបដើម្បីពិនិត្យមើលថាតើកម្រង អក្សរឈ្មោះ នាម១ ពិតជាធំជាងកម្រងអក្សរឈ្មោះ នាម២ មែនដែរឬទេ។ លទ្ធផលបានមកពី ប្រមាណវិធីនេះគឺជាតក្កវត្ត True បញ្ជាក់ប្រាប់ថាកម្រងអក្សរឈ្មោះ នាម១ ពិតជាធំជាង កម្រងអក្សរឈ្មោះ នាម២ មែន។

កម្រងអក្សរឈ្មោះ នាម១ ធំជាងកម្រងអក្សរឈ្មោះ នាម២ បានន័យថា កម្រងអក្សរឈ្មោះ នាម១ មានលំដាប់ថ្នាក់នៅក្រោយកម្រងអក្សរឈ្មោះ នាម២ បើសិនជាយើងយកកម្រងអក្សរ ទាំងពីរនោះទៅតម្រៀបជាពាក្យនៅក្នុងវចនានុក្រមណាមួយ។ ហើយការធ្វើប្រមាណវិធី ប្រៀបធៀបដើម្បីកំណត់ពីទីតាំងនៃវត្ថុផ្សេងៗធៀបទៅនឹងពាក្យនៅក្នុងវចនានុក្រម គឺជា ការធ្វើប្រមាណវិធីប្រៀបធៀប *តាមរបៀបវចនានុក្រម* (lexicographically) ។ យើងនឹងឃើញ នៅពេលក្រោយៗទៀតថា ការធ្វើប្រមាណវិធីប្រៀបធៀបរវាងសមាសវត្ថុមួយចំនួនទៀត ក៏ជា កាធ្វើប្រមាណវិធីប្រៀបធៀបតាមវចនានុក្រមដូចគ្នានេះដែរ។

នាម១ < នាម២ គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីប្រៀបធៀបដើម្បីពិនិត្យមើលថាតើកម្រងអក្សរឈ្មោះ នាម១ ពិតជាតូចជាងកម្រងអក្សរឈ្មោះ នាម២ មែនដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ថុ False បញ្ជាក់ប្រាប់ថាកម្រងអក្សរឈ្មោះ នាម១ មិនតូចជាងកម្រងអក្សរឈ្មោះ នាម២ ទេ។

នាម១ >= នាម២ គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីប្រៀបធៀបដើម្បីពិនិត្យមើលថាតើកម្រងអក្សរឈ្មោះ នាម១ ពិតជាធំជាងឬស្មើនឹងកម្រងអក្សរឈ្មោះ នាម២ មែនដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ថុ True បញ្ជាក់ប្រាប់ថាកម្រងអក្សរឈ្មោះ នាម១ ពិតជាធំជាងឬស្មើនឹងកម្រងអក្សរឈ្មោះ នាម២ មែន។

នាម១ <= នាម២ គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីប្រៀបធៀបដើម្បីពិនិត្យមើលថាតើកម្រងអក្សរឈ្មោះ នាម១ ពិតជាតូចជាងឬស្មើនឹងកម្រងអក្សរឈ្មោះ នាម២ មែនដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ថុ False បញ្ជាក់ប្រាប់ថាកម្រងអក្សរឈ្មោះ នាម១ មិនតូចជាងឬស្មើនឹងកម្រងអក្សរឈ្មោះ នាម២ ទេ។

ប្រមាណវិធីអត្តសញ្ញាណ

យើងអាចយកកម្រងអក្សរផ្សេងៗមកធ្វើប្រមាណវិធីអត្តសញ្ញាណដូចខាងក្រោមនេះ៖

```
ឃ្លា១ = "តក់ៗពេញបំពង់"
ឃ្លា២ = "សេចក្តីព្យាយាមគង់បានសម្រេច"
print(ឃ្លា១ is ឃ្លា២)
print(ឃ្លា១ is not ឃ្លា២)
```

ឃ្លា១ is *ឃ្លា២* គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីអត្តសញ្ញាណ ដើម្បីពិនិត្យមើលថាតើកម្រងអក្សរឈ្មោះ ឃ្លា១ និងកម្រងអក្សរឈ្មោះ ឃ្លា២ ជាវត្ថុតែមួយមែនដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ថុ False បញ្ជាក់ប្រាប់ថា កម្រងអក្សរឈ្មោះ ឃ្លា១ និងកម្រងអក្សរឈ្មោះ ឃ្លា២ មិនមែនជាកម្រងអក្សរតែមួយនោះទេ។

ឃ្លា១ is not *ឃ្លា២* គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីអត្តសញ្ញាណ ដើម្បីពិនិត្យមើលថាតើកម្រងអក្សរឈ្មោះ ឃ្លា១ និងកម្រងអក្សរឈ្មោះ ឃ្លា២ ពិតជាមិនមែនជាវត្ថុតែមួយមែនដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ថុ True បញ្ជាក់ប្រាប់ថា កម្រងអក្សរឈ្មោះ ឃ្លា១ និងកម្រងអក្សរឈ្មោះ ឃ្លា២ ពិតជាមិនមែនជាកម្រងអក្សរតែមួយមែន។

ប្រមាណវិធីរកធាតុ

ប្រមាណវិធីរកធាតុ (membership) គឺជាប្រមាណវិធីទាំងឡាយណាដែលនៅក្នុងនោះមានការប្រើប្រាស់ប្រមាណសញ្ញា in និងឬ not in ដើម្បីពិនិត្យមើលថាតើនៅក្នុងសមាសវត្ថុណាមួយមានធាតុដូចទៅនឹងវត្ថុណាមួយនោះដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ថុ។

យើងអាចយកកម្រងអក្សរផ្សេងៗមកធ្វើប្រមាណវិធីរកធាតុដូចខាងក្រោមនេះ៖

```
ឃ្លា = "តក់ៗពេញបំពង់"
ពាក្យ = "បំពង់"
print(ពាក្យ in ឃ្លា)
print(ពាក្យ not in ឃ្លា)
```

ពាក្យ in *ឃ្លា* គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីរកធាតុដោយប្រើប្រមាណសញ្ញា in ដើម្បីពិនិត្យមើលថាតើនៅក្នុងកម្រងអក្សរឈ្មោះ ឃ្លា មានពាក្យណាមួយដូចទៅនឹងកម្រងអក្សរឈ្មោះ

ពាក្យ នោះដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ត True បញ្ជាក់ប្រាប់ថា នៅក្នុងកម្រងអក្សរឈ្មោះ ឃ្លា ពិតជាមានពាក្យដូចទៅនឹងកម្រងអក្សរឈ្មោះ ពាក្យ នោះមែន។

ពាក្យ not in *ឃ្លា* គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីរកធាតុដោយប្រើប្រមាណសញ្ញា not in ដើម្បីពិនិត្យមើលថាតើនៅក្នុងកម្រងអក្សរឈ្មោះ ឃ្លា គ្មានពាក្យណាមួយដូចទៅនឹងកម្រងអក្សរឈ្មោះ ពាក្យ នោះមែនដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ត False បញ្ជាក់ប្រាប់ថា ប្រការដែលថានៅក្នុងកម្រងអក្សរឈ្មោះ ឃ្លា គ្មានពាក្យណាមួយដូចទៅនឹងកម្រងអក្សរឈ្មោះ ពាក្យ នោះគឺខុស។

កំណត់ពន្យល់

កំណត់ពន្យល់ (comment) គឺជាឃ្លាប្រយោគទាំងឡាយណាដែលយើងសរសេរនៅក្នុងកម្មវិធីដើម្បីធ្វើការពន្យល់ពីការប្រើបញ្ជាផ្សេងៗ។ ឃ្លាប្រយោគទាំងនោះត្រូវតែមានសញ្ញា # នេះនៅពីមុខ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
#ការបង្កើតកម្រងអក្សរ
ឃ្លា = "តក់ៗពេញបំពង់"
ពាក្យ = "បំពង់"
print(ពាក្យ in ឃ្លា) #ការធ្វើប្រមាណវិធីរកធាតុ
```

#ការបង្កើតកម្រងអក្សរ គឺជាកំណត់ពន្យល់ប្រើសម្រាប់ពន្យល់ពីការប្រើបញ្ជាបន្ទាប់ពីនោះ។

#ការធ្វើប្រមាណវិធីរកធាតុ គឺជាកំណត់ពន្យល់មួយទៀតប្រើសម្រាប់ពន្យល់ពីការប្រើបញ្ជានៅខាងដើមបន្ទាត់។

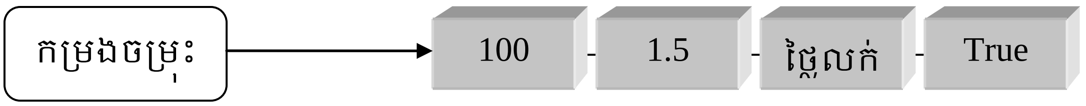
ដូចនេះ គ្រប់ឃ្លាប្រយោគទាំងឡាយណាដែលមានសញ្ញា # នេះនៅពីមុខ ត្រូវចាត់ទុកថាជាកំណត់ពន្យល់។ ហើយផ្នែកទន់បកប្រែនឹងរំលងកំណត់ពន្យល់ទាំងនោះចោលធ្វើហាក់ដូចជាគ្មាននៅក្នុងកម្មវិធី។ ម្យ៉ាងទៀតកំណត់ពន្យល់អាចត្រូវសរសេរនៅលើបន្ទាត់មួយដាច់តែឯង និងឬអាចត្រូវសរសេរនៅខាងចុងបន្ទាត់បន្ទាប់ពីបញ្ជាណាមួយ។

កម្រងថេរ

កម្រងថេរ (tuple) គឺជាកម្រងមួយដែលនៅក្នុងនោះអាចមានវត្ថុផ្សេងៗទៀតជាច្រើនតម្រៀបជួរគ្នាតាមលេខរៀងចាប់ពី 0 ឡើងទៅ បើរាប់ពីដើមកម្រងឡើងទៅ និងមានលេខរៀង -1 ចុះមកវិញ បើរាប់ពីចុងកម្រងចុះមកវិញ។ ដើម្បីបង្កើតកម្រងថេរ យើងត្រូវសរសេរកម្មវិធីដូចខាងក្រោមនេះ៖

```
កម្រងចម្រុះ = (100, 1.5, 'ថ្ងៃលក់', True)
print(កម្រងចម្រុះ)
```

កម្រងចម្រុះ = (100, 1.5, '*ថ្ងៃលក់*', True) គឺជាបញ្ជាតម្រូវឲ្យបង្កើតកម្រងថេរឈ្មោះ កម្រងចម្រុះ មួយដែលនៅក្នុងនោះមានវត្ថុជាច្រើនប្រភេទខុសៗគ្នា។



វត្ថុនៅក្នុងកម្រងថេរគឺជាធាតុនៃកម្រងថេរ។ ធាតុទាំងនោះអាចជាវត្ថុមានប្រភេទតែមួយឬជាវត្ថុមានប្រភេទខុសៗគ្នា។ ប្រការនេះធ្វើឲ្យកម្រងថេរខុសពីកម្រងអក្សរដែលជាកម្រងមានធាតុតែមួយប្រភេទដូចៗគ្នា។ ធាតុនៅក្នុងកម្រងអក្សរមិនអាចជាវត្ថុណាក្រៅពីតួអក្សរបានឡើយ។

កម្រងថេរអាចត្រូវបង្កើតឡើងដោយមិនបាច់ប្រើសញ្ញាវង្វង់ក្រចក។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
កម្រងចម្រុះ = 100, 1.5, 'ថ្លៃលក់', True
print(កម្រងចម្រុះ)
```

កម្រងចម្រុះ = 100, 1.5, 'ថ្លៃលក់', True គឺជាបញ្ជាតម្រូវឲ្យបង្កើតកម្រងថេរឈ្មោះ កម្រងចម្រុះ មួយដោយមិនចាំបាច់ប្រើសញ្ញាវង្វង់ក្រចក។

នៅក្នុងកម្រងថេរមួយអាចមានធាតុមានចំនួនមិនកំណត់ ពោលគឺចាប់ពីសូន្យរហូតដល់អនន្ត ។ ក៏ប៉ុន្តែ ដើម្បីបង្កើតកម្រងថេរមានធាតុតែមួយ យើងត្រូវធ្វើតាមរបៀបពិសេសម៉្យាងដូចខាងក្រោមនេះ៖

```
កម្រង = (1000,)
ថ្លៃលក់ = (1000)
print(កម្រង)
print(ថ្លៃលក់)
```

កម្រង = (1000,) គឺជាបញ្ជាតម្រូវឲ្យបង្កើតកម្រងថេរឈ្មោះ កម្រង ដែលមានធាតុតែមួយនៅក្នុងនោះ។

ថ្លៃលក់ = (1000) គឺជាបញ្ជាតម្រូវឲ្យបង្កើតចំនួនគត់ឈ្មោះ ថ្លៃលក់ មួយ។

ដូចនេះយើងឃើញថា បើយើងមិនប្រើសញ្ញាក្បៀសទេ គ្រប់ការសាកល្បងបង្កើតកម្រងថេរមានធាតុតែមួយ គឺជាការបង្កើតវត្ថុទោលទៅវិញទេ។ ម៉្យាងទៀតនៅក្នុងកម្មវិធីផ្សេងៗ យើង

អាចដឹងថាវត្ថុណាមួយជាកម្រងថេរមានធាតុតែមួយឬជាវត្ថុទោល គឺដោយសារកម្រងថេរមានធាតុតែមួយមានសញ្ញាក្បៀសនៅជិតនោះ។

ប្រមាណវិធីបូកបន្ត

យើងអាចយកកម្រងថេរផ្សេងៗមកធ្វើប្រមាណវិធីបូកបន្តដូចខាងក្រោមនេះ៖

```
កម្រងទីមួយ = (100, 1.5, "ថ្ងៃលក់", True)
កម្រងទីពីរ = (200, 3.14, False)
print(កម្រងទីមួយ + កម្រងទីពីរ)
```

កម្រងទីមួយ + កម្រងទីពីរ គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីបូកបន្ត ដោយចម្លងយកកម្រងថេរឈ្មោះ កម្រងទីមួយ និង កម្រងទីពីរ មកដាក់បន្តគ្នាបង្កើតបានជាកម្រងថេរថ្មីមួយទៀត។

ប្រមាណវិធីគុនបន្ត

យើងអាចយកកម្រងថេរផ្សេងៗមកធ្វើប្រមាណវិធីគុនបន្តដូចខាងក្រោមនេះ៖

```
កម្រងដើម = (100, 1.5, "ថ្ងៃលក់", True)
print(កម្រងដើម * 3)
```

*កម្រងដើម * 3* គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណគុនបន្តដោយចម្លងយកកម្រងថេរឈ្មោះ កម្រងដើម មកគុននឹង 3 បង្កើតជាកម្រងថេរចំនួនបីដូចគ្នា រួចដាក់បន្តគ្នាបង្កើតជាកម្រងថេរមួយថ្មីទៀត។

ប្រមាណវិធីតក្កវិទ្យា

យើងអាចយកកម្រងថេរផ្សេងៗមកធ្វើប្រមាណវិធីតក្កវិទ្យាដូចខាងក្រោមនេះ៖

```
កម្រងទទេ = ()
កម្រងចំរុះ = (100, 1.5, "ថ្ងៃលក់", True)
print(កម្រងទទេ and កម្រងចំរុះ)
print(កម្រងទទេ or កម្រងចំរុះ)
print(not កម្រងទទេ)
print(not កម្រងចំរុះ)
```

កម្រងទទេ and *កម្រងចំរុះ* គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីតក្កវិទ្យាដោយប្រើប្រមាណសញ្ញា and រវាងកម្រងថេរឈ្មោះ កម្រងទទេ និងកម្រងថេរឈ្មោះ កម្រងចំរុះ ។ ដោយកម្រងថេរឈ្មោះ កម្រងទទេ ជាកម្រងដែលគ្មានធាតុនៅក្នុងនោះ ដូចនេះវាសមមូលនឹងតក្កវត្ត False ដែលនាំឲ្យលទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាកម្រងថេរឈ្មោះ កម្រងទទេ នេះតែម្តង។

កម្រងទទេ or *កម្រងចំរុះ* គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីតក្កវិទ្យាដោយប្រើប្រមាណសញ្ញា or រវាងកម្រងថេរឈ្មោះ កម្រងទទេ និងកម្រងថេរឈ្មោះ កម្រងចំរុះ ។ ដោយកម្រងថេរឈ្មោះ កម្រងទទេ ជាកម្រងដែលគ្មានធាតុនៅក្នុងនោះ ដូចនេះវាសមមូលនឹងតក្កវត្ត False ដែលនាំឲ្យលទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាកម្រងថេរឈ្មោះ កម្រងចំរុះ ព្រោះវាជាប្រមាណអង្គនៅខាងស្តាំ។

not *កម្រងទទេ* គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីតក្កវិទ្យាដោយប្រើប្រមាណសញ្ញា not ជាមួយនឹងកម្រងថេរឈ្មោះ កម្រងទទេ ។ ដោយកម្រងថេរឈ្មោះ កម្រងទទេ ជាកម្រងដែលគ្មានធាតុនៅក្នុងនោះ ដូចនេះវាសមមូលនឹងតក្កវត្ត False ដែលនាំឲ្យលទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ត True ។

not *កម្រងចំរុះ* គឺជាបញ្ហាតម្រូវឲ្យធ្វើប្រមាណវិធីតក្កវិទ្យាដោយប្រើប្រមាណសញ្ញា not ជាមួយនឹងកម្រងថេរឈ្មោះ កម្រងចំរុះ ។ ដោយកម្រងថេរឈ្មោះ កម្រងចំរុះ ជាកម្រងដែលមានធាតុនៅក្នុងនោះ ដូចនេះវាសមមូលនឹងតក្កវត្ត True ដែលនាំឲ្យលទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ត False ។

ប្រមាណវិធីប្រៀបធៀប

យើងអាចយកកម្រងថេរផ្សេងៗមកធ្វើប្រមាណវិធីប្រៀបធៀបដូចខាងក្រោមនេះ៖

```
កម្រងទីមួយ = (200, 0.35, 20, 73)
កម្រងទីពីរ = (100, 1.5, 400, 3.14)
print(កម្រងទីមួយ == កម្រងទីពីរ)
print(កម្រងទីមួយ != កម្រងទីពីរ)
print(កម្រងទីមួយ > កម្រងទីពីរ)
print(កម្រងទីមួយ < កម្រងទីពីរ)
print(កម្រងទីមួយ >= កម្រងទីពីរ)
print(កម្រងទីមួយ <= កម្រងទីពីរ)
```

ប្រមាណវិធីប្រៀបធៀបរវាងកម្រងថេរផ្សេងៗ គឺជាប្រមាណវិធីប្រៀបធៀបតាមរបៀបវចនានុក្រម។ បានន័យថាធាតុនៅខាងឆ្វេងគេបង្អស់នៃកម្រងថេរនិមួយៗ ត្រូវយកមកប្រៀបធៀបគ្នា ហើយបើធាតុណាធំឬតូចជាងគេ កម្រងថេរដែលមានធាតុនោះត្រូវចាត់ទុកថាជាកម្រងថេរដែលធំឬតូចជាងគេ តែបើធាតុទាំងពីរនោះស្មើគ្នា ធាតុខាងឆ្វេងបន្ទាប់មកទៀតត្រូវយកមកប្រៀបធៀប បើធាតុទាំងអស់ដូចគ្នា កម្រងថេរទាំងនោះត្រូវចាត់ទុកថាជាកម្រងថេរដូចគ្នា។

កម្រងទីមួយ == កម្រងទីពីរ គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីប្រៀបធៀបដើម្បីពិនិត្យមើលថាតើកម្រងថេរឈ្មោះ កម្រងទីមួយ ពិតជាដូចគ្នាទៅនឹងកម្រងថេរឈ្មោះ កម្រងទីពីរ មែនដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ត False បញ្ជាក់ប្រាប់ថាកម្រងថេរឈ្មោះ កម្រងទីមួយ មិនដូចទៅនឹងកម្រងថេរឈ្មោះ កម្រងទីពីរ ទេ។

កម្រងទីមួយ != កម្រងទីពីរ គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីប្រៀបធៀបដើម្បីពិនិត្យមើលថាតើកម្រងថេរឈ្មោះ កម្រងទីមួយ ពិតជាខុសគ្នានឹងកម្រងថេរឈ្មោះ កម្រងទីពីរ មែនដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ត True បញ្ជាក់ប្រាប់ថាកម្រងថេរឈ្មោះ កម្រងទីមួយ ពិតជាមិនដូចទៅនឹងកម្រងថេរឈ្មោះ កម្រងទីពីរ មែន។

កម្រងទីមួយ > កម្រងទីពីរ គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីប្រៀបធៀបដើម្បីពិនិត្យមើលថាតើកម្រងថេរឈ្មោះ កម្រងទីមួយ ពិតជាធំជាងកម្រងថេរឈ្មោះ កម្រងទីពីរ មែនដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ត True បញ្ជាក់ប្រាប់ថាកម្រងថេរឈ្មោះ កម្រងទីមួយ ពិតជាធំជាងកម្រងថេរឈ្មោះ កម្រងទីពីរ មែន។

កម្រងទីមួយ < កម្រងទីពីរ គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីប្រៀបធៀបដើម្បីពិនិត្យមើលថាតើកម្រងថេរឈ្មោះ កម្រងទីមួយ ពិតជាតូចជាងកម្រងថេរឈ្មោះ កម្រងទីពីរ មែនដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ត False បញ្ជាក់ប្រាប់ថាកម្រងថេរឈ្មោះ កម្រងទីមួយ មិនតូចជាងកម្រងថេរឈ្មោះ កម្រងទីពីរ ទេ។

កម្រងទីមួយ >= កម្រងទីពីរ គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីប្រៀបធៀបដើម្បីពិនិត្យមើលថាតើកម្រងថេរឈ្មោះ កម្រងទីមួយ ពិតជាធំជាងឬស្មើនឹងកម្រងថេរឈ្មោះ កម្រងទីពីរ មែនដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ត True បញ្ជាក់ប្រាប់ថាកម្រងថេរឈ្មោះ កម្រងទីមួយ ពិតជាធំជាងឬស្មើនឹងកម្រងថេរឈ្មោះ កម្រងទីពីរ មែន។

កម្រងទីមួយ <= *កម្រងទីពីរ* គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីប្រៀបធៀបដើម្បីពិនិត្យមើលថាតើកម្រងថេរឈ្មោះ កម្រងទីមួយ ពិតជាតូចជាងឬស្មើនឹងកម្រងថេរឈ្មោះ កម្រងទីពីរ មែនដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ត False បញ្ជាក់ប្រាប់ថាកម្រងថេរឈ្មោះ កម្រងទីមួយ មិនតូចជាងឬស្មើនឹងកម្រងថេរឈ្មោះ កម្រងទីពីរ ទេ។

ប្រមាណវិធីរកធាតុ

យើងអាចយកកម្រងថេរផ្សេងៗមកធ្វើប្រមាណវិធីរកធាតុដូចខាងក្រោមនេះ៖

```
កម្រងដើម = (200, 0.35, 20, 73)
ថ្លៃលក់ = 73
print(ថ្លៃលក់ in កម្រងដើម)
print(ថ្លៃលក់ not in កម្រងដើម)
```

ថ្លៃលក់ in *កម្រងដើម* គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីរកធាតុដើម្បីពិនិត្យមើលថាតើនៅក្នុងកម្រងថេរឈ្មោះ កម្រងដើម មានធាតុណាមួយដូចទៅនឹងវត្ថុឈ្មោះ ថ្លៃលក់ ដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណនេះគឺជាតក្កវត្ត True បញ្ជាក់ប្រាប់ថានៅក្នុងកម្រងនោះពិតជាមានធាតុដូចទៅនឹងវត្ថុឈ្មោះ ថ្លៃលក់ មែន។

ថ្លៃលក់ not in *កម្រងដើម* គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីរកធាតុដើម្បីពិនិត្យមើលថាតើនៅក្នុងកម្រងថេរឈ្មោះ កម្រងដើម ពិតជាគ្មានធាតុណាមួយដូចទៅនឹងវត្ថុឈ្មោះ ថ្លៃលក់ ដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណនេះគឺជាតក្កវត្ត False បញ្ជាក់ប្រាប់ថា ប្រការដែលថានៅក្នុងកម្រងថេរឈ្មោះ កម្រងដើម ពិតជាមានធាតុដូចទៅនឹងវត្ថុឈ្មោះ ថ្លៃលក់ គឺខុស។

ប្រមាណវិធីអត្តសញ្ញាណ

យើងអាចយកកម្រងថេរផ្សេងៗមកធ្វើប្រមាណវិធីអត្តសញ្ញាណដូចខាងក្រោមនេះ៖

```
កម្រង៍ទីមួយ = (200, 0.35, 20, 73)
កម្រង៍ទីពីរ = (100, 1.5, 400, 3.14)
print(កម្រង៍ទីមួយ is កម្រង៍ទីពីរ)
print(កម្រង៍ទីមួយ is not កម្រង៍ទីពីរ)
```

កម្រង៍ទីមួយ is *កម្រង៍ទីពីរ* គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីអត្តសញ្ញាណដើម្បីពិនិត្យមើលថាតើកម្រងថេរឈ្មោះ កម្រងទីមួយ និងកម្រងថេរឈ្មោះ កម្រងទីពីរ ពិតជាកម្រងថេរតែមួយមែនដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ត False បញ្ជាក់ប្រាប់ថា កម្រងថេរឈ្មោះ កម្រងទីមួយ និងកម្រងថេរឈ្មោះ កម្រងទីពីរ មិនមែនជាវត្ថុតែមួយនោះទេ។

កម្រង៍ទីមួយ is not *កម្រង៍ទីពីរ* គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីអត្តសញ្ញាណដើម្បីពិនិត្យមើលថាតើកម្រងថេរឈ្មោះ កម្រងទីមួយ និងកម្រងថេរឈ្មោះ កម្រងទីពីរ ពិតជាមិនមែនជាកម្រងថេរតែមួយមែនដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ត True បញ្ជាក់ប្រាប់ថា កម្រងថេរឈ្មោះ កម្រងទីមួយ និងកម្រងថេរឈ្មោះ កម្រងទីពីរ ពិតជាមិនមែនជាវត្ថុតែមួយមែន។

ប្រមាណវិធីលេខរៀង

យើងអាចយកកម្រងថេរផ្សេងៗមកធ្វើប្រមាណវិធីលេខរៀងដូចខាងក្រោមនេះ៖

```
កម្រង៍ដើម = (200, 0.35, 20, 73)
print(កម្រង៍ដើម[2])
```

កម្រងដើម[2] គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីលេខរៀងដើម្បីចម្លងយកធាតុមានលេខរៀង 2 នៅក្នុងកម្រងថេរឈ្មោះ កម្រងដើម មកប្រើ។

ប្រមាណវិធីកាត់ចម្លង

យើងអាចយកកម្រងថេរផ្សេងៗមកធ្វើប្រមាណវិធីកាត់ចម្លងដូចខាងក្រោមនេះ៖

```
កម្រងដើម = (100, 10.5, "ប្រាក់ចំណេញ", True, 200, "ថ្លៃលក់", False)
print(កម្រងដើម[1:7])
print(កម្រងដើម[1:])
print(កម្រងដើម[:7])
print(កម្រងដើម[:])
print(កម្រងដើម[1:7:2])
```

កម្រងដើម[1:7] គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីកាត់ចម្លងដោយចម្លងយកធាតុមួយចំនួនដែលមានលេខរៀងចាប់ពី 1 រហូតដល់ 6 នៅក្នុងកម្រងថេរឈ្មោះ កម្រងដើម មកបង្កើតជាកម្រងថេរមួយថ្មីទៀត។

កម្រងដើម[1:] គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីកាត់ចម្លងដោយចម្លងយកធាតុមួយចំនួនដែលមានលេខរៀងចាប់ពី 1 រហូតដល់ធាតុចុងគេបង្អស់នៅក្នុងកម្រងថេរឈ្មោះ កម្រងដើម មកបង្កើតជាកម្រងថេរមួយថ្មីទៀត។

កម្រងដើម[:7] គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីកាត់ចម្លងដោយចម្លងយកធាតុមួយចំនួនដែលមានលេខរៀងចាប់ពី 0 រហូតដល់ 6 នៅក្នុងកម្រងថេរឈ្មោះ កម្រងដើម មកបង្កើតជាកម្រងថេរមួយថ្មីទៀត។

កម្រងដើម[:] គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីកាត់ចម្លងដោយចម្លងយកធាតុទាំងអស់ដែលមាននៅក្នុងកម្រងថេរឈ្មោះ កម្រងដើម មកបង្កើតជាកម្រងថេរមួយថ្មីទៀត។

កម្រងដើម[1:7:2] គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីកាត់ចម្លងដើម្បីចម្លងយកធាតុមួយចំនួននៅក្នុងកម្រងថេរឈ្មោះ កម្រងដើម មកបង្កើតជាកម្រងថេរមួយថ្មីទៀត។ ធាតុដែលត្រូវចម្លងយក គឺជាធាតុដែលមានលេខរៀងចាប់ពី 1 រហូតដល់ 6 ដោយរំលងធាតុចំនួនមួយចោលរហូត។

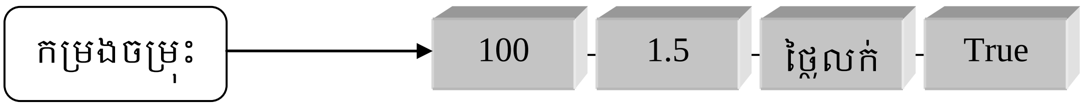
កម្រងអថេរ

កម្រងអថេរ (list) ស្រដៀងនឹងកម្រងថេរដែរ ព្រោះវាក៏ជាកម្រងមួយដែលនៅក្នុងនោះអាចមានវត្ថុជាច្រើនតម្រៀបជួរគ្នាតាមលេខរៀងចាប់ពីសូន្យឡើងទៅ បើយើងរាប់ពីដើមកម្រងឡើងទៅ និងមានលេខរៀងចាប់ពី -1 ចុះមកបើយើងរាប់ចាប់ពីចុងកម្រងចុះមកវិញ។

ដើម្បីបង្កើតកម្រងអថេរ យើងត្រូវធ្វើដូចខាងក្រោមនេះ៖

```
កម្រងដើម = [100, 10.5,  "ថ្ងៃលក់", True]
print(កម្រងដើម)
```

កម្រងដើម = [100, 10.5, "ថ្ងៃលក់", True] គឺជាបញ្ជាតម្រូវឲ្យបង្កើតកម្រងអថេរមួយមានឈ្មោះថា កម្រងដើម ។



ប្រមាណវិធីបូកបន្ត

យើងអាចយកកម្រងអថេរផ្សេងៗមកធ្វើប្រមាណវិធីបូកបន្តដូចខាងក្រោមនេះ៖

```
កម្រងទីមួយ = [100, 10.5,  "ថ្ងៃលក់", True]
កម្រងទីពីរ = [200, False, 3.14]
print(កម្រងទីមួយ + កម្រងទីពីរ)
```

កម្រងទីមួយ + កម្រងទីពីរ គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីបូកបន្តដោយចម្លងយកកម្រងអថេរឈ្មោះ កម្រងទីមួយ និងកម្រងអថេរឈ្មោះ កម្រងទីពីរ មកដាក់បន្តគ្នាបង្កើតជាកម្រងអថេរថ្មីមួយទៀត។

ប្រមាណវិធីគុនបន្ត

យើងអាចយកកម្រងអថេរផ្សេងៗមកធ្វើប្រមាណវិធីគុនបន្តដូចខាងក្រោមនេះ៖

```
កម្រងដើម = [100, 10.5,  "ថ្ងៃលក់", True]
print(កម្រងដើម * 3)
```

*កម្រងដើម * 3* គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីគុនបន្ត ដោយចម្លងយកកម្រងអថេរឈ្មោះ កម្រងដើម មកគុននឹងលេខ 3 បង្កើតបានជាកម្រងអថេរដូចគ្នាចំនួនបី រួចដាក់បន្តគ្នាបង្កើតជាកម្រងអថេរមួយថ្មីទៀត។

ប្រមាណវិធីតក្កវិទ្យា

យើងអាចយកកម្រងអថេរផ្សេងៗមកធ្វើប្រមាណវិធីតក្កវិទ្យាដូចខាងក្រោមនេះ៖

```
កម្រងដើម = [100, 10.5,  "ថ្ងៃលក់", True]
```

```
កម្រង់ទទេ = [ ]
print(កម្រង់ដើម and កម្រង់ទទេ)
print(កម្រង់ដើម or កម្រង់ទទេ)
print(not កម្រង់ដើម)
print(not កម្រង់ទទេ)
```

កម្រង់ដើម and *កម្រង់ទទេ* គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីតក្កវិទ្យាដោយប្រើប្រមាណសញ្ញា and ។ ដោយកម្រងអថេរឈ្មោះ កម្រងដើម ជាកម្រងអថេរមានធាតុនៅក្នុងនោះ ដូចនេះវាសមមូលនឹងតក្កវត្ត True ដែលនាំឲ្យលទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាកម្រងអថេរឈ្មោះ កម្រងទទេ ព្រោះវាជាប្រមាណអង្គនៅខាងស្តាំ។

កម្រង់ដើម or *កម្រង់ទទេ* គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីតក្កវិទ្យាដោយប្រើប្រមាណសញ្ញា or ។ ដោយកម្រងអថេរឈ្មោះ កម្រងដើម ជាកម្រងអថេរមានធាតុនៅក្នុងនោះ ដូចនេះវាសមមូលនឹងតក្កវត្ត True ដែលនាំឲ្យលទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាកម្រងអថេរនោះតែម្តង។

not *កម្រង់ដើម* គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីតក្កវិទ្យាដោយប្រើប្រមាណសញ្ញា not ។ ដោយកម្រងអថេរឈ្មោះ កម្រងដើម ជាកម្រងអថេរមានធាតុនៅក្នុងនោះ ដូចនេះវាសមមូលនឹងតក្កវត្ត True ដែលនាំឲ្យលទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ត False ។

not *កម្រង់ទទេ* គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីតក្កវិទ្យាដោយប្រើប្រមាណសញ្ញា not ។ ដោយកម្រងអថេរឈ្មោះ កម្រងទទេ ជាកម្រងអថេរគ្មានធាតុនៅក្នុងនោះ ដូចនេះវាសមមូលនឹងតក្កវត្ត False ដែលនាំឲ្យលទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ត True ។

ប្រមាណវិធីប្រៀបធៀប

យើងអាចយកកម្រងអថេរផ្សេងៗមកធ្វើប្រមាណវិធីប្រៀបធៀបដូចខាងក្រោមនេះ៖

```
កម្រង៍ទីមួយ = [100, 10.5,  300, 0.33]
កម្រង៍ទីពីរ = [200, 0.5, 50 ]
print(កម្រង៍ទីមួយ == កម្រង៍ទីពីរ)
print(កម្រង៍ទីមួយ != កម្រង៍ទីពីរ)
print(កម្រង៍ទីមួយ > កម្រង៍ទីពីរ)
print(កម្រង៍ទីមួយ < កម្រង៍ទីពីរ)
print(កម្រង៍ទីមួយ >= កម្រង៍ទីពីរ)
print(កម្រង៍ទីមួយ <= កម្រង៍ទីពីរ)
```

ប្រមាណវិធីប្រៀបធៀបរវាងកម្រងអថេរផ្សេងៗ គឺជាប្រមាណវិធីប្រៀបធៀបតាមរបៀបវចនានុក្រម។ ពោលគឺធាតុនៅខាងឆ្វេងគេបង្អស់នៃកម្រងអថេរដែលជាប្រមាណអង្គនិមួយៗត្រូវយកមកប្រៀបធៀបគ្នា ហើយបើធាតុណាធំឬតូចជាងគេ កម្រងអថេរដែលមានធាតុនោះត្រូវចាត់ទុកថាជាកម្រងអថេរធំឬតូចជាងគេ។ តែបើធាតុទាំងពីរនោះស្មើគ្នា ធាតុខាងឆ្វេងបន្ទាប់មកទៀតនឹងត្រូវយកមកប្រៀបធៀបជាបន្តទៅទៀត ហើយបើធាតុទាំងអស់ដូចគ្នា កម្រងអថេរទាំងពីរនោះនឹងត្រូវចាត់ទុកថាជាកម្រងអថេរដូចគ្នា។

កម្រង៍ទីមួយ == កម្រង៍ទីពីរ គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីប្រៀបធៀបដើម្បីពិនិត្យមើលថា តើកម្រងអថេរឈ្មោះ កម្រងទីមួយ និងកម្រងអថេរឈ្មោះ កម្រងទីពីរ ដូចគ្នាដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនោះគឺជាតក្កវត្ថុ False បញ្ជាក់ប្រាប់ថាកម្រងអថេរឈ្មោះ កម្រងទីមួយ ខុសពីកម្រងអថេរឈ្មោះ កម្រងទីពីរ ។

កម្រង៍ទីមួយ != កម្រង៍ទីពីរ គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីប្រៀបធៀបដើម្បីពិនិត្យមើលថា តើកម្រងអថេរឈ្មោះ កម្រងទីមួយ និងកម្រងអថេរឈ្មោះ កម្រងទីពីរ ដូចគ្នាដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនោះគឺជាតក្កវត្ត False បញ្ជាក់ប្រាប់ថាកម្រងអថេរឈ្មោះ កម្រងទីមួយ ខុសពីកម្រងអថេរឈ្មោះ កម្រងទីពីរ ។

កម្រង៍ទីមួយ > កម្រង៍ទីពីរ គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីប្រៀបធៀបដើម្បីពិនិត្រមើលថា តើកម្រងអថេរឈ្មោះ កម្រងទីមួយ ធំជាងកម្រងអថេរឈ្មោះ កម្រងទីពីរ ដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនោះគឺជាតក្កវត្ត False បញ្ជាក់ប្រាប់ថាកម្រងអថេរឈ្មោះ កម្រងទីមួយ មិនធំជាងកម្រងអថេរឈ្មោះ កម្រងទីពីរ ទេ។

កម្រង៍ទីមួយ < កម្រង៍ទីពីរ គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីប្រៀបធៀបដើម្បីពិនិត្រមើលថា តើកម្រងអថេរឈ្មោះ កម្រងទីមួយ តូចជាងកម្រងអថេរឈ្មោះ កម្រងទីពីរ ដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនោះគឺជាតក្កវត្ត True បញ្ជាក់ប្រាប់ថាកម្រងអថេរឈ្មោះ កម្រងទីមួយ ពិតជាតូចជាងកម្រងអថេរឈ្មោះ កម្រងទីពីរ មែន។

កម្រង៍ទីមួយ >= កម្រង៍ទីពីរ គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីប្រៀបធៀបដើម្បីពិនិត្រមើលថា តើកម្រងអថេរឈ្មោះ កម្រងទីមួយ ធំជាងឬស្មើនឹងកម្រងអថេរឈ្មោះ កម្រងទីពីរ មែនដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនោះគឺជាតក្កវត្ត False បញ្ជាក់ប្រាប់ថាកម្រងអថេរឈ្មោះ កម្រងទីមួយ មិនធំជាងឬស្មើនឹងកម្រងអថេរឈ្មោះ កម្រងទីពីរ ទេ។

កម្រង៍ទីមួយ <= កម្រង៍ទីពីរ គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីប្រៀបធៀបដើម្បីពិនិត្រមើលថា តើកម្រងអថេរឈ្មោះ កម្រងទីមួយ តូចជាងឬស្មើនឹងកម្រងអថេរឈ្មោះ កម្រងទីពីរ ដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនោះគឺជាតក្កវត្ត True បញ្ជាក់ប្រាប់ថាកម្រងអថេរឈ្មោះ កម្រងទីមួយ ពិតជាតូចជាងឬស្មើនឹងកម្រងអថេរឈ្មោះ កម្រងទីពីរ មែន។

ប្រមាណវិធីរកធាតុ

យើងអាចយកកម្រងអថេរផ្សេងៗមកធ្វើប្រមាណវិធីរកធាតុដូចខាងក្រោមនេះ៖

```
កម្រង៍ដើម = [100, 1.5, "ប្រាក់ចំណេញ", True]
ពាក្យ = "ប្រាក់ចំណេញ"
print(ពាក្យ in កម្រង៍ដើម)
print(ពាក្យ not in កម្រង៍ដើម)
```

ពាក្យ in កម្រង៍ដើម គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីរកធាតុដើម្បីពិនិត្យមើលថាតើនៅក្នុងកម្រងអថេរឈ្មោះ កម្រងដើម មានធាតុណាមួយដូចទៅនឹងវត្ថុឈ្មោះ ពាក្យ ដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះ គឺជាតក្កវត្ត True បញ្ជាក់ប្រាប់ថានៅក្នុងកម្រងអថេរឈ្មោះ កម្រងដើម ពិតជាមានធាតុណាមយយដូចទៅនឹងវត្ថុមានឈ្មោះថា ពាក្យ នោះមែន។

ពាក្យ not in កម្រង៍ដើម គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីរកធាតុដើម្បីពិនិត្យមើលថាតើនៅក្នុងកម្រងអថេរឈ្មោះ កម្រងដើម គ្មានធាតុណាមួយដូចទៅនឹងវត្ថុឈ្មោះ ពាក្យ មែនដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះ គឺជាតក្កវត្ត False បញ្ជាក់ប្រាប់ថា ប្រការដែលថានៅក្នុងកម្រងអថេរឈ្មោះ កម្រងដើម គ្មានធាតុដូចទៅនឹងវត្ថុមានឈ្មោះថា ពាក្យ គឺខុស។

ប្រមាណវិធីអត្តសញ្ញាណ

យើងអាចយកកម្រងអថេរផ្សេងៗមកធ្វើប្រមាណវិធីអត្តសញ្ញាណដូចខាងក្រោមនេះ៖

```
កម្រង៍ទីមួយ = [100, 1.5, "ប្រាក់ចំណេញ", True]
កម្រង៍ទីពីរ = កម្រង៍ទីមួយ
កម្រង៍ទីបី = [100, 1.5, "ប្រាក់ចំណេញ", True]
print(កម្រង៍ទីមួយ is កម្រង៍ទីពីរ)
```

```
print(កម្រង៍ទីមួយ is កម្រង៍ទីបី)
print(កម្រង៍ទីមួយ is not កម្រង៍ទីបី)
```

កម្រង៍ទីមួយ is *កម្រង៍ទីពីរ* គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីអត្តសញ្ញាណដើម្បីពិនិត្យមើលថាតើកម្រងអថេរឈ្មោះ កម្រងទីមួយ និងកម្រងអថេរឈ្មោះ កម្រងទីពីរ ជាវត្ថុតែមួយមែនឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ថុ True បញ្ជាក់ប្រាប់ថាកម្រងអថេរឈ្មោះ កម្រងទីមួយ និងកម្រងអថេរឈ្មោះ កម្រងទីពីរ ពិតជាវត្ថុតែមួយមែន។

កម្រង៍ទីមួយ is *កម្រង៍ទីបី* គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីអត្តសញ្ញាណដើម្បីពិនិត្យមើលថាតើកម្រងអថេរឈ្មោះ កម្រងទីមួយ និងកម្រងអថេរឈ្មោះ កម្រងទីបី ជាវត្ថុតែមួយមែនឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ថុ False បញ្ជាក់ប្រាប់ថាកម្រងអថេរឈ្មោះ កម្រងទីមួយ និងកម្រងអថេរឈ្មោះ កម្រងទីបី មិនមែនជាវត្ថុតែមួយទេ។

កម្រង៍ទីមួយ is not *កម្រង៍ទីបី* គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីអត្តសញ្ញាណដើម្បីពិនិត្យមើលថាតើកម្រងអថេរឈ្មោះ កម្រងទីមួយ និងកម្រងអថេរឈ្មោះ កម្រងទីបី ពិតជាមិនមែនជាវត្ថុតែមួយមែនឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ថុ True បញ្ជាក់ប្រាប់ថាកម្រងអថេរឈ្មោះ កម្រងទីមួយ និងកម្រងអថេរឈ្មោះ កម្រងទីបី ពិតជាមិនមែនជាវត្ថុតែមួយមែន។

យើងសង្កេតឃើញថាកម្រងអថេរឈ្មោះ កម្រងទីមួយ និងកម្រងអថេរឈ្មោះ កម្រងទីបី គឺជាកម្រងអថេរពីរដូចគ្នា តែវាមិនមែនជាកម្រងអថេរតែមួយទេ។ បានន័យថាកម្រងអថេរទាំងនោះ ជាកម្រងអថេរពីរដែលមានធាតុដូចគ្នា តែវាជាវត្ថុពីរស្ថិតនៅក្នុងកន្លែងផ្សេងគ្នានៅក្នុងសតិរបស់កំព្យូទ័រ។ ហើយជាទូទៅ វត្ថុពីរដូចគ្នាមិនមែនជាវត្ថុតែមួយទេ។

ប្រមាណវិធីលេខរៀង

យើងអាចយកកម្រងអថេរផ្សេងៗមកធ្វើប្រមាណវិធីលេខរៀងដូចខាងក្រោមនេះ៖

```
កម្រងដើម = [100, 1.5, "នាម", True]
print(កម្រងដើម[-2])
កម្រងដើម[-2] = "នាមត្រកូល"
print(កម្រងដើម)
```

កម្រងដើម[-2] គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីលេខរៀងដើម្បីចម្លងយកធាតុមានលេខរៀង -2 នៅក្នុងកម្រងអថេរឈ្មោះ កម្រងដើម មកប្រើការ។

កម្រងដើម[-2] = "*នាមត្រកូល*" គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីលេខរៀងដើម្បីជំនួសធាតុមានលេខរៀង -2 នៅក្នុងកម្រងអថេរឈ្មោះ កម្រងដើម ។

ដូចនេះយើងឃើញថា ធាតុនៅក្នុងកម្រងអថេរអាចត្រូវជំនួសដោយវត្ថុផ្សេងៗទៀតបាន។ គឺលក្ខណៈនេះហើយដែលធ្វើឲ្យកម្រងអថេរខុសពីកម្រងថេរនិងកម្រងអក្សរ។ ធាតុនៅក្នុងកម្រងថេរនិងកម្រងអក្សរមិនអាចត្រូវដោះដូរឬជំនួសដោយវត្ថុផ្សេងទៀតបានឡើយ។ ការប៉ុនប៉ងដោះដូរធាតុនៅក្នុងកម្រងទាំងពីរនេះនឹងបណ្តាលឲ្យកំហុសកើតមានឡើង។

នៅក្នុងភាសា Python គ្រប់វត្ថុទាំងឡាយណាដែលមានធាតុអាចត្រូវដោះដូរបាន គឺជាវត្ថុ *អាចដោះដូរបាន* (mutable) និងគ្រប់វត្ថុទាំងឡាយណាដែលមានធាតុមិនអាចត្រូវដោះដូរបាន គឺជាវត្ថុ *មិនអាចដោះដូរបាន* (immutable) ។ ដូចនេះ លេខ តក្កវត្ថុ មោឃៈវត្ថុ កម្រងអក្សរ និងកម្រងថេរ គឺជាវត្ថុមិនអាចដោះដូរបាន ចំណែកឯកម្រងអថេរវិញគឺជាវត្ថុអាចដោះដូរបាន។

ប្រមាណវិធីកាត់ចម្លង

យើងអាចយកកម្រងអថេរផ្សេងៗមកធ្វើប្រមាណវិធីកាត់ចម្លងដូចខាងក្រោមនេះ៖

```
កម្រងដើម = [100, 1.5, "នាម", True, 200, 3.5, "ប្រាក់ចំណេញ", "គោត្តនាម"]
print(កម្រងដើម[1:7])
print(កម្រងដើម[1:])
print(កម្រងដើម[:7])
print(កម្រងដើម[:])
print(កម្រងដើម[1:7:2])
```

កម្រងដើម[1:7] គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីកាត់ចម្លងដោយចម្លងយកធាតុមួយចំនួនពីក្នុងកម្រងអថេរឈ្មោះ កម្រងដើម មកបង្កើតជាកម្រងអថេរថ្មីមួយទៀត។ ធាតុដែលត្រូវចម្លងយកមកនោះ គឺជាធាតុទាំងឡាយណាដែលមានលេខរៀងចាប់ពី 1 រហូតដល់ 6 ។

កម្រងដើម[1:] គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីកាត់ចម្លងដោយចម្លងយកធាតុមួយចំនួនពីក្នុងកម្រងអថេរឈ្មោះ កម្រងដើម មកបង្កើតជាកម្រងអថេរថ្មីមួយទៀត។ ធាតុដែលត្រូវចម្លងយកមកនោះ គឺជាធាតុទាំងឡាយណាដែលមានលេខរៀងចាប់ពី 1 រហូតដល់ធាតុនៅខាងចុងគេបំផុត។

កម្រងដើម[:7] គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីកាត់ចម្លងដោយចម្លងយកធាតុមួយចំនួនពីក្នុងកម្រងអថេរឈ្មោះ កម្រងដើម មកបង្កើតជាកម្រងអថេរថ្មីមួយទៀត។ ធាតុដែលត្រូវចម្លងយកមកនោះ គឺជាធាតុទាំងឡាយណាដែលមានលេខរៀងចាប់ពី 0 រហូតដល់ 6 ។

កម្រងដើម[:] គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីកាត់ចម្លងដោយចម្លងយកធាតុទាំងអស់ពីក្នុងកម្រងអថេរឈ្មោះ កម្រងដើម មកបង្កើតជាកម្រងអថេរថ្មីមួយទៀត។

កម្រងដើម[1:7:2] គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីកាត់ចម្លងដោយចម្លងយកធាតុមួយចំនួនពីក្នុងកម្រងអថេរឈ្មោះ កម្រងដើម មកបង្កើតជាកម្រងអថេរថ្មីមួយទៀត។ ធាតុដែលត្រូវចម្លងយកមកនោះ គឺជាធាតុទាំងឡាយណាដែលមានលេខរៀងចាប់ពី 1 រហូតដល់ 6 ដោយរំលងធាតុចំនួនមួយចោលរហូត។

ដោយហេតុថាកម្រងអថេរជាវត្ថុអាចដោះដូរបាន ដូចនេះយើងអាចយកវាមកធ្វើប្រមាណវិធីកាត់ចម្លងដើម្បីដោះដូរធាតុរបស់វាមួយចំនួនដូចខាងក្រោមនេះ៖

```
កម្រងដើម = [100, 1.5, "នាម", True, 200, 3.5, "ប្រាក់ចំណេញ", "គោត្តនាម"]
កម្រងដើម[1:7] = [1, 2, 3, 4, 5, 6]
print(កម្រងដើម)
```

កម្រងដើម[1:7] = [1, 2, 3, 4, 5, 6] គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីកាត់ចម្លងដើម្បីជំនួសធាតុមានលេខរៀងចាប់ពី 1 រហូតដល់ 6 ដោយធាតុនៅក្នុងកម្រងអថេរ [1, 2, 3, 4, 5, 6] ។

វត្ថុដែលត្រូវយកទៅជំនួសធាតុនៅក្នុងកម្រងអថេរ អាចជាវត្ថុដែលជាធាតុនៃកម្រងអថេរណាមួយក៏បានដែរ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
កម្រងដើម = [100, 1.5, "នាម", True, 200, 3.5, "ប្រាក់ចំណេញ", "គោត្តនាម"]
កម្រងដើម[1:7] = (1, 2, 3, 4, 5, 6)
print(កម្រងដើម)
```

កម្រងដើម[1:7] = (1, 2, 3, 4, 5, 6) គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណកាត់ចម្លងដោយចម្លងយកធាតុទាំងអស់នៅក្នុងកម្រងថេរមួយមកជំនួសធាតុមួយចំនួននៅក្នុងកម្រងអថេរឈ្មោះ កម្រងដើម ។

វចនានុក្រម

វចនានុក្រម (dictionary) គឺជាសមាសវត្ថុមួយដែលនៅក្នុងនោះអាចមានវត្ថុជាប់គ្នាមួយគូជាច្រើន។ ដើម្បីបង្កើតវចនានុក្រម យើងត្រូវធ្វើដូចខាងក្រោមនេះ៖

```
ប្រវត្តិរូប = {"នាម":"កុសល", "គោត្តនាម":"កែវ", "អាយុ":35, "ភេទ":"ប្រុស"}
print(ប្រវត្តិរូប)
```

ប្រវត្តិរូប = {"នាម":"កុសល", "គោត្តនាម":"កែវ", "អាយុ":35, "ភេទ":"ប្រុស"} គឺជាបញ្ជាតម្រូវឲ្យបង្កើតវចនានុក្រមមួយមានឈ្មោះថា ប្រវត្តិរូប ។

ធាតុដែលជាប់គ្នាពីរៗនៅក្នុងវចនានុក្រមហៅថា *ធាតុគូ* (item) ។ ធាតុទាំងនោះស្ថិតនៅជាប់គ្នាដោយសារសញ្ញា : ។ នៅក្នុងធាតុគូ វត្ថុនៅខាងឆ្វេងត្រូវហៅថា *កូនសោរ* (key) និងវត្ថុនៅខាងស្តាំគឺជា *តម្លៃ* (value) ។ ធាតុគូនៅក្នុងវចនានុក្រមគ្មានលេខរៀងច្បាស់លាស់ទេ គឺវាអាចត្រូវផ្លាស់ប្តូរទីតាំងបានគ្រប់ពេលវេលា។ ម៉្យាងទៀត កូនសោរនៅក្នុងធាតុគូត្រូវតែជាវត្ថុមិនអាចដោះដូរបាន តែចំពោះតម្លៃវិញអាចជាវត្ថុប្រភេទណាក៏បានដែរ។ ជាទូទៅ កូនសោរមាននាទីជាផ្លាកសម្រាប់សម្គាល់តម្លៃនិមួយៗនៅក្នុងវចនានុក្រម ហើយគឺដោយសារកូនសោរនេះហើយដែលតម្លៃអាចត្រូវយកទៅប្រើការផ្សេងៗទៀតបាន។

ប្រមាណវិធីលេខរៀង

យើងអាចយកវចនានុក្រមផ្សេងៗមកធ្វើប្រមាណវិធីលេខរៀងដូចខាងក្រោមនេះ៖

```
ប្រវត្តិរូប = {"នាម":"កុសល", "គោត្តនាម":"កែវ", "អាយុ":35, "ភេទ":"ប្រុស"}
print(ប្រវត្តិរូប["គោត្តនាម"])
print(ប្រវត្តិរូប["អាយុ"])
```

ប្រវត្តិរូប["គោត្តនាម"] គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីលេខរៀងដើម្បីចម្លងយកតម្លៃជាប់នឹងកូនសោរ "គោត្តនាម" នៅក្នុងវចនានុក្រមឈ្មោះ ប្រវត្តិរូប មកប្រើការ។

ប្រវត្តិរូប["អាយុ"] គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីលេខរៀងដើម្បីចម្លងយកតម្លៃជាប់នឹងកូនសោរ "អាយុ" នៅក្នុងវចនានុក្រមឈ្មោះ ប្រវត្តិរូប មកប្រើការ។

វចនានុក្រមគឺជាវត្ថុអាចដោះដូរបាន បានន័យថាតម្លៃនៅក្នុងវចនានុក្រមអាចត្រូវជំនួសដោយវត្ថុផ្សេងៗទៀតបាន។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
ប្រវត្តិរូប = {"នាម":"កុសល", "គោត្តនាម":"កែវ", "អាយុ":35, "ភេទ":"ប្រុស"}
ប្រវត្តិរូប["គោត្តនាម"] = "លឹម"
print(ប្រវត្តិរូប)
```

ប្រវត្តិរូប["គោត្តនាម"] = "លឹម" គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីលេខរៀងដើម្បីជំនួសតម្លៃជាប់នឹងកូនសោរ "គោត្តនាម" នៅក្នុងវចនានុក្រមឈ្មោះ ប្រវត្តិរូប ដោយកម្រងអក្សរឈ្មោះ "លឹម" ។ ដូចនេះបន្ទាប់ពីការដោះដូរនេះរួចមក តម្លៃជាប់នឹងកូនសោរ "គោត្តនាម" នៅក្នុងវចនានុក្រមឈ្មោះ ប្រវត្តិរូប គឺជាកម្រងអក្សរ "លឹម" ។

លក្ខណៈពិសេសមួយទៀតរបស់វចនានុក្រម គឺយើងអាចធ្វើប្រមាណវិធីលេខរៀងដើម្បីបន្ថែមធាតុគូថ្មីចូលទៅក្នុងវចនានុក្រម។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
ប្រវត្តិរូប = {"នាម":"កុសល", "គោត្តនាម":"កែវ", "អាយុ":35, "ភេទ":"ប្រុស"}
ប្រវត្តិរូប["ទីលំនៅ"] = "រាជាណាចក្រកម្ពុជា"
print(ប្រវត្តិរូប)
```

ប្រវត្តិរូប["ទីលំនៅ"] = "រាជាណាចក្រកម្ពុជា" គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីលេខរៀង ដើម្បីបន្ថែមធាតុគូថ្មីចូលទៅក្នុងវចនានុក្រមឈ្មោះ ប្រវត្តិរូប ។ ធាតុគូថ្មីនោះមានកម្រងអក្សរ "ទីលំនៅ" ជាកូនសោរ និង "រាជាណាចក្រកម្ពុជា" ជាតម្លៃ។

ប្រមាណវិធីតក្កវិទ្យា

យើងអាចយកវចនានុក្រមផ្សេងៗមកធ្វើប្រមាណវិធីតក្កវិទ្យាដូចខាងក្រោមនេះ៖

```
ប្រវត្តិរូប = {"នាម":"កុសល", "គោត្តនាម":"កែវ", "អាយុ":35, "ភេទ":"ប្រុស"}
វចនានុក្រមទទេ = { }
print(ប្រវត្តិរូប and វចនានុក្រមទទេ)
print(ប្រវត្តិរូប or វចនានុក្រមទទេ)
print(not ប្រវត្តិរូប)
print(not វចនានុក្រមទទេ)
```

ប្រវត្តិរូប and វចនានុក្រមទទេ គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីតក្កវិទ្យាដោយប្រើប្រមាណសញ្ញា and ។ ដោយវចនានុក្រមឈ្មោះ ប្រវត្តិរូប ជាវចនានុក្រមមានធាតុនៅក្នុងនោះ ដូចនេះ វាសមមូលនឹងតក្កវត្ត True ដែលនាំឲ្យលទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាវចនានុក្រមឈ្មោះ វចនានុក្រមទទេ ព្រោះវាជាប្រមាណអង្គនៅខាងស្តាំ។

ប្រវត្តិរូប or វចនានុក្រមទទេ គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីតក្កវិទ្យាដោយប្រើប្រមាណសញ្ញា or ។ ដោយវចនានុក្រមឈ្មោះ ប្រវត្តិរូប ជាវចនានុក្រមមានធាតុនៅក្នុងនោះ ដូចនេះវាសមមូលនឹងតក្កវត្ត True ដែលនាំឲ្យលទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាវចនានុក្រមនេះតែម្តង។

not *ប្រវត្តិរូប* គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីតក្កវិទ្យាដោយប្រើប្រមាណសញ្ញា not ជាមួយនឹងវចនានុក្រមឈ្មោះ ប្រវត្តិរូប ។ ដោយវចនានុក្រមនេះជាវចនានុក្រមមានធាតុនៅក្នុងនោះ ដូចនេះវាសមមូលនឹងតក្កវត្ត True ដែលនាំឲ្យលទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ត False ។

not *វចនានុក្រមទទេ* គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីតក្កវិទ្យាដោយប្រើប្រមាណសញ្ញា not ជាមួយនឹងវចនានុក្រមឈ្មោះ វចនានុក្រមទទេ ។ ដោយវចនានុក្រមនេះជាវចនានុក្រមគ្មានធាតុនៅក្នុងនោះ ដូចនេះវាសមមូលនឹងតក្កវត្ត False ដែលនាំឲ្យលទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ត True ។

ប្រមាណវិធីប្រៀបធៀប

យើងអាចយកវចនានុក្រមផ្សេងៗមកធ្វើប្រមាណវិធីប្រៀបធៀបដូចខាងក្រោមនេះ៖

```
ប្រវត្តិរូបដើម = {"នាម":"កុសល", "គោត្តនាម":"កែវ", "អាយុ":35, "ភេទ":"ប្រុស"}
ប្រវត្តិរូបចម្លង = {"នាម":"កុសល", "គោត្តនាម":"កែវ", "អាយុ":35, "ភេទ":"ប្រុស"}
print(ប្រវត្តិរូបដើម == ប្រវត្តិរូបចម្លង)
print(ប្រវត្តិរូបដើម != ប្រវត្តិរូបចម្លង)
```

ប្រវត្តិរូបដើម == ប្រវត្តិរូបចម្លង គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីប្រៀបធៀបដើម្បីពិនិត្យមើលថាតើវចនានុក្រមឈ្មោះ ប្រវត្តិរូបដើម ដូចទៅនឹងវចនានុក្រមឈ្មោះ ប្រវត្តិរូបចម្លង ដែឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ត True បញ្ជាក់ប្រាប់ថាវចនានុក្រមទាំងពីរខាងលើនេះពិតជាដូចគ្នាមែន។

ប្រវត្តិរូបដើម != ប្រវត្តិរូបចម្លង គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីប្រៀបធៀបដើម្បីពិនិត្យមើលថាតើវចនានុក្រមឈ្មោះ ប្រវត្តិរូបដើម ខុសពីវចនានុក្រមឈ្មោះ ប្រវត្តិរូបចម្លង ដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ត False បញ្ជាក់ប្រាប់ថាប្រការដែលថាវចនានុក្រមទាំងពីរខាងលើនេះខុសគ្នាគឺខុស។

វចនានុក្រមពីរដូចគ្នាគឺជាវចនានុក្រមដែលមានធាតុទាំងអស់ដូចគ្នា។ ហើយម៉្យាងទៀត នៅក្នុងការធ្វើប្រមាណវិធីប្រៀបធៀបរវាងវចនានុក្រមផ្សេងៗ យើងមិនអាចប្រើប្រមាណសញ្ញាណាក្រៅពី == និង != នេះបានឡើយ។

ប្រមាណវិធីអត្តសញ្ញាណ

យើងអាចយកវចនានុក្រមផ្សេងៗមកធ្វើប្រមាណវិធីអត្តសញ្ញាណដូចខាងក្រោមនេះ៖

```
ប្រវត្តិរូបដើម = {"នាម":"កុសល", "គោត្តនាម":"កែវ", "អាយុ":35, "ភេទ":"ប្រុស"}
ប្រវត្តិរូបចម្លង = {"នាម":"កុសល", "គោត្តនាម":"កែវ", "អាយុ":35, "ភេទ":"ប្រុស"}
print(ប្រវត្តិរូបដើម is ប្រវត្តិរូបចម្លង)
print(ប្រវត្តិរូបដើម is not ប្រវត្តិរូបចម្លង)
```

ប្រវត្តិរូបដើម is ប្រវត្តិរូបចម្លង គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីអត្តសញ្ញាណ ដើម្បីពិនិត្យមើលថាតើវចនានុក្រមឈ្មោះ ប្រវត្តិរូបដើម និងវចនានុក្រមឈ្មោះ ប្រវត្តិរូបចម្លង ជាវត្ថុតែមួយមែនដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ត False បញ្ជាក់ប្រាប់ថាវចនានុក្រមទាំងពីរនេះមិនមែនជាវត្ថុតែមួយទេ។

ប្រវត្តិរូបដើម is notប្រវត្តិរូបចម្លង គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីអត្តសញ្ញាណ ដើម្បីពិនិត្យមើលថាតើវចនានុក្រមឈ្មោះ ប្រវត្តិរូបដើម និងវចនានុក្រមឈ្មោះ ប្រវត្តិរូបចម្លង ពិតជា

មិនមែនជាវត្ថុតែមួយមែនដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ថុ True បញ្ជាក់ប្រាប់ថាវចនានុក្រមទាំងពីរនេះពិតជាមិនមែនជាវត្ថុតែមួយមែន។

ប្រមាណវិធីរកធាតុ

យើងអាចយកវចនានុក្រមផ្សេងៗមកធ្វើប្រមាណវិធីរកធាតុដូចខាងក្រោមនេះ៖

```
ប្រវត្តិរូប = {"នាម":"កុសល", "គោត្តនាម":"កែវ", "អាយុ":35, "ភេទ":"ប្រុស"}
ពាក្យ = "គោត្តនាម"
print(ពាក្យ in ប្រវត្តិរូប)
print(ពាក្យ not in ប្រវត្តិរូប)
```

ពាក្យ in *ប្រវត្តិរូប* គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីរកធាតុដើម្បីពិនិត្យមើលថាតើនៅក្នុងវចនានុក្រមឈ្មោះ ប្រវត្តិរូប មានកូនសោរណាមួយដូចទៅនឹងកម្រងអក្សរឈ្មោះ ពាក្យ ដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ថុ True បញ្ជាក់ប្រាប់ថានៅក្នុងវចនានុក្រមឈ្មោះ ប្រវត្តិរូប ពិតជាមានកូនសោរដូចទៅនឹងវត្ថុឈ្មោះ ពាក្យ នោះមែន។

ពាក្យ not in *ប្រវត្តិរូប* គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីរកធាតុដើម្បីពិនិត្យមើលថាតើនៅក្នុងវចនានុក្រមឈ្មោះ ប្រវត្តិរូប ពិតជាគ្មានកូនសោរណាមួយដូចទៅនឹងកម្រងអក្សរឈ្មោះ ពាក្យ ដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ថុ False បញ្ជាក់ប្រាប់ថាប្រការដែលថានៅក្នុងវចនានុក្រមឈ្មោះ ប្រវត្តិរូប គ្មានកូនសោរណាមួយដូចទៅនឹងវត្ថុឈ្មោះ ពាក្យ នោះគឺខុស។

សំណុំ

សំណុំ (set) គឺជាសមាសវត្ថុម៉្យាងដែលនៅក្នុងនោះអាចមានវត្ថុជាច្រើនទៀត។ ដើម្បីបង្កើតសំណុំ យើងត្រូវសរសេរកម្មវិធីដូចខាងក្រោមនេះ៖

```
ថ្ងៃក្នុងសប្តាហ៍ = {"ច័ន្ទ", "អង្គារ", "ពុធ", "ព្រហស្បតិ៍", "សុក្រ", "សៅរ៍", "អាទិត្យ"}
print(ថ្ងៃក្នុងសប្តាហ៍)
```

ថ្ងៃក្នុងសប្តាហ៍ = {"ច័ន្ទ", "អង្គារ", "ពុធ", "ព្រហស្បតិ៍", "សុក្រ", "សៅរ៍", "អាទិត្យ"} គឺជាបញ្ជាតម្រូវឲ្យបង្កើតសំណុំមួយមានឈ្មោះថា ថ្ងៃក្នុងសប្តាហ៍ ។

វត្ថុនៅក្នុងសំណុំគឺជាធាតុរបស់សំណុំ។ ធាតុទាំងនោះគ្មានលេខរៀងច្បាស់លាស់ទេ គឺវាអាចផ្លាស់ប្តូរទីតាំងបានគ្រប់ពេលវេលា។ ម៉្យាងទៀតធាតុរបស់សំណុំត្រូវតែជាវត្ថុមិនអាចដោះដូរបាន។ ដូចនេះយើងមិនអាចយកកម្រងអថេរនិងឬវចនានុក្រមមកធ្វើជាធាតុរបស់សំណុំបានឡើយ។ ផ្ទុយទៅវិញ សំណុំគឺជាវត្ថុអាចដោះដូរបាន បានន័យថាធាតុនៅក្នុងសំណុំអាចត្រូវជំនួសដោយវត្ថុមិនអាចដោះដូរបានផ្សេងៗទៀតបាន។ មួយវិញទៀតធាតុនៅក្នុងសំណុំមិនអាចដូចគ្នាបានឡើយ ធាតុដែលដូចគ្នានឹងត្រូវបង្រួមមកឲ្យនៅតែមួយ។

ប្រមាណវិធីដកធាតុ

ប្រមាណវិធីដកធាតុ (differentiation) គឺជាប្រមាណវិធីទាំងឡាយណាដែលនៅក្នុងនោះមានការប្រើប្រាស់ប្រមាណសញ្ញាដកដើម្បីចម្លងយកសំណុំណាមួយមកដកធាតុដែលដូចទៅនឹងធាតុនៃសំណុំណាមួយទៀតចេញ ក្នុងគោលបំណងបង្កើតសំណុំថ្មីមួយទៀត។

យើងអាចយកសំណុំផ្សេងៗមកធ្វើប្រមាណវិធីដកធាតុដូចខាងក្រោមនេះ៖

```
ថ្ងៃក្នុងសប្តាហ៍ = {"ច័ន្ទ", "អង្គារ", "ពុធ", "ព្រហស្បតិ៍", "សុក្រ", "សៅរ៍", "អាទិត្យ"}
ថ្ងៃឈប់ = {"សៅរ៍", "អាទិត្យ"}
```

```
ថ្ងៃធ្វើការ = ថ្ងៃក្នុងសប្តាហ៍ - ថ្ងៃឈប់
print(ថ្ងៃធ្វើការ)
```

ថ្ងៃធ្វើការ = ថ្ងៃក្នុងសប្តាហ៍ - ថ្ងៃឈប់ គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីដកធាតុ ដោយចម្លងយកសំណុំឈ្មោះ ថ្ងៃក្នុងសប្តាហ៍ មកដកធាតុដែលដូចទៅនឹងធាតុនៃសំណុំឈ្មោះ ថ្ងៃឈប់ ដើម្បីបង្កើតសំណុំឈ្មោះ ថ្ងៃធ្វើការ ថ្មីមួយទៀត។

ប្រមាណវិធីប្រជុំ

ប្រមាណវិធីប្រជុំ (union) គឺជាប្រមាណវិធីទាំងឡាយណាដែលនៅក្នុងនោះមានការប្រើប្រាស់ប្រមាណសញ្ញា | ដើម្បីចម្លងយកធាតុនៅក្នុងសំណុំពីរមកប្រជុំគ្នាបង្កើតបានជាសំណុំថ្មីមួយទៀត។

យើងអាចយកសំណុំផ្សេងៗមកធ្វើប្រមាណវិធីប្រជុំដូចខាងក្រោមនេះ៖

```
ថ្ងៃក្នុងសប្តាហ៍ = {"ច័ន្ទ", "អង្គារ", "ពុធ", "ព្រហស្បតិ៍", "សុក្រ", "សៅរ៍", "អាទិត្យ"}
ឆ្នាំ = {2009, 2008}
ថ្ងៃឆ្នាំ = ថ្ងៃក្នុងសប្តាហ៍ | ឆ្នាំ
print(ថ្ងៃឆ្នាំ)
```

ថ្ងៃឆ្នាំ = ថ្ងៃក្នុងសប្តាហ៍ | ឆ្នាំ គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីប្រជុំ ដោយចម្លងយកធាតុនៅក្នុងសំណុំឈ្មោះ ថ្ងៃក្នុងសប្តាហ៍ និងធាតុនៅក្នុងសំណុំឈ្មោះ ឆ្នាំ មកប្រជុំគ្នាបង្កើតជាសំណុំថ្មីមួយទៀតមានឈ្មោះថា ថ្ងៃឆ្នាំ ។

នៅក្នុងប្រមាណវិធីប្រជុំ ធាតុទាំងអស់ដែលត្រូវបានចម្លងមកពីសំណុំទាំងពីរត្រូវបានដាក់បញ្ចូលគ្នាដើម្បីបង្កើតជាសំណុំថ្មីមួយទៀត តែចំពោះធាតុដែលដូចគ្នា គឺត្រូវជ្រើសរើសយកតែមួយតែប៉ុណ្ណោះ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
ថ្ងៃក្នុងសប្តាហ៍ = {"ច័ន្ទ", "អង្គារ", "ពុធ", "ព្រហស្បតិ៍", "សុក្រ", "សៅរ៏", "អាទិត្យ"}
ឆ្នាំថ្ងៃឈប់ = {2009, 2008, "សៅរ៏", "អាទិត្យ"}
ថ្ងៃឆ្នាំ = ថ្ងៃក្នុងសប្តាហ៍ | ឆ្នាំថ្ងៃឈប់
print(ថ្ងៃឆ្នាំ)
```

ថ្ងៃឆ្នាំ = ថ្ងៃក្នុងសប្តាហ៍ | ឆ្នាំថ្ងៃឈប់ គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីប្រជុំ ដោយចម្លងយកធាតុនៅក្នុងសំណុំឈ្មោះ ថ្ងៃក្នុងសប្តាហ៍ និងធាតុនៅក្នុងសំណុំឈ្មោះ ឆ្នាំថ្ងៃឈប់ មកប្រជុំគ្នាបង្កើតជាសំណុំមួយទៀតមានឈ្មោះថា ថ្ងៃឆ្នាំ ។

នៅក្នុងកម្មវិធីខាងលើនេះ ធាតុទាំងអស់នៅក្នុងសំណុំឈ្មោះ ថ្ងៃក្នុងសប្តាហ៍ និងសំណុំឈ្មោះ ឆ្នាំថ្ងៃឈប់ ត្រូវបានចម្លងយកមកផ្តុំគ្នាបង្កើតជាសំណុំមួយថ្មីទៀត។ ក៏ប៉ុន្តែចំពោះធាតុ "សៅរ៏" និង "អាទិត្យ" ដែលជាធាតុដូចគ្នានៅក្នុងសំណុំទាំងពីរ គឺត្រូវបានជ្រើសរើសយកតែមួយតែប៉ុណ្ណោះ។ បានន័យថា សំណុំថ្មីឈ្មោះ ថ្ងៃឆ្នាំ មានធាតុ "សៅរ៏" និងធាតុ "អាទិត្យ" តែមួយប៉ុណ្ណោះ។

ដោយហេតុថានៅក្នុងសំណុំមិនអាចមានធាតុដូចគ្នាបាន កត្តានេះធ្វើឲ្យសំណុំមានលក្ខណៈខុសពីកម្រងផ្សេងៗទៀត។ ធាតុនៅក្នុងកម្រងអាចជាវត្ថុប្រភេទណាក៏បានដែរ ហើយធាតុទាំងនោះអាចដូចៗគ្នាជាច្រើនទៀតផង។ ភាពខុសគ្នារវាងធាតុទាំងឡាយនៅក្នុងកម្រង គឺជាលេខរៀងរបស់វាដែលជាលំដាប់ថ្នាក់សម្រាប់សម្គាល់ធាតុទាំងនោះនៅក្នុងកម្រង។ ផ្ទុយមកវិញ ធាតុនៅក្នុងសំណុំគ្មានលេខរៀងឬលំដាប់ថ្នាក់ច្បាស់លាស់ទេ ហើយទីតាំងរបស់វាអាចត្រូវប្រែប្រួលបានគ្រប់ពេលវេលា។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
ថ្ងៃក្នុងសប្តាហ៍ = {"ច័ន្ទ", "អង្គារ", "ពុធ", "ព្រហស្បតិ៍", "សុក្រ", "សៅរ៍", "អាទិត្យ", "ច័ន្ទ"}
សប្តាហ៍ = ["ច័ន្ទ", "អង្គារ", "ពុធ", "ព្រហស្បតិ៍", "សុក្រ", "សៅរ៍", "អាទិត្យ", "ច័ន្ទ"]
print(ថ្ងៃក្នុងសប្តាហ៍)
print(សប្តាហ៍)
```

ថ្ងៃក្នុងសប្តាហ៍ = {"ច័ន្ទ", "អង្គារ", "ពុធ", "ព្រហស្បតិ៍", "សុក្រ", "សៅរ៍", "អាទិត្យ", "ច័ន្ទ"} គឺជាបញ្ជាតម្រូវឲ្យបង្កើតសំណុំឈ្មោះ ថ្ងៃក្នុងសប្តាហ៍ ដែលមានធាតុជាកម្រងអក្សរ "ច័ន្ទ" ចំនួនពីរនៅក្នុងនោះ។ ក៏ប៉ុន្តែនៅពេលសំណុំខាងលើនេះត្រូវបានបង្កើតរួចហើយ យើងឃើញថាធាតុទាំងអស់នៅក្នុងនោះត្រូវបានផ្លាស់ប្តូរទីតាំង ហើតធាតុដែលជាកម្រងអក្សរ "ច័ន្ទ" មានតែមួយប៉ុណ្ណោះ។

សប្តាហ៍ = ["ច័ន្ទ", "អង្គារ", "ពុធ", "ព្រហស្បតិ៍", "សុក្រ", "សៅរ៍", "អាទិត្យ", "ច័ន្ទ"] គឺជាបញ្ជាតម្រូវឲ្យបង្កើតកម្រងអថេរមួយមានឈ្មោះថា សប្តាហ៍ ដែលមានធាតុជាកម្រងអក្សរ "ច័ន្ទ" ចំនួនពីរនៅក្នុងនោះ។ បន្ទាប់ពីកម្រងអថេរខាងលើនេះត្រូវបានបង្កើតរួចហើយ យើងឃើញថាធាតុទាំងអស់នៅក្នុងនោះនៅតែរក្សាទីតាំងនៅដដែល ហើយធាតុដែលជាកម្រងអក្សរ "ច័ន្ទ" នៅតែមានចំនួនពីរដដែល។

ប្រមាណវិធីប្រសព្វ

ប្រមាណវិធីប្រសព្វ (intersection) គឺជាប្រមាណវិធីទាំងឡាយណាដែលនៅក្នុងនោះមានការប្រើប្រាស់ប្រមាណសញ្ញា & ដើម្បីចម្លងយកធាតុដូចគ្នានៅក្នុងសំណុំពីរមកបង្កើតជាសំណុំថ្មីមួយទៀត។

យើងអាចយកសំណុំផ្សេងៗមកធ្វើប្រមាណវិធីប្រសព្វដូចខាងក្រោមនេះ៖

```
ថ្ងៃក្នុងសប្តាហ៍ = {"ច័ន្ទ", "អង្គារ", "ពុធ", "ព្រហស្បតិ៍", "សុក្រ", "សៅរ៍", "អាទិត្យ"}
```

```
ឆ្នាំថ្ងៃឈប់ = {2009, 2008, "សៅរ៍", "អាទិត្យ"}
ធាតុដូចគ្នា = ថ្ងៃក្នុងសប្តាហ៍ & ឆ្នាំថ្ងៃឈប់
print(ធាតុដូចគ្នា)
```

ធាតុដូចគ្នា = ថ្ងៃក្នុងសប្តាហ៍ & ឆ្នាំថ្ងៃឈប់ គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីប្រសព្វ ដោយចម្លងយកធាតុដូចគ្នានៅក្នុងសំណុំឈ្មោះ ថ្ងៃក្នុងសប្តាហ៍ និងនៅក្នុងសំណុំឈ្មោះ ឆ្នាំថ្ងៃឈប់ មកបង្កើតជាសំណុំមួយថ្មីទៀតមានឈ្មោះថា ធាតុដូចគ្នា ។

ប្រមាណវិធីប្រជុំធាតុខុសគ្នា

ប្រមាណវិធីប្រជុំធាតុខុសគ្នា (symmetric difference) គឺជាប្រមាណវិធីទាំងឡាយណាដែលនៅក្នុងនោះមានការប្រើប្រាស់ប្រមាណសញ្ញា ^ ដើម្បីចម្លងយកធាតុខុសគ្នានៅក្នុងសំណុំពីរមកបង្កើតជាសំណុំថ្មីមួយទៀត។

យើងអាចយកសំណុំផ្សេងៗមកធ្វើប្រមាណវិធីប្រជុំធាតុខុសគ្នាដូចខាងក្រោមនេះ៖

```
ថ្ងៃក្នុងសប្តាហ៍ = {"ច័ន្ទ", "អង្គារ", "ពុធ", "ព្រហស្បតិ៍", "សុក្រ", "សៅរ៍", "អាទិត្យ"}
ឆ្នាំថ្ងៃឈប់ = {2009, 2008, "សៅរ៍", "អាទិត្យ"}
ធាតុខុសគ្នា = ថ្ងៃក្នុងសប្តាហ៍ ^ ឆ្នាំថ្ងៃឈប់
print(ធាតុខុសគ្នា)
```

ធាតុខុសគ្នា = ថ្ងៃក្នុងសប្តាហ៍ ^ ឆ្នាំថ្ងៃឈប់ គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីប្រជុំធាតុខុសគ្នា ដោយចម្លងយកធាតុខុសគ្នានៅក្នុងសំណុំឈ្មោះ ថ្ងៃក្នុងសប្តាហ៍ និងនៅក្នុងសំណុំឈ្មោះ ថ្ងៃឈប់ មកបង្កើតជាសំណុំថ្មីមួយទៀតមានឈ្មោះថា ធាតុខុសគ្នា ។

ប្រមាណវិធីតក្កវិទ្យា

យើងអាចយកសំណុំផ្សេងៗមកធ្វើប្រមាណវិធីតក្កវិទ្យាដូចខាងក្រោមនេះ៖

```
ថ្ងៃក្នុងសប្តាហ៍ = {"ច័ន្ទ", "អង្គារ", "ពុធ", "ព្រហស្បតិ៍", "សុក្រ", "សៅរ៍", "អាទិត្យ"}
សំណុំទទេ = set()
print(ថ្ងៃក្នុងសប្តាហ៍ and សំណុំទទេ)
print(ថ្ងៃក្នុងសប្តាហ៍ or សំណុំទទេ)
print(not ថ្ងៃក្នុងសប្តាហ៍)
print(not សំណុំទទេ)
```

សំណុំទទេ = set() គឺជាបញ្ជាតម្រូវឲ្យបង្កើតសំណុំទទេមួយមានឈ្មោះថា សំណុំទទេ ។

យើងត្រូវធ្វើការកត់សំគាល់ថា ដើម្បីបង្កើតសំណុំទទេ យើងត្រូវសរសេរ set() គឺយើងមិនត្រូវសរសេរថា {} ដូចនេះឡើយ។ ការសរសេរ {} គឺជាការបង្កើតវចនានុក្រមទទេ។

ថ្ងៃក្នុងសប្តាហ៍ and *សំណុំទទេ* គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីតក្កវិទ្យា ដោយប្រើប្រមាណសញ្ញា and ។ ដោយសំណុំឈ្មោះ ថ្ងៃក្នុងសប្តាហ៍ ជាសំណុំមានធាតុនៅក្នុងនោះ ដូចនេះវាសមមូលនឹងតក្កវត្ត True ដែលនាំឲ្យលទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាសំណុំឈ្មោះ សំណុំទទេ ព្រោះវាជាប្រមាណអង្គនៅខាងស្តាំ។

ថ្ងៃក្នុងសប្តាហ៍ or *សំណុំទទេ* គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីតក្កវិទ្យា ដោយប្រើប្រមាណសញ្ញា or ។ ដោយសំណុំឈ្មោះ ថ្ងៃក្នុងសប្តាហ៍ ជាសំណុំមានធាតុនៅក្នុងនោះ ដូចនេះវាសមមូលនឹងតក្កវត្ត True ដែលនាំឲ្យលទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាសំណុំនេះតែម្តង ។

not *ថ្ងៃក្នុងសប្តាហ៍* គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីតក្កវិទ្យាដោយប្រើប្រមាណសញ្ញា not ជាមួយនឹងសំណុំឈ្មោះ ថ្ងៃក្នុងសប្តាហ៍ ។ ដោយសំណុំនេះជាសំណុំមានធាតុនៅក្នុងនោះ ដូចនេះវាសមមូលនឹងតក្កវត្ត True ដែលនាំឲ្យលទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ត False ។

not *សំណុំទទេ* គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីតក្កវិទ្យាដោយប្រើប្រមាណសញ្ញា not ជាមួយនឹងសំណុំឈ្មោះ ថ្ងៃក្នុងសប្តាហ៍ ។ ដោយសំណុំនេះជាសំណុំទទេ ដូចនេះវាសមមូលនឹងតក្កវត្ត False ដែលនាំឲ្យលទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ត True ។

ប្រមាណវិធីប្រៀបធៀប

យើងអាចយកសំណុំផ្សេងៗមកធ្វើប្រមាណវិធីប្រៀបធៀបដូចខាងក្រោមនេះ៖

```
ថ្ងៃក្នុងសប្តាហ៍ = {"ច័ន្ទ", "អង្គារ", "ពុធ", "ព្រហស្បតិ៍", "សុក្រ", "សៅរ៍", "អាទិត្យ"}
ថ្ងៃសម្រាក = {"សៅរ៍", "អាទិត្យ" }
print(ថ្ងៃក្នុងសប្តាហ៍ == ថ្ងៃសម្រាក)
print(ថ្ងៃក្នុងសប្តាហ៍ != ថ្ងៃសម្រាក)
print(ថ្ងៃក្នុងសប្តាហ៍ > ថ្ងៃសម្រាក)
print(ថ្ងៃក្នុងសប្តាហ៍ < ថ្ងៃសម្រាក)
print(ថ្ងៃក្នុងសប្តាហ៍ >= ថ្ងៃសម្រាក)
print(ថ្ងៃក្នុងសប្តាហ៍ <= ថ្ងៃសម្រាក)
```

ប្រមាណវិធីប្រៀបធៀបរវាងសំណុំផ្សេងៗពុំមែនជាប្រមាណវិធីប្រៀបធៀបតាមរបៀបវចនានុក្រមឡើយ។ ផ្ទុយទៅវិញ ប្រមាណវិធីប្រៀបធៀបរវាងសំណុំផ្សេងៗ គឺជាប្រមាណវិធីម្យ៉ាងដូចតទៅនេះ៖

ថ្ងៃក្នុងសប្តាហ៍ == ថ្ងៃសម្រាក គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីប្រៀបធៀបដើម្បីពិនិត្យមើលថាតើសំណុំឈ្មោះ ថ្ងៃក្នុងសប្តាហ៍ និងសំណុំឈ្មោះ ថ្ងៃសម្រាក មានចំនួនធាតុស្មើគ្នានិងដូចគ្នាទាំងអស់ដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ថុ False បញ្ជាក់ប្រាប់ថាសំណុំទាំងពីរនេះគ្មានធាតុមានចំនួនស្មើគ្នានិងដូចគ្នាទាំងអស់នោះទេ។

ថ្ងៃក្នុងសប្តាហ៍ != ថ្ងៃសម្រាក គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីប្រៀបធៀបដើម្បីពិនិត្យមើលថាតើសំណុំឈ្មោះ ថ្ងៃក្នុងសប្តាហ៍ និងសំណុំឈ្មោះ ថ្ងៃសម្រាក មានចំនួនធាតុខុសគ្នានិងឬខុសគ្នាដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ថុ True បញ្ជាក់ប្រាប់ថាសំណុំទាំងពីរនេះពិតជាមានធាតុមានចំនួនខុសគ្នានិងឬមានធាតុខុសគ្នាមែន។

ថ្ងៃក្នុងសប្តាហ៍ > ថ្ងៃសម្រាក គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីប្រៀបធៀបដើម្បីពិនិត្យមើលថាតើសំណុំឈ្មោះ ថ្ងៃក្នុងសប្តាហ៍ ជា *សំណុំមេ* (superset) នៃសំណុំឈ្មោះ ថ្ងៃសម្រាក មែនដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ថុ True បញ្ជាក់ប្រាប់ថាសំណុំឈ្មោះ ថ្ងៃក្នុងសប្តាហ៍ ពិតជាសំណុំមេនៃសំណុំឈ្មោះ ថ្ងៃសម្រាក មែន។

សំណុំឈ្មោះ ថ្ងៃក្នុងសប្តាហ៍ ជាសំណុំមេនៃសំណុំឈ្មោះ ថ្ងៃសម្រាក មានន័យថាសំណុំឈ្មោះ ថ្ងៃក្នុងសប្តាហ៍ មានធាតុច្រើនជាងសំណុំឈ្មោះ ថ្ងៃសម្រាក ហើយធាតុទាំងអស់នៃសំណុំឈ្មោះ ថ្ងៃសម្រាក មាននៅក្នុងសំណុំឈ្មោះ ថ្ងៃក្នុងសប្តាហ៍ ។

ថ្ងៃក្នុងសប្តាហ៍ < ថ្ងៃសម្រាក គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីប្រៀបធៀបដើម្បីពិនិត្យមើលថាតើសំណុំឈ្មោះ ថ្ងៃក្នុងសប្តាហ៍ ជា *សំណុំរង* (subset) នៃសំណុំឈ្មោះ ថ្ងៃសម្រាក មែនដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ថុ False បញ្ជាក់ប្រាប់ថាសំណុំឈ្មោះ ថ្ងៃក្នុងសប្តាហ៍ មិនមែនជាសំណុំរងនៃសំណុំឈ្មោះ ថ្ងៃសម្រាក ទេ។

សំណុំឈ្មោះ ថ្ងៃក្នុងសប្តាហ៍ ជាសំណុំរងនៃសំណុំឈ្មោះ ថ្ងៃសម្រាក មានន័យថាសំណុំឈ្មោះ ថ្ងៃសម្រាក មានធាតុច្រើនជាងសំណុំឈ្មោះ ថ្ងៃក្នុងសប្តាហ៍ ហើយធាតុទាំងអស់នៃសំណុំឈ្មោះ ថ្ងៃក្នុងសប្តាហ៍ មាននៅក្នុងសំណុំឈ្មោះ ថ្ងៃសម្រាក ។

ថ្ងៃក្នុងសប្តាហ៍ >= ថ្ងៃសម្រាក គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីប្រៀបធៀបដើម្បីពិនិត្យមើលថាតើសំណុំឈ្មោះ ថ្ងៃក្នុងសប្តាហ៍ ជាសំណុំមេនៃសំណុំឈ្មោះ ថ្ងៃសម្រាក ឬដូចទៅនឹងសំណុំឈ្មោះ ថ្ងៃសម្រាក នោះមែនដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ថុ True បញ្ជាក់ប្រាប់ថាសំណុំឈ្មោះ ថ្ងៃក្នុងសប្តាហ៍ ពិតជាសំណុំមេនៃសំណុំឈ្មោះ ថ្ងៃសម្រាក ឬដូចទៅនឹងសំណុំឈ្មោះ ថ្ងៃសម្រាក នោះមែន។

ថ្ងៃក្នុងសប្តាហ៍ <= ថ្ងៃសម្រាក គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីប្រៀបធៀបដើម្បីពិនិត្យមើលថាតើសំណុំឈ្មោះ ថ្ងៃក្នុងសប្តាហ៍ ជាសំណុំរងនៃសំណុំឈ្មោះ ថ្ងៃសម្រាក ឬដូចទៅនឹងសំណុំឈ្មោះ ថ្ងៃសម្រាក នោះមែនដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ថុ False បញ្ជាក់ប្រាប់ថាសំណុំឈ្មោះ ថ្ងៃក្នុងសប្តាហ៍ មិនមែនជាសំណុំរងនៃសំណុំឈ្មោះ ថ្ងៃសម្រាក ឬដូចទៅនឹងសំណុំឈ្មោះ ថ្ងៃសម្រាក នោះទេ។

ប្រមាណវិធីរកធាតុ

យើងអាចយកសំណុំផ្សេងៗមកធ្វើប្រមាណវិធីរកធាតុដូចខាងក្រោមនេះ៖

```
ថ្ងៃក្នុងសប្តាហ៍ = {"ច័ន្ទ", "អង្គារ", "ពុធ", "ព្រហស្បតិ៍", "សុក្រ", "សៅរ៍", "អាទិត្យ"}
ថ្ងៃក្រហម = "អាទិត្យ"
print(ថ្ងៃក្រហម in ថ្ងៃក្នុងសប្តាហ៍)
print(ថ្ងៃក្រហម not in ថ្ងៃក្នុងសប្តាហ៍)
```

ថ្ងៃក្រហម in *ថ្ងៃក្នុងសប្តាហ៍* គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីរកធាតុដើម្បីពិនិត្យមើលថាតើនៅក្នុងសំណុំឈ្មោះ ថ្ងៃក្នុងសប្តាហ៍ មានធាតុណាមួយដូចទៅនឹងវត្ថុឈ្មោះ ថ្ងៃក្រហម ដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ថុ True បញ្ជាក់ប្រាប់ថានៅក្នុងសំណុំឈ្មោះ ថ្ងៃក្នុងសប្តាហ៍ ពិតជាមានធាតុណាមួយដូចទៅនឹងវត្ថុឈ្មោះ ថ្ងៃក្រហម នោះមែន។

ថ្ងៃក្រហម not in *ថ្ងៃក្នុងសប្តាហ៍* គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីរកធាតុដើម្បីពិនិត្យមើលថាតើនៅក្នុងសំណុំឈ្មោះ ថ្ងៃក្នុងសប្តាហ៍ ពិតជាគ្មានធាតុណាមួយដូចទៅនឹងវត្ថុឈ្មោះ ថ្ងៃក្រហម ដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ថុ False បញ្ជាក់ប្រាប់ថាប្រការដែលថានៅក្នុងសំណុំឈ្មោះ ថ្ងៃក្នុងសប្តាហ៍ គ្មានធាតុណាមួយដូចទៅនឹងវត្ថុឈ្មោះ ថ្ងៃក្រហម នោះគឺខុស។

ប្រមាណវិធីអត្តសញ្ញាណ

យើងអាចយកសំណុំផ្សេងៗមកធ្វើប្រមាណវិធីអត្តសញ្ញាណដូចខាងក្រោមនេះ៖

```
ថ្ងៃក្នុងសប្តាហ៍ = {"ច័ន្ទ", "អង្គារ", "ពុធ", "ព្រហស្បតិ៍", "សុក្រ", "សៅរ៍", "អាទិត្យ"}
ថ្ងៃទាំងប្រាំពីរ = {"ច័ន្ទ", "អង្គារ", "ពុធ", "ព្រហស្បតិ៍", "សុក្រ", "សៅរ៍", "អាទិត្យ"}
print(ថ្ងៃក្នុងសប្តាហ៍ is ថ្ងៃទាំងប្រាំពីរ)
print(ថ្ងៃក្នុងសប្តាហ៍ is not ថ្ងៃទាំងប្រាំពីរ)
```

ថ្ងៃក្នុងសប្តាហ៍ is *ថ្ងៃទាំងប្រាំពីរ* គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីអត្តសញ្ញាណដើម្បីពិនិត្យមើលថាតើសំណុំឈ្មោះ ថ្ងៃក្នុងសប្តាហ៍ និងសំណុំឈ្មោះ ថ្ងៃទាំងប្រាំពីរ ជាវត្ថុតែមួយមែនដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ថុ False បញ្ជាក់ប្រាប់ថាសំណុំទាំងពីរនោះមិនមែនជាវត្ថុតែមួយទេ គឺវាគ្រាន់តែជាវត្ថុដូចគ្នាតែប៉ុណ្ណោះ។

ថ្ងៃក្នុងសប្ដាហ៍ is not *ថ្ងៃទាំងប្រាំពីរ* គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីអត្តសញ្ញាណដើម្បីពិនិត្យមើលថាតើសំណុំឈ្មោះ ថ្ងៃក្នុងសប្ដាហ៍ និងសំណុំឈ្មោះ ថ្ងៃទាំងប្រាំពីរ ពិតជាមិនមែនជាវត្ថុតែមួយមែនដែរឬទេ។ លទ្ធផលបានមកពីប្រមាណវិធីនេះគឺជាតក្កវត្ថុ True បញ្ជាក់ប្រាប់ថាសំណុំទាំងពីរនោះពិតជាមិនមែនជាវត្ថុតែមួយមែន។

សមាសវត្ថុនៃសមាសវត្ថុ

សមាសវត្ថុទាំងឡាយអាចមានធាតុជាសមាសវត្ថុផ្សេងទៀតបាន។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
ថ្ងៃក្នុងសប្ដាហ៍ = ("ច័ន្ទ", "អង្គារ", "ពុធ", "ព្រហស្បតិ៍", "សុក្រ", "សៅរ៍", "អាទិត្យ")
វិមាត្រ = ["បណ្ដោយ", "ទទឹង", "កំពស់"]
ប្រវត្តិរូប = {"នាម":"កុសល", "នាមត្រកូល":"កែវ", "អាយុ":35, "ទីលំនៅ":"កម្ពុជា"}
សំណុំតួលេខ = {100, 1.754, 16}
កម្រងថេរចម្រុះ = (ថ្ងៃក្នុងសប្ដាហ៍, វិមាត្រ, ប្រវត្តិរូប, សំណុំតួលេខ)
កម្រងអថេរចម្រុះ = [ថ្ងៃក្នុងសប្ដាហ៍, វិមាត្រ, ប្រវត្តិរូប, សំណុំតួលេខ]
វចនានុក្រមចម្រុះ = {1:ថ្ងៃក្នុងសប្ដាហ៍, 2:វិមាត្រ, 3:ប្រវត្តិរូប, 4:សំណុំតួលេខ}
សំណុំចម្រុះ = {ថ្ងៃក្នុងសប្ដាហ៍}
print(កម្រងថេរចម្រុះ)
print(កម្រងអថេរចម្រុះ)
print(វចនានុក្រមចម្រុះ)
print(សំណុំចម្រុះ)
```

កម្រងថេរចម្រុះ = (ថ្ងៃក្នុងសប្តាហ៍, វិមាត្រ, ប្រវត្តិរូប, សំណុំតួលេខ) គឺជាបញ្ជាតម្រូវឲ្យបង្កើតកម្រងថេរមួយមានឈ្មោះថា កម្រងចម្រុះ ដែលមានធាតុជាកម្រងថេរឈ្មោះ ថ្ងៃក្នុងសប្តាហ៍ កម្រងអថេរឈ្មោះ វិមាត្រ វចនានុក្រមឈ្មោះ ប្រវត្តិរូប និងសំណុំឈ្មោះ សំណុំតួលេខ ។

កម្រងអថេរចម្រុះ = [ថ្ងៃក្នុងសប្តាហ៍, វិមាត្រ, ប្រវត្តិរូប, សំណុំតួលេខ] គឺជាបញ្ជាតម្រូវឲ្យបង្កើតកម្រងអថេរឈ្មោះ កម្រងអថេរចម្រុះ មួយដែលមានធាតុជាកម្រងថេរឈ្មោះ ថ្ងៃក្នុងសប្តាហ៍ កម្រងអថេរឈ្មោះ វិមាត្រ វចនានុក្រមឈ្មោះ ប្រវត្តិរូប និងសំណុំឈ្មោះ សំណុំតួលេខ ។

វចនានុក្រមចម្រុះ = {1:ថ្ងៃក្នុងសប្តាហ៍, 2:វិមាត្រ, 3:ប្រវត្តិរូប, 4:សំណុំតួលេខ} គឺជាបញ្ជាតម្រូវឲបង្កើតវចនានុក្រមឈ្មោះ វចនានុក្រមចម្រុះ មួយដែលនៅក្នុងនោះមានកម្រងថេរឈ្មោះ ថ្ងៃក្នុងសប្តាហ៍ កម្រងអថេរឈ្មោះ វិមាត្រ វចនានុក្រមឈ្មោះ ប្រវត្តិរូប និងសំណុំឈ្មោះ សំណុំតួលេខ ជាតម្លៃជាប់នឹងកូនសោរ 1, 2, 3, 4 រៀងគ្នា។

សំណុំចម្រុះ = {ថ្ងៃក្នុងសប្តាហ៍} គឺជាបញ្ជាតម្រូវឲ្យបង្កើតសំណុំឈ្មោះ សំណុំចម្រុះ មួយដែលមានធាតុជាកម្រងថេរឈ្មោះ ថ្ងៃក្នុងសប្តាហ៍ ។

យើងគួររំលឹកឡើងវិញថា ធាតុនៅក្នុងសំណុំមិនអាចជាវត្ថុអាចដោះដូរបានឡើយ ដូចនេះក្រៅតែកម្រងថេរនិងឬកម្រងអក្សរ យើងមិនអាចយកសមាសវត្ថុណាផ្សេងមកធ្វើជាធាតុនៃសំណុំបានឡើយ។

ក្នុងករណីសមាសវត្ថុមួយមានធាតុជាសមាសវត្ថុផ្សេងៗទៀត ដើម្បីចម្លងយកធាតុឬតម្លៃនៃសមាសវត្ថុដែលជាធាតុយកមកប្រើ យើងត្រូវធ្វើដូចខាងក្រោមនេះ៖

ថ្ងៃក្នុងសប្តាហ៍ = ("ច័ន្ទ", "អង្គារ", "ពុធ", "ព្រហស្បតិ៍", "សុក្រ", "សៅរ៍", "អាទិត្យ")

```
វិមាត្រ = ["បណ្តោយ", "ទទឹង", "កំពស់"]
ប្រវត្តិរូប = {"នាម":"កុសល", "នាមត្រកូល":"កែវ", "អាយុ":35, "ទីលំនៅ":"កម្ពុជា"}
សំណុំតួលេខ = {100, 1.754, 16}
កម្រងថេរចម្រុះ = (ថ្ងៃក្នុងសប្តាហ៍, វិមាត្រ, ប្រវត្តិរូប, សំណុំតួលេខ)
កម្រងអថេរចម្រុះ = [ថ្ងៃក្នុងសប្តាហ៍, វិមាត្រ, ប្រវត្តិរូប, សំណុំតួលេខ]
វចនានុក្រមចម្រុះ = {1:ថ្ងៃក្នុងសប្តាហ៍, 2:វិមាត្រ, 3:ប្រវត្តិរូប, 4:សំណុំតួលេខ}
print(កម្រងថេរចម្រុះ[1][-1])
print(កម្រងអថេរចម្រុះ[2]["នាមត្រកូល"])
print(វចនានុក្រមចម្រុះ[3]["អាយុ"])
```

print(កម្រងថេរចម្រុះ[1][-1]) គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីលេខរៀងដើម្បីចម្លងយកធាតុមានលេខរៀង -1 នៅក្នុងកម្រងអថេរឈ្មោះ វិមាត្រ ដែលជាធាតុមានលេខរៀង 1 នៅក្នុងកម្រងថេរឈ្មោះ កម្រងថេរចម្រុះ ។

print(កម្រងអថេរចម្រុះ[2]["នាមត្រកូល"]) គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីលេខរៀងដើម្បីចម្លងយកតម្លៃជាប់នឹងកូនសោរ “ត្រកូល” នៅក្នុងវចនានុក្រមឈ្មោះ ប្រវត្តិរូប ដែលជាធាតុមានលេខរៀង 2 នៅក្នុងកម្រងអថេរឈ្មោះ កម្រងអថេរចម្រុះ ។

print(វចនានុក្រមចម្រុះ[3]["អាយុ"]) គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីលេខរៀងដើម្បីចម្លងយកតម្លៃជាប់នឹងកូនសោរ “អាយុ” នៅក្នុងវចនានុក្រមឈ្មោះ ប្រវត្តិរូប ដែលជាតម្លៃជាប់នឹងកូនសោរ 3 នៅក្នុងវចនានុក្រមឈ្មោះ វចនានុក្រមចម្រុះ ។

ប្រភេទនៃបញ្ជា

នៅក្នុងភាសា Python មានបញ្ជាជាច្រើនប្រភេទដែលយើងអាចយកមកប្រើសម្រាប់សរសេរកម្មវិធីតម្រូវឲ្យកំព្យូទ័រយកទៅអនុវត្តដើម្បីដោះស្រាយបញ្ហាផ្សេងៗ។ បញ្ជាទាំងនោះមាន៖

បញ្ជាចាត់តាំង

បញ្ជាចាត់តាំង (assignment statement) គឺជាសញ្ញា = ដែលជាបញ្ជាតម្រូវឲភ្ជាប់ឈ្មោះណាមួយទៅនឹងវត្ថុណាមួយ។ ពោលគឺជាបញ្ជាចាត់តាំងឈ្មោះណាមួយឲ្យដំណាងឲ្យវត្ថុណាមួយ ។ យើងគួររំលឹកឡើងវិញថា ដើម្បីចាត់តាំងឈ្មោះណាមួយឲ្យដំណាងឲ្យវត្ថុណាមួយ យើងត្រូវសរសេរកម្មវិធីដូចខាងក្រោមនេះ៖

```
ប្រាក់ចំណេញ = 1000
print(ប្រាក់ចំណេញ)
```

ប្រាក់ចំណេញ = 1000 គឺជាការចាត់តាំងឈ្មោះ ប្រាក់ចំណេញ ឲ្យដំណាងឲ្យវត្ថុដែលជាលេខ 1000 ។

ក្រៅពីការចាត់តាំងឈ្មោះមួយឲ្យដំណាងឲ្យវត្ថុមួយ យើងអាចចាត់តាំងឈ្មោះជាច្រើនឲ្យដំណាងឲ្យវត្ថុផ្សេងៗគ្នាដោយសរសេរកម្មវិធីដូចខាងក្រោមនេះ៖

```
ថ្លៃលក់, ថ្លៃទិញ, ប្រាក់ចំណេញ = 1000, 900, 100
print(ថ្លៃលក់)
print(ថ្លៃទិញ)
print(ប្រាក់ចំណេញ)
```

ថ្លៃលក់, ថ្លៃទិញ, ប្រាក់ចំណេញ = 1000, 900, 100 គឺជាការចាត់តាំងឈ្មោះ ថ្លៃលក់, ថ្លៃទិញ, ប្រាក់ចំណេញ ឲ្យជំណាងឲ្យវត្ថុលេខ 1000, 900, 100 រៀងគ្នា។

យើងបានដឹងរួចមកហើយថា ការសរសេរ ថ្លៃលក់, ថ្លៃទិញ, ប្រាក់ចំណេញ និងការសរសេរ 1000, 900, 100 គឺជាការបង្កើតកម្រងថេរ (ថ្លៃលក់, ថ្លៃទិញ, ប្រាក់ចំណេញ) និងកម្រងថេរ (1000, 900, 100) ។ ដូចនេះការសរសេរ ថ្លៃលក់, ថ្លៃទិញ, ប្រាក់ចំណេញ = 1000, 900, 100 សមមូលនឹង (ថ្លៃលក់, ថ្លៃទិញ, ប្រាក់ចំណេញ) = (1000, 900, 100) ។ សរុបមកការសរសេរ៖ ថ្លៃលក់, ថ្លៃទិញ, ប្រាក់ចំណេញ = 1000, 900, 100 គឺជាការចាត់តាំងឈ្មោះនៅក្នុងកម្រងថេរមួយឲ្យជំណាងឲ្យវត្ថុនៅក្នុងកម្រងថេរមួយទៀត។

ដូចគ្នាដែរ ក្រៅពីកម្រងថេរ យើងអាចចាត់តាំងឈ្មោះនៅក្នុងកម្រងណាមួយក៏បានដែរឲ្យជំណាងឲ្យវត្ថុនៅក្នុងកម្រងណាមួយទៀតក៏បានដែរ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
[ថ្លៃលក់, ថ្លៃទិញ, ប្រាក់ចំណេញ] = [1000, 900, 100]
(បណ្តោយ, ទទឹង, កំពស់) = [30, 15, 10]
[នាមត្រកូល, នាម, អាយុ] = ("កុសល", "កែវ", 35)
print(ថ្លៃលក់)
print(ទទឹង)
print(នាមត្រកូល)
```

[ថ្លៃលក់, ថ្លៃទិញ, ប្រាក់ចំណេញ] = [1000, 900, 100] គឺជាការចាត់តាំងឈ្មោះនៅក្នុងកម្រងអថេរ *[ថ្លៃលក់, ថ្លៃទិញ, ប្រាក់ចំណេញ]* ឲ្យជំណាងឲ្យវត្ថុនៅក្នុងកម្រងអថេរ [1000, 900, 100] រៀងគ្នា។

(បណ្តោយ, ទទឹង, កំពស់) = [30, 15, 10] គឺជាការចាត់តាំងឈ្មោះនៅក្នុងកម្រងថេរ *(បណ្តោយ, ទទឹង, កំពស់)* ឲ្យដំណាងឲ្យវត្ថុនៅក្នុងកម្រងអថេរ [30, 15, 10] រៀងគ្នា។

[*នាមត្រកូល, នាម, អាយុ*] = ("*កុសល*", "*កែវ*", 35) គឺជាការចាត់តាំងឈ្មោះនៅក្នុងកម្រងអថេរ [*នាមត្រកូល, នាម, អាយុ*] ឲ្យដំណាងឲ្យវត្ថុនៅក្នុងកម្រងថេរ ("*កុសល*", "*កែវ*", 35) រៀងគ្នា។

លើសពីនេះទៀត យើងអាចប្រើបញ្ជាចាត់តាំងសម្រាប់ប្តូរឈ្មោះរបស់វត្ថុពីរទៅវិញទៅមកដោយសរសេរកម្មវិធីដូចខាងក្រោមនេះ៖

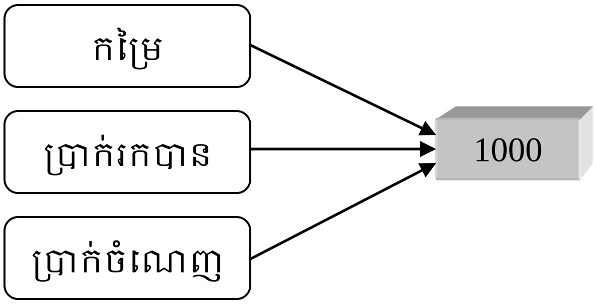
```
បណ្តោយ = 173.5
ប្រវែង = 200
បណ្តោយ, ប្រវែង = ប្រវែង, បណ្តោយ
print(បណ្តោយ)
print(ប្រវែង)
```

បណ្តោយ, ប្រវែង = *ប្រវែង, បណ្តោយ* គឺជាការប្រើបញ្ជាចាត់តាំងដើម្បីប្តូរឈ្មោះរបស់វត្ថុឈ្មោះ បណ្តោយ និងវត្ថុឈ្មោះ ប្រវែង ទៅវិញទៅមក។

ក្រៅពីការចាត់តាំងឈ្មោះមួយឲ្យដំណាងឲ្យវត្ថុមួយ យើងអាចចាត់តាំងឈ្មោះជាច្រើនឲ្យដំណាងឲ្យវត្ថុតែមួយដោយសរសេរកម្មវិធីដូចខាងក្រោមនេះ៖

```
កម្រៃ = ប្រាក់រកបាន = ប្រាក់ចំណេញ = 1000
print(កម្រៃ)
print(ប្រាក់រកបាន)
print(ប្រាក់ចំណេញ)
```

កម្រៃ = ប្រាក់រកបាន = ប្រាក់ចំណេញ = 1000 គឺជាការចាត់តាំងឈ្មោះ កម្រៃ ប្រាក់រកបាន និង ប្រាក់ចំណេញ ឲ្យដំណាងឲ្យវត្ថុតែមួយដូចគ្នាដែលជាលេខ 1000 ។

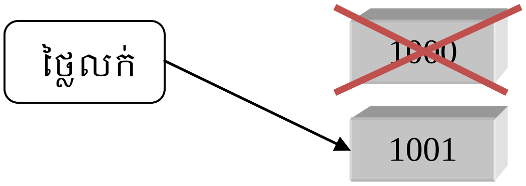


ស្វ័យប្រមាណវិធី

នៅក្នុងភាសា Python យើងអាចធ្វើប្រមាណវិធីបូកម៉្យាងដូចខាងក្រោមនេះ៖

```
ថ្លៃលក់ = 1000
ថ្លៃលក់ = ថ្លៃលក់ + 1
print(ថ្លៃលក់)
```

ថ្លៃលក់ = ថ្លៃលក់ + 1 គឺជាបញ្ជាតម្រូវឲ្យចម្លងយកវត្ថុឈ្មោះ ថ្លៃលក់ មកបូកនឹង 1 បង្កើតជាវត្ថុថ្មីមួយទៀត រួចចាត់តាំងឈ្មោះ ថ្លៃលក់ ដដែលឲ្យដំណាងឲ្យវត្ថុថ្មីនោះវិញ។ ជាលទ្ធផល វត្ថុចាស់ដែលជាលេខ 1000 ត្រូវបាត់ឈ្មោះនិងត្រូវលុបចេញពីក្នុងសតិរបស់កំព្យូទ័រដោយយន្តការបោសសម្អាត។



ទង្វើដូចខាងលើនេះហៅថា *ស្វ័យប្រមាណវិធី* (augmented assignment) ដែលអាចត្រូវសរសេរតាមរបៀបម៉្យាងទៀតដូចខាងក្រោមនេះ៖

```
ថ្លៃលក់ = 1000
ថ្លៃលក់ += 1
print(ថ្លៃលក់)
```

ថ្លៃលក់ += 1 គឺជាបញ្ជាតម្រូវឲ្យធ្វើស្វ័យប្រមាណវិធីដើម្បីបង្កើតវត្ថុឈ្មោះ ថ្លៃលក់ ថ្មីដែលធំជាងមុន 1 ។

សរុបមកយើងឃើញថា ការធ្វើស្វ័យប្រមាណវិធីដូចខាងលើនេះមិនមែនជាការយកវត្ថុចាស់មកកែប្រែទេ វាគឺជាការយកវត្ថុចាស់មកដូរថ្មី។ ហើយវត្ថុចាស់ត្រូវលុបចោលបើសិនជាគ្មានឈ្មោះណាមួយនៅជាប់នឹងវាទេនោះ។ យើងត្រូវធ្វើការកត់សំគាល់ថា ការយកវត្ថុមកកែប្រែ និងការយកវត្ថុមកដូរថ្មី គឺជារឿងពីរខុសគ្នា។ ការយកវត្ថុមកកែប្រែបានន័យថា គឺជាការយកវត្ថុដដែលមកដោះដូរធាតុរបស់វា ចំណែកឯការយកវត្ថុមកដូរថ្មី គឺជាការយកវត្ថុទាំងមូលមកប្តូរជាមួយនឹងវត្ថុថ្មីផ្សេងទៀត។ ម៉្យាងទៀត ការដូរថ្មីគឺត្រូវធ្វើឡើងដោយយកឈ្មោះរបស់វត្ថុចាស់ទៅភ្ជាប់នឹងវត្ថុថ្មី។ ដូចនេះការធ្វើស្វ័យប្រមាណវិធីកន្លងមក គឺជាការយកវត្ថុមកដូរថ្មី មិនមែនជាការយកវត្ថុដដែលមកកែប្រែឡើយ។

ក្រៅពីការធ្វើស្វ័យប្រមាណវិធីដោយប្រើប្រមាណសញ្ញាបូក (+) យើងអាចធ្វើស្វ័យប្រមាណវិធីដោយប្រើប្រមាណសញ្ញាផ្សេងៗទៀតដូចខាងក្រោមនេះ៖

```
ថ្លៃលក់ = 1000
```

```
ថ្លៃលក់ += 2
ថ្លៃលក់ -= 2
ថ្លៃលក់ *= 2
ថ្លៃលក់ /= 2
ថ្លៃលក់ //= 2
ថ្លៃលក់ %= 2
ថ្លៃលក់ **= 2
```

ថ្លៃលក់ += 2 គឺជាបញ្ជាតម្រូវឲ្យធ្វើស្វ័យប្រមាណវិធីដើម្បីបង្កើតវត្ថុឈ្មោះ ថ្លៃលក់ ថ្មីធំជាងមុន 2 ។

ថ្លៃលក់ -= 2 គឺជាបញ្ជាតម្រូវឲ្យធ្វើស្វ័យប្រមាណវិធីដើម្បីបង្កើតវត្ថុឈ្មោះ ថ្លៃលក់ ថ្មីតូចជាងមុន 2 ។

*ថ្លៃលក់ *= 2* គឺជាបញ្ជាតម្រូវឲ្យធ្វើស្វ័យប្រមាណវិធីដើម្បីបង្កើតវត្ថុឈ្មោះ ថ្លៃលក់ ថ្មីធំជាងមុន 2 ដង។

ថ្លៃលក់ /= 2 គឺជាបញ្ជាតម្រូវឲ្យធ្វើស្វ័យប្រមាណវិធីដើម្បីបង្កើតវត្ថុឈ្មោះ ថ្លៃលក់ ថ្មីតូចជាងមុន 2 ដង។

ថ្លៃលក់ //= 2 គឺជាបញ្ជាតម្រូវឲ្យធ្វើស្វ័យប្រមាណវិធីដើម្បីបង្កើតវត្ថុឈ្មោះ ថ្លៃលក់ ថ្មីដែលជាលទ្ធផលបានមកពីការធ្វើប្រមាណវិធីចែកបន្ថយរវាងវត្ថុឈ្មោះ ថ្លៃលក់ ចាស់និងលេខ 2 ។

ថ្លៃលក់ %= 2 គឺជាបញ្ជាតម្រូវឲ្យធ្វើស្វ័យប្រមាណវិធីដើម្បីបង្កើតវត្ថុឈ្មោះ ថ្លៃលក់ ថ្មីដែលជាលទ្ធផលបានមកពីការធ្វើប្រមាណវិធីចែកយកសំណល់រវាងវត្ថុឈ្មោះ ថ្លៃលក់ ចាស់និងលេខ 2 ។

ថ្លៃលក់ **= 2 គឺជាបញ្ជាតម្រូវឲ្យធ្វើស្វ័យប្រមាណវិធីដើម្បីបង្កើតវត្ថុឈ្មោះ ថ្លៃលក់ ថ្មីដែលជាលទ្ធផលបានមកពីការធ្វើប្រមាណវិធីស្វ័យគុនរវាងវត្ថុឈ្មោះ ថ្លៃលក់ ចាស់និងលេខ 2 ។

ដូចគ្នាដែរ យើងអាចយកកម្រងអក្សរផ្សេងៗមកធ្វើស្វ័យប្រមាណវិធីដូចខាងក្រោមនេះ៖

```
ឃ្លាដើម = "តក់ៗ"
ឃ្លាដើម += "ពេញបំពង់\t"
print(ឃ្លាដើម)
ឃ្លាដើម *= 3
print(ឃ្លាដើម)
```

ឃ្លាដើម += "*ពេញបំពង់*\t" គឺជាបញ្ជាតម្រូវឲ្យធ្វើស្វ័យប្រមាណវិធីដើម្បីបង្កើតកម្រងអក្សរឈ្មោះ ឃ្លាដើម ថ្មីមួយដែលជាឃ្លាវែងជាងមុន។

ឃ្លាដើម *= 3 គឺជាបញ្ជាតម្រូវឲ្យធ្វើស្វ័យប្រមាណវិធីដើម្បីបង្កើតកម្រងអក្សរឈ្មោះ ឃ្លាដើម ថ្មីមួយដែលជាឃ្លាវែងជាងមុន 3 ដង។

ដូចគ្នាដែរ យើងអាចយកកម្រងថេរផ្សេងៗមកធ្វើស្វ័យប្រមាណវិធីដូចខាងក្រោមនេះ៖

```
កម្រងចម្រុះ = (100, 1.5, "ប្រាក់ចំណេញ")
កម្រងចម្រុះ += (50, 4, True)
print(កម្រងចម្រុះ)
កម្រងចម្រុះ *= 3
print(កម្រងចម្រុះ)
```

កម្រងចម្រុះ += (50, 4, True) គឺជាបញ្ជាតម្រូវឲ្យធ្វើស្វ័យប្រមាណវិធីដើម្បីបង្កើតកម្រងថេរឈ្មោះ កម្រងចម្រុះ ថ្មីមានធាតុច្រើនជាងមុន។

កម្រងចម្រុះ *= 3 គឺជាបញ្ជាតម្រូវឲ្យធ្វើស្វ័យប្រមាណវិធីដើម្បីបង្កើតកម្រងថេរឈ្មោះ កម្រងចម្រុះ ថ្មីមានធាតុលើសមុន 3 ដង។

ដូចគ្នាដែរ យើងអាចយកកម្រងអថេរផ្សេងៗមកធ្វើស្វ័យប្រមាណវិធីដូចខាងក្រោមនេះ៖

```
កម្រងចម្រុះ = [100, 1.5, "ប្រាក់ចំណេញ"]
កម្រងចម្រុះ += [50, 4, True]
print(កម្រងចម្រុះ)
កម្រងចម្រុះ *= 3
print(កម្រងចម្រុះ)
```

ការយកកម្រងអថេរមកធ្វើស្វ័យប្រមាណវិធីតាមរបៀបដូចខាងលើនេះ គឺជាករណីពិសេស។ ពោលគឺមិនមែនជាការយកកម្រងអថេរចាស់មកដូរថ្មីនោះទេ ផ្ទុយទៅវិញ គឺជាការយកកម្រងអថេរចាស់ដដែលមកកែប្រែ ពីព្រោះកម្រងអថេរគឺជាវត្ថុអាចដោះដូរបាន។

កម្រងចម្រុះ += [50, 4, True] គឺជាបញ្ជាតម្រូវឲធ្វើស្វ័យប្រមាណវិធីដើម្បីបង្កើនកម្រងអថេរឈ្មោះ កម្រងចម្រុះ ដដែលឲ្យមានធាតុលើសមុន។

កម្រងចម្រុះ *= 3 គឺជាបញ្ជាតម្រូវឲ្យធ្វើស្វ័យប្រមាណវិធីដើម្បីបង្កើនកម្រងអថេរឈ្មោះ កម្រងចម្រុះ ដដែលឲ្យមានធាតុលើសមុន 3 ដង។

ក៏ប៉ុន្តែបើយើងចង់យកកម្រងអថេរមកធ្វើស្វ័យប្រមាណវិធីក្នុងគោលបំណងយកកម្រងអថេរចាស់មកដូរថ្មី មិនមែនយកមកកែប្រែ យើងត្រូវធ្វើតាមរបៀបដូចខាងក្រោមនេះ៖

```
កម្រងចម្រុះ = [100, 1.5, "ប្រាក់ចំណេញ"]
កម្រងចម្រុះ = កម្រងចម្រុះ + [50, 4, True]
print(កម្រងចម្រុះ)
```

```
កម្រងចម្រុះ = កម្រងចម្រុះ * 3
print(កម្រងចម្រុះ)
```

កម្រងចម្រុះ = កម្រងចម្រុះ + [50, 4, True] គឺជាបញ្ជាតម្រូវឲ្យធ្វើស្វ័យប្រមាណវិធីដើម្បីបង្កើតកម្រងអថេរឈ្មោះ កម្រងចម្រុះ ថ្មីមានធាតុលើសមុន។

កម្រងចម្រុះ = កម្រងចម្រុះ * 3 គឺជាបញ្ជាតម្រូវឲ្យធ្វើស្វ័យប្រមាណវិធីដើម្បីបង្កើតកម្រងអថេរឈ្មោះ កម្រងចម្រុះ ថ្មីមានធាតុលើសមុន 3 ដង។

ដូចគ្នាដែរ យើងអាចយកសំណុំផ្សេងៗមកធ្វើស្វ័យប្រមាណវិធីដូចខាងក្រោមនេះ៖

```
សំណុំចម្រុះ = {100, 1.5, "ប្រាក់ចំណេញ"}
សំណុំចម្រុះ -= {50, 4, True}
print(សំណុំចម្រុះ)
សំណុំចម្រុះ |= {50, 4, True}
print(សំណុំចម្រុះ)
សំណុំចម្រុះ &= {50, 4, True}
print(សំណុំចម្រុះ)
សំណុំចម្រុះ ^= {50, 4, True}
print(សំណុំចម្រុះ)
```

ចំពោះសំណុំក៏ដូចជាកម្រងអថេរដែរ ការធ្វើស្វ័យប្រមាណវិធីតាមរបៀបដូចខាងលើនេះ គឺជាការយកសំណុំចាស់មកកែប្រែ ពោលគឺមិនមែនជាការយកសំណុំចាស់មកដូរថ្មីឡើយ។

សំណុំចម្រុះ -= {50, 4, True} គឺជាបញ្ជាតម្រូវឲ្យធ្វើស្វ័យប្រមាណវិធីដើម្បីដកយកធាតុខ្លះដែលដូចទៅនឹងធាតុនៃសំណុំ {50, 4, True} ចេញពីសំណុំឈ្មោះ សំណុំចម្រុះ ។

សំណុំចម្រុះ |= {50, 4, True} គឺជាបញ្ជាតម្រូវឲ្យធ្វើស្ស័យប្រមាណវិធីដើម្បីបន្ថែមធាតុខ្លះដែលដូចទៅនឹងធាតុនៃសំណុំ {50, 4, True} ចូលទៅក្នុងសំណុំឈ្មោះ សំណុំចម្រុះ ។

សំណុំចម្រុះ &= {50, 4, True} គឺជាបញ្ជាតម្រូវឲ្យធ្វើស្ស័យប្រមាណវិធីដើម្បីដកយកធាតុខ្លះចេញពីសំណុំឈ្មោះ សំណុំចម្រុះ ដោយរក្សាទុកតែធាតុទាំងឡាយណាដែលដូចទៅនឹងធាតុនៃសំណុំ {50, 4, True} ។

សំណុំចម្រុះ ^= {50, 4, True} គឺជាបញ្ជាតម្រូវឲ្យធ្វើស្ស័យប្រមាណវិធីដើម្បីបន្ថែមធាតុខ្លះដែលមានតែនៅក្នុងសំណុំ {50, 4, True} ចូលទៅក្នុងសំណុំឈ្មោះ សំណុំចម្រុះ ។

ក៏ប៉ុន្តែ បើយើងចង់យកសំណុំមកធ្វើស្ស័យប្រមាណវិធីក្នុងគោលបំណងយកសំណុំចាស់មកដូរថ្មី យើងត្រូវធ្វើតាមរបៀបដូចខាងក្រោមនេះ៖

```
សំណុំចម្រុះ = {100, 1.5, "ប្រាក់ចំណេញ"}
សំណុំចម្រុះ = សំណុំចម្រុះ - {50, 4, True}
print(សំណុំចម្រុះ)
សំណុំចម្រុះ = សំណុំចម្រុះ | {50, 4, True}
print(សំណុំចម្រុះ)
សំណុំចម្រុះ = សំណុំចម្រុះ & {50, 4, True}
print(សំណុំចម្រុះ)
សំណុំចម្រុះ = សំណុំចម្រុះ ^ {50, 4, True}
print(សំណុំចម្រុះ)
```

សំណុំចម្រុះ = *សំណុំចម្រុះ* - {50, 4, True} គឺជាបញ្ជាតម្រូវឲ្យធ្វើស្ស័យប្រមាណវិធីដើម្បីបង្កើតសំណុំឈ្មោះ សំណុំចម្រុះ ថ្មីគ្មានធាតុដូចទៅនឹងសំណុំ {50, 4, True} ។

សំណុំចម្រុះ = *សំណុំចម្រុះ* | {50, 4, True} គឺជាបញ្ជាតម្រូវឲ្យធ្វើស្វ័យប្រមាណវិធីដើម្បីបង្កើតសំណុះឈ្មោះ សំណុំចម្រុះ ថ្មីមានធាតុមួយចំនួនដូចទៅនឹងសំណុំ {50, 4, True} ។

សំណុំចម្រុះ = *សំណុំចម្រុះ* & {50, 4, True} គឺជាបញ្ជាតម្រូវឲ្យធ្វើស្វ័យប្រមាណវិធីដើម្បីបង្កើតសំណុំឈ្មោះ សំណុំចម្រុះ ថ្មីមានធាតុដូចទៅនឹងសំណុំ {50, 4, True} ។

សំណុំចម្រុះ = *សំណុំចម្រុះ* ^ {50, 4, True} គឺជាបញ្ជាតម្រូវឲ្យធ្វើស្វ័យប្រមាណវិធីដើម្បីបង្កើតសំណុំឈ្មោះ សំណុំចម្រុះ ថ្មីមានធាតុជាការប្រជុំធាតុខុសគ្នានៃសំណុំឈ្មោះ សំណុំចម្រុះ និងសំណុំ {50, 4, True} ។

កន្សោមប្រមាណវិធី

កន្សោមប្រមាណវិធីក៏ត្រូវចាត់ទុកថាជាបញ្ជាមួយដែរ គឺជាបញ្ជាតម្រូវឲ្យមានការធ្វើប្រមាណវិធីផ្សេងៗ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
បណ្តោយ = 150
ទទឹង = 23
កំពស់ = 15
មាឌ = បណ្តោយ * ទទឹង * កំពស់
print(មាឌ)
```

*បណ្តោយ * ទទឹង * កំពស់* គឺជាកន្សោមប្រមាណវិធីដែលជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីគុនរវាងវត្ថុចំនួនបីគឺ បណ្តោយ ទទឹង និង កំពស់ ។

បញ្ជា if

if គឺជាបញ្ជាតម្រូវឲ្យអនុវត្តបញ្ជាមួយចំនួនទៀត ក្នុងករណីកន្សោមប្រមាណវិធីមួយផ្តល់លទ្ធផលជាតក្កវត្ត True ឬសមមូលនឹង True ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
ថ្លៃលក់ = 1000
ថ្លៃទិញ = 900
if ថ្លៃលក់ > ថ្លៃទិញ :
   ប្រាក់ចំណេញ = ថ្លៃលក់ - ថ្លៃទិញ
   print(ប្រាក់ចំណេញ)
```

if *ថ្លៃលក់* > *ថ្លៃទិញ* : គឺជាការប្រើបញ្ជា if តម្រូវឲ្យអនុវត្តបញ្ជាពីរទៀតនៅខាងក្រោមនោះក្នុងករណីកន្សោមប្រមាណវិធី *ថ្លៃលក់* > *ថ្លៃទិញ* ផ្តល់លទ្ធផលជាតក្កវត្ត True ។ បើពុំនោះសោតទេ បញ្ជាទាំងពីរនោះនឹងត្រូវរំលងចោល។ ដោយ ថ្លៃលក់ ជាលេខ 1000 និង ថ្លៃទិញ ជាលេខ 900 ដូចនេះកន្សោមប្រមាណវិធី *ថ្លៃលក់* > *ថ្លៃទិញ* ផ្តល់លទ្ធផលជាតក្កវត្ត True ដែលនាំឲ្យបញ្ជាទាំងពីរនៅខាងក្រោមនោះត្រូវបានយកទៅអនុវត្ត។

សញ្ញាចុចពីរ (:) នៅខាងចុងកន្សោមប្រមាណវិធីគឺជាសញ្ញាកំណត់ពីចំនុចចាប់ផ្តើមនៃបញ្ជាទាំងឡាយណាដែលត្រូវយកទៅអនុវត្តក្នុងករណីកន្សោមប្រមាណវិធីផ្តល់លទ្ធផលជាតក្កវត្ត True ។ បញ្ជាទាំងនោះត្រូវតែសរសេរចូលបន្ទាត់ ដើម្បីបញ្ជាក់ប្រាប់ថាវាជាបញ្ជានៅក្នុងបញ្ជា if ។ ការសរសេរចូលបន្ទាត់អាចត្រូវដកឃ្លាចំនួនប៉ុន្មានក៏បានដែរ តែគេនិយមចូលចិត្តសរសេរចូលបន្ទាត់ដោយដកឃ្លាចំនួនបួន។ បញ្ជាដែលត្រូវសរសេរចូលបន្ទាត់ទាំងនោះគឺជា *ក្រុមបញ្ជា* (block) ស្ថិតនៅក្នុងបញ្ជា if ។ ដូចនេះក្រុមបញ្ជាស្ថិតនៅក្នុងបញ្ជា if ត្រូវយកទៅអនុវត្តតែក្នុងករណីកន្សោមប្រមាណវិធីជាប់នឹងបញ្ជា if នោះផ្តល់លទ្ធផលជាតក្កវត្ត True ឬ

សមមូលនឹង True តែប៉ុណ្ណោះ បើពុំនោះសោតទេ ក្រុមបញ្ជានោះនឹងត្រូវរំលងចោល។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
ថ្លៃលក់ = 900
ថ្លៃទិញ = 900
if ថ្លៃលក់ > ថ្លៃទិញ :
   ប្រាក់ចំណេញ = ថ្លៃលក់ - ថ្លៃទិញ
   print(ប្រាក់ចំណេញ)
```

if ថ្លៃលក់ > ថ្លៃទិញ : គឺជាការប្រើបញ្ជា if តម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងនោះក្នុងករណីកន្សោមប្រមាណវិធី ថ្លៃលក់ > ថ្លៃទិញ ផ្តល់លទ្ធផលជាតក្កវត្ត True បើពុំនោះសោតទេ ក្រុមបញ្ជានោះនឹងត្រូវរំលងចោល។ ដោយ ថ្លៃលក់ ជាលេខ 900 និង ថ្លៃទិញ ជាលេខ 900 ដូចនេះកន្សោមប្រមាណវិធី ថ្លៃលក់ > ថ្លៃទិញ ផ្តល់លទ្ធផលជាតក្កវត្ត False ដែលនាំឲ្យក្រុមបញ្ជានៅក្នុងបញ្ជា if ត្រូវបានរំលងចោល។

បញ្ជា *if/else*

if/else គឺជាបញ្ជាតម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងបញ្ជា if និងតម្រូវឲ្យរំលងចោលក្រុមបញ្ជានៅក្នុងបញ្ជា else ក្នុងករណីកន្សោមប្រមាណវិធីមួយផ្តល់លទ្ធផលជាតក្កវត្ត True ឬសមមូលនឹង True ។ តែបើកន្សោមប្រមាណវិធីនោះផ្តល់លទ្ធផលជាតក្កវត្ត False ឬសមមូលនឹង False ក្រុមបញ្ជានៅក្នុងបញ្ជា if នឹងត្រូវរំលងចោល ហើយក្រុមបញ្ជានៅក្នុងបញ្ជា else នឹងត្រូវយកទៅអនុវត្ត។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
ថ្លៃលក់ = 1000
```

```
ថ្លៃទិញ = 900
if ថ្លៃលក់ > ថ្លៃទិញ :
    ប្រាក់ចំណេញ = ថ្លៃលក់ - ថ្លៃទិញ
    print(ប្រាក់ចំណេញ)
else :
    print("ថ្លៃលក់មិនច្រើនជាងថ្លៃទិញទេ")
    print("រកស៊ីមិនចំណេញ")
```

if ថ្លៃលក់ > ថ្លៃទិញ : គឺជាការប្រើបញ្ជា if តម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងនោះនិងរំលងចោលក្រុមបញ្ជានៅក្នុងនៅក្នុងបញ្ជា else ក្នុងករណីកន្សោមប្រមាណវិធី ថ្លៃលក់ > ថ្លៃទិញ ផ្តល់លទ្ធផលជាតក្កវត្ត True ។

else : គឺជាការប្រើបញ្ជា else តម្រូវឲ្យរំលងចោលក្រុមបញ្ជានៅក្នុងបញ្ជា if និងអនុវត្តក្រុមបញ្ជានៅក្នុងបញ្ជា else នេះក្នុងករណីកន្សោមប្រមាណវិធី ថ្លៃលក់ > ថ្លៃទិញ ផ្តល់លទ្ធផលជាតក្កវត្ត False ។

ដោយវត្ថុឈ្មោះ ថ្លៃលក់ ជាលេខ 1000 និងវត្ថុឈ្មោះ ថ្លៃទិញ ជាលេខ 900 ដូចនេះកន្សោមប្រមាណវិធី ថ្លៃលក់ > ថ្លៃទិញ ផ្តល់លទ្ធផលជាតក្កវត្ត True ដែលនាំឲ្យក្រុមបញ្ជានៅក្នុងបញ្ជា if ត្រូវបានយកទៅអនុវត្ត និងក្រុមបញ្ជានៅក្នុងបញ្ជា else ត្រូវបានរំលងចោល។

បញ្ជា if/elif/else

if/elif/else គឺជាបញ្ជាតម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងបញ្ជា if ឬបញ្ជា elif ក្នុងករណីកន្សោមប្រមាណវិធីជាប់នឹងបញ្ជា if ឬ elif នោះផ្តល់លទ្ធផលជាតក្កវត្ត True ឬសមមូលនឹង True ។

តែបើគ្មានកន្សោមប្រមាណវិធីណាមួយផ្តល់លទ្ធផលជាតក្កវត្ត True ឬសមមូលនឹង True ទេ ក្រុមបញ្ជានៅក្នុងបញ្ជា else នឹងត្រូវយកទៅអនុវត្ត។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
a = 15
b = 15
if a < b :
    print("a តូចជាង b")
    print("ក្រុមបញ្ជានៅក្នុងបញ្ជា if ត្រូវបានយកទៅអនុវត្ត។")
elif a == b :
    print("a ស្មើនឹង b")
    print("ក្រុមបញ្ជានៅក្នុងបញ្ជា elif ទីមួយត្រូវបានយកទៅអនុវត្ត។")
elif a < b < 100 :
    print("a < b < 100")
    print("ក្រុមបញ្ជានៅក្នុងបញ្ជា elif ទីពីរត្រូវបានយកទៅអនុវត្ត។")
else :
    print("គ្មានកន្សោមប្រមាណវិធីណាមួយផ្តល់លទ្ធផលជាតក្កវត្ត True ទេ។")
    print("ក្រុមបញ្ជានៅក្នុងបញ្ជា else ត្រូវបានយកទៅអនុវត្ត។")
```

if a < b : គឺជាបញ្ជាតម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងនោះ និងរំលងចោលក្រុមបញ្ជានៅក្នុងបញ្ជាផ្សេងៗទៀត ក្នុងករណីកន្សោមប្រមាណវិធី a < b ផ្តល់លទ្ធផលជាតក្កវត្ត True ។

if a == b : គឺជាបញ្ជាតម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងនោះ និងរំលងចោលក្រុមបញ្ជានៅក្នុងបញ្ជាផ្សេងៗទៀត ក្នុងករណីកន្សោមប្រមាណវិធី a == b ផ្តល់លទ្ធផលជាតក្កវត្ត True ។

if a < b < 100 : គឺជាបញ្ជាតម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងនោះ និងរំលងចោលក្រុមបញ្ជានៅក្នុងបញ្ជាផ្សេងៗទៀត ក្នុងករណីកន្សោមប្រមាណវិធី a < b < 100 ផ្តល់លទ្ធផលជាតក្កវត្ត True ។

else : គឺជាបញ្ជាតម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងនោះ និងរំលងចោលក្រុមបញ្ជានៅក្នុងបញ្ជាផ្សេងៗទៀត ក្នុងករណីគ្មានកន្សោមប្រមាណវិធីណាមួយផ្តល់លទ្ធផលជាតក្កវត្ត True ។

ដោយ a ជាលេខ 15 និង b ក៏ជាលេខ 15 ដែរ ដូចនេះកន្សោមប្រមាណវិធី a == b ផ្តល់លទ្ធផលជាតក្កវត្ត True ដែលនាំឲ្យក្រុមបញ្ជានៅក្នុងបញ្ជា elif ទីពីរត្រូវបានយកទៅអនុវត្ត និងក្រុមបញ្ជានៅក្នុងបញ្ជាផ្សេងៗទៀតត្រូវរំលងចោលទាំងអស់។

យើងត្រូវធ្វើការកត់សំគាល់ថា យើងអាចប្រើបញ្ជា elif មានចំនួនប៉ុន្មានក៏បានដែរនៅក្នុងបញ្ជា if/elif/else អាស្រ័យទៅតាមសេចក្តីត្រូវការនៃការដោះស្រាយបញ្ហា។

កន្សោមប្រមាណវិធីមានជម្រើស

កន្សោមប្រមាណវិធីមានជម្រើសគឺជាកន្សោមប្រមាណវិធីដែលមានទម្រង់ស្រដៀងនឹងបញ្ជា if/else ដែរ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
ថ្លៃលក់ = 1000
ថ្លៃទិញ = 900
លទ្ធផល = "ចំណេញ" if (ថ្លៃលក់ - ថ្លៃទិញ > 0) else "ខាត"
print(លទ្ធផល)
```

លទ្ធផល = "ចំណេញ" if (ថ្លៃលក់ - ថ្លៃទិញ > 0) else "ខាត" គឺជាកន្សោមប្រមាណវិធីមានជម្រើសដែលផ្តល់លទ្ធផលជាកម្រងអក្សរ "ចំណេញ" ក្នុងករណីកន្សោមប្រមាណវិធី (ថ្លៃលក់ - ថ្លៃទិញ > 0) ផ្តល់លទ្ធផលជាតក្កវត្ត True បើពុំនោះសោតទេ កន្សោមប្រមណវិធីមានជម្រើសនោះនឹងផ្តល់លទ្ធផលជាកម្រងអក្សរ "ខាត" ។

ដោយវត្ថុឈ្មោះ ថ្លៃលក់ ជាលេខ 1000 និងវត្ថុឈ្មោះ ថ្លៃទិញ ជាលេខ 900 ដូចនេះកន្សោមប្រមាណវិធី (ថ្លៃលក់ - ថ្លៃទិញ > 0) ផ្តល់លទ្ធផលជាតក្កវត្ថុ True ដែលនាំឲ្យលទ្ធផលបានមកពីកន្សោមប្រមាណវិធីមានជម្រើសខាងលើនេះគឺជាកម្រងអក្សរ "ចំណេញ" ។

ក្រៅពីកម្រងអក្សរ វត្ថុដែលជាលទ្ធផលបានមកពីកន្សោមប្រមាណវិធីមានជម្រើសអាចជាវត្ថុប្រភេទណាក៏បានដែរ។

បញ្ជា while

while គឺជាបញ្ជាតម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងនោះសារចុះសារឡើងគ្មានឈប់ដរាបណាកន្សោមប្រមាណវិធីមួយនៅតែផ្តល់លទ្ធផលជាតក្កវត្ថុ True ឬសមមូលនឹង True ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
a = 0
while a < 10 :
    print("ក្រុមបញ្ជានៅក្នុងបញ្ជា while ត្រូវបានយកទៅអនុវត្តជាលើកទី", a + 1)
    print("វត្ថុឈ្មោះ a គឺជាលេខ", a)
    a += 1
    print()
```

while a < 10 : គឺជាបញ្ជាតម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងនោះសារចុះសារឡើងគ្មានឈប់ដរាបណាកន្សោមប្រមាណវិធី a < 10 នៅតែផ្តល់លទ្ធផលជាតក្កវត្ថុ True ។

នៅពេលដែលកម្មវិធីខាងលើចាប់ផ្តើមដំណើរការ វត្ថុឈ្មោះ a គឺជាលេខ 0 ។ ដូចនេះកន្សោមប្រមណវិធី a < 10 ផ្តល់លទ្ធផលជាតក្កវត្ថុ True ដែលជាប្រការធ្វើឲ្យក្រុមបញ្ជានៅក្នុងបញ្ជា while ត្រូវយកទៅអនុវត្តជាលើកទីមួយ។ កត្តានេះធ្វើឲ្យកម្រងអក្សរពីរត្រូវបានសរសេរនៅ

លើបង្អួចបឋម និងវត្ថុឈ្មោះ a ក្លាយទៅជាវត្ថុថ្មីធំជាងមុន 1 ។ ហើយក្រោយពីក្រុមបញ្ជានៅក្នុងបញ្ជា while ត្រូវបានយកទៅអនុវត្តបានចប់សព្វគ្រប់រួចហើយ កន្សោមប្រមាណវិធី a < 10 ត្រូវយកមកធ្វើការគណនាជាថ្មីម្តងទៀត ហើយបើកន្សោមប្រមាណវិធីនោះនៅតែផ្តល់លទ្ធផលជាតក្កវត្ថុ True ក្រុមបញ្ជានៅក្នុងបញ្ជា while នោះនឹងត្រូវយកទៅអនុវត្តជាថ្មីម្តងទៀត។ ទង្វើរបៀបនេះត្រូវប្រព្រឹត្តទៅជាដដែលៗរហូតដល់កន្សោមប្រមាណវិធី a < 10 លែងផ្តល់លទ្ធផលជាតក្កវត្ថុ True គឺនៅពេលដែលវត្ថុឈ្មោះ a ក្លាយទៅជាលេខ 10 ។ ដូចនេះនៅពេលដែលក្រុមបញ្ជានៅក្នុងបញ្ជា while ត្រូវបានយកទៅអនុវត្តចំនួន 10 ដង វត្ថុឈ្មោះ a ក្លាយជាលេខ 10 ពីព្រោះរាល់លើកដែលបញ្ជា a += 1 ត្រូវបានយកទៅអនុវត្ត វត្ថុឈ្មោះ a ក្លាយទៅជាលេខធំជាងមុន 1 រហូត។ ហើយនៅពេលដែលវត្ថុឈ្មោះ a ក្លាយទៅជាលេខ 10 កន្សោមប្រមាណវិធី a < 10 ផ្តល់លទ្ធផលជាតក្កវត្ថុ False ដែលជាប្រការធ្វើឲ្យការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងបញ្ជា while ត្រូវបញ្ចប់។

សរុបមក ដើម្បីឲ្យការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងបញ្ជា while អាចត្រូវបញ្ចប់បាន លុះត្រាណាតែកន្សោមប្រមាណវិធីជាប់នឹងបញ្ជា while នោះលែងផ្តល់លទ្ធផលជាតក្កវត្ថុ True ឬសមមូលនឹង True នៅពេលណាមួយ បើពុំនោះសោតទេ ការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងបញ្ជា while នឹងប្រព្រឹត្តទៅជាប់រហូតគ្មានឈប់។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
a = 0
while a < 10 :
    print("ក្រុមបញ្ជានៅក្នុងបញ្ជា while ត្រូវបានយកទៅអនុវត្តជាលើកទី", a + 1)
    print("វត្ថុឈ្មោះ a គឺជាលេខ", a)
    print()
```

បើយើងដំណើរការកម្មវិធីខាងលើនេះ វានឹងដំណើរការជាប់រហូតគ្មានឈប់។ ដំណើរការគ្មានឈប់នេះ ភាសាអង់គ្លេសហៅថា infinite loop ដែលយើងអាចបកប្រែជាភាសាខ្មែរថា *វដ្តកម្មជានិរន្តន៍* ។

មូលហេតុដែលនាំឲ្យកម្មវិធីខាងលើនេះមានដំណើរការជាវដ្តកម្មជានិរន្តន៍ គឺថាគ្រប់ពេលដែលក្រុមបញ្ជានៅក្នុងបញ្ជា while ត្រូវយកទៅអនុវត្ត វត្ថុឈ្មោះ a នៅរក្សាតម្លៃដដែល ដូចនេះកន្សោមប្រមាណវិធី a < 10 នៅតែផ្តល់លទ្ធផលជាតក្កវត្ត True ជានិច្ច។ ប្រការនេះធ្វើឲ្យក្រុមបញ្ជានៅក្នុងបញ្ជា while ត្រូវយកទៅអនុវត្តជារៀងរហូត ព្រោះការអនុវត្តន៍ក្រុមបញ្ជានោះអាចត្រូវឈប់បានលុះត្រាណាតែកន្សោមប្រមាណវិធី a < 10 ផ្តល់លទ្ធផលជាតក្កវត្ត False តែប៉ុណ្ណោះ។

បញ្ជា for

for គឺជាបញ្ជាតម្រូវឲ្យពិនិត្យមើលធាតុទាំងអស់នៅក្នុងសមាសវត្ថុណាមួយបណ្តើរនិងតម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងបញ្ជា for នោះបណ្តើរ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
កម្រងចម្រុះ = [100, 1.5, "កែវ កុសល", True]
លេខរៀង = 0
for វត្ថុ in កម្រងចម្រុះ :
   print("ជាតុនៅក្នុងកម្រងអថេរឈ្មោះ កម្រងចម្រុះ មានលេខរៀង", លេខរៀង, "គឺ", វត្ថុ)
   លេខរៀង += 1
```

for វត្ថុ in កម្រងចម្រុះ : គឺជាបញ្ជាតម្រូវឲ្យពិនិត្យមើលគ្រប់ធាតុទាំងអស់នៅក្នុងកម្រងអថេរឈ្មោះ កម្រងចម្រុះ បណ្តើរនិងតម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងបញ្ជា for នោះបណ្តើរ។

ការពិនិត្យមើលធាតុនៅក្នុងកម្រងអថេរឈ្មោះ កម្រងចម្រុះ គឺត្រូវធ្វើឡើងដោយភ្ជាប់ឈ្មោះ វត្ថុ ទៅនឹងគ្រប់ធាតុនៅក្នុងកម្រងអថេរនោះម្តងមួយៗពីឆ្វេងទៅស្តាំ។ ហើយគ្រប់ការភ្ជាប់ឈ្មោះ វត្ថុ ទៅនឹងធាតុណាមួយ ក្រុមបញ្ជានៅក្នុងបញ្ជា for ត្រូវយកទៅអនុវត្តម្តងដែរ។ ដោយកម្រងអថេរឈ្មោះ កម្រងចម្រុះ មានធាតុចំនួន 4 ដូចនេះការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងកបញ្ជា for ក៏ត្រូវធ្វើឡើងចំនួន 4 ដងដែរ។

ក្នុងករណីសមាសវត្ថុជាវចនានុក្រម ការប្រើបញ្ជា for ដើម្បីពិនិត្យមើលធាតុនៃសមាសវត្ថុនោះ គឺជាការប្រើបញ្ជា for ដើម្បីពិនិត្យមើលកូនសោរនៅក្នុងវចនានុក្រមនោះ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
ប្រវត្តិរូប = {"ឈ្មោះ":"កុសល", "ត្រកូល":"កែវ", "អាយុ":35, "ទីលំនៅ":"កម្ពុជា"}
for វត្ថុ in ប្រវត្តិរូប :
    print("តម្លៃជាប់នឹងកូនសោរ", វត្ថុ, "គឺ", ប្រវត្តិរូប[វត្ថុ])
```

for វត្ថុ in ប្រវត្តិរូប : គឺជាការប្រើបញ្ជា for តម្រូវឲ្យពិនិត្យមើលកូនសោរនៅក្នុងវចនានុក្រមឈ្មោះ ប្រវត្តិរូប និងតម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងបញ្ជា for នោះព្រមពេលជាមួយគ្នា។

បញ្ជា break

break គឺជាបញ្ជាតម្រូវឲ្យបញ្ចប់ជាបន្ទាន់នូវការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងបញ្ជា while ឬ for ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
ប្រវត្តិរូប = {"ឈ្មោះ":"កុសល", "ត្រកូល":"កែវ", "អាយុ":35, "ទីលំនៅ":"កម្ពុជា"}
b = 0
for វត្ថុ in ប្រវត្តិរូប :
    b += 1
    if b == 3 :
```

```
        break
    print("តម្លៃជាប់នឹងកូនសោរ", វត្ថុ, "គឺ", ប្រវត្តិរូប)
while True :
    b += 1
    print("b គឺជាលេខ", b)
    if b == 10 :
        break
```

break គឺជាបញ្ជាតម្រូវឲ្យបញ្ចប់ការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងបញ្ជា for និងបញ្ឈប់ការពិនិត្យមើលកូនសោរនៅក្នុងវចនានុក្រមឈ្មោះ ប្រវត្តិរូប ជាបន្ទាន់ក្នុងករណី b ជាលេខ 3 ។ បានន័យថានៅពេលដែល b ជាលេខ 3 ការពិនិត្យមើលកូនសោរនៅក្នុងវចនានុក្រមឈ្មោះ ប្រវត្តិរូប និងការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងបញ្ជា for ត្រូវបញ្ឈប់ជាបន្ទាន់ ហើយបញ្ជាបន្ទាប់ពីក្រុមបញ្ជានោះនឹងត្រូវយកទៅអនុវត្តជាបន្ទាន់ដែរ។

break គឺជាបញ្ជាតម្រូវឲ្យបញ្ចប់ជាបន្ទាន់ការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងបញ្ជា while ក្នុងករណី b ជាលេខ 10 ។

បញ្ជា continue

continue គឺជាបញ្ជាតម្រូវឲ្យអនុវត្តសារឡើងវិញជាបន្ទាន់នូវក្រុមបញ្ជានៅក្នុងបញ្ជា while ឬ បញ្ជា for ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
កម្រងចម្រុះ = [100, 1.5, "នាមត្រកូល", True, 200, "គោត្តនាម"]
b = 0
for វត្ថុ in កម្រងចម្រុះ :
    b += 1
    if b == 4 :
        continue
    print("ធាតុមានលេខរៀង", b -1, "នៅក្នុងកម្រងអថេរឈ្មោះ កម្រងចម្រុះ គឺ", វត្ថុ)
while b < 10 :
    b += 1
```

```
if b == 7 :
    continue
print("b គឺជាលេខ", b)
```

continue គឺជាបញ្ជាតម្រូវឲ្យអនុវត្តសារឡើងវិញជាបន្ទាន់នូវក្រុមបញ្ជានៅក្នុងបញ្ជា for ក្នុងករណី b ជាលេខ 4 ។ ដូចនេះក្នុងករណី b ជាលេខ 4 បញ្ជាបន្ទាប់ពីបញ្ជា continue នៅក្នុងក្រុមបញ្ជាជាមួយគ្នានឹងត្រូវទុកចោល ព្រោះក្រុមបញ្ជានៅក្នុងបញ្ជា for ត្រូវយកទៅអនុវត្តសារជាថ្មីឡើងវិញជាបន្ទាន់ចាប់ពីដើមដំបូងមក។

continue គឺជាបញ្ជាតម្រូវឲ្យអនុវត្តសារឡើងវិញជាបន្ទាន់នូវក្រុមបញ្ជានៅក្នុងបញ្ជា while ក្នុងករណី b ជាលេខ 7 ។ ដូចនេះក្នុងករណី b ជាលេខ 7 បញ្ជាបន្ទាប់ពីបញ្ជា continue នៅក្នុងក្រុមបញ្ជានៅក្នុងបញ្ជា while នឹងត្រូវទុកចោលព្រោះក្រុមបញ្ជានោះត្រូវយកទៅអនុវត្តសារដើមឡើងវិញជាបន្ទាន់។

បញ្ជា pass

pass គឺជាបញ្ជាតម្រូវឲ្យរំលងចោលដោយមិនធ្វើអ្វីទាំងអស់។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
print("កម្មវិធីត្រូវចាប់ផ្តើមនៅកន្លែងនេះ")
if True :
    pass
print("កម្មវិធីត្រូវចប់ត្រឹមនេះ")
```

pass គឺជាបញ្ជាតម្រូវឲ្យរំលងចោលដោយមិនធ្វើអ្វីទាំងអស់។

នៅក្នុងបញ្ជាមួយចំនួនដូចជាបញ្ជា if ជាដើម បើយើងមិនចង់ធ្វើអ្វីសោះនោះ យើងមិនអាចទុកឲ្យនៅទទេបានឡើយ យើងត្រូវប្រើបញ្ជា pass នេះតម្រូវឲ្យរំលងចោលដោយមិនធ្វើអ្វីទាំងអស់។ បើយើងមិនសរសេរអ្វីសោះនោះ កំហុសនឹងកើតមានឡើង។

បញ្ជា while/else

while/else គឺជាបញ្ជាតម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងបញ្ជា while សារចុះសារឡើងគ្មានឈប់ ដរាបណាកន្សោមប្រមាណវិធីមួយនៅតែផ្តល់លទ្ធផលជាតក្កវត្ត True ឬសមមូលនឹង True ។ ហើយបើការអនុវត្តន៍ក្រុមបញ្ជានោះត្រូវបានបញ្ចប់ទៅដោយគ្មានបានជួបប្រទះនឹងបញ្ជា break ក្រុមបញ្ជានៅក្នុងបញ្ជា else នឹងត្រូវយកទៅអនុវត្ត។ ផ្ទុយទៅវិញ បើការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងបញ្ជា while បានជួបប្រទះនឹងបញ្ជា break ក្រុមបញ្ជានៅក្នុងបញ្ជា else នឹងត្រូវរំលងចោល។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
a = 0
while a < 10 :
    print("a ជាលេខ", a)
    a += 1
    if a == 11 :
        break
else :
    print("ការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងបញ្ជា while គ្មានបានជួបប្រទះនឹងបញ្ជា break ទេ។")
```

while a < 10 : គឺជាការប្រើបញ្ជា while តម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងនោះរហូតទាល់តែកន្សោមប្រមាណវិធី a < 10 ផ្តល់លទ្ធផលជាតក្កវត្ត False ឬជួបប្រទះនឹងបញ្ជា break ។

else : គឺជាការប្រើបញ្ជា else តម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងនោះ ក្នុងករណីការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងបញ្ជា while ត្រូវបានបញ្ចប់ទៅដោយសម្រួល ដោយមិនបានជួបប្រទះនឹងបញ្ជា break ។

ដោយការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងបញ្ជា while ត្រូវបានបញ្ចប់ទៅតាមសម្រួល ដោយគ្មានបានជួបប្រទះនឹងបញ្ជា break ដូចនេះក្រុមបញ្ជានៅក្នុងបញ្ជា else ក៏ត្រូវបានយកទៅអនុវត្តដែរ។

លើសពីនេះទៀត ក្រុមបញ្ជានៅក្នុងបញ្ជា else នឹងត្រូវយកទៅអនុវត្តដែរបើសិនជាក្រុមបញ្ជានៅក្នុងបញ្ជា while ត្រូវបានរំលងចោលមកពីកន្សោមប្រមាណវិធីនៅជាប់នឹងបញ្ជាwhile នោះផ្តល់លទ្ធផលជាតក្កវត្ត False ឬសមមូលនឹង False តាំងពីដំបូងដែមកម្លេះ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
a = 0
while a > 10 :
    print("a ជាលេខ", a)
    a += 1
    if a == 11 :
        break
else :
    print("ការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងបញ្ជា while គ្មានបានជួបប្រទះនឹងបញ្ជា break ទេ។")
```

while a > 10 : គឺជាការប្រើបញ្ជា while តម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងនោះរហូតទាល់តែកន្សោមប្រមាណវិធី a > 10 ផ្តល់លទ្ធផលជាតក្កវត្ត False ឬជួបប្រទះនឹងបញ្ជា break ។

else : គឺជាការប្រើបញ្ជា else តម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងនោះ ក្នុងករណីការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងបញ្ជា while ត្រូវបានបញ្ចប់ទៅដោយសម្រួលដោយមិនបានជួបប្រទះនឹងបញ្ជា break ឬក្នុងករណីក្រុមបញ្ជានៅក្នុងបញ្ជា while ត្រូវរំលងចោលដោយសារកន្សោមប្រមាណវិធី a > 10 ផ្តល់លទ្ធផលជាតក្កវត្ត False តាំងពីដំបូងដែមកម្លេះ។

ដោយវត្ថុឈ្មោះ a ជាលេខ 0 ដូចនេះកន្សោមប្រមាណវិធី a > 10 ផ្តល់លទ្ធផលជាតក្កវត្ថុ False ដែលនាំឲ្យក្រុមបញ្ជានៅក្នុងបញ្ជា while ត្រូវរំលងចោល និងក្រុមបញ្ជានៅក្នុងបញ្ជា else ត្រូវយកទៅអនុវត្ត។

បញ្ជា for/else

for/else គឺជាបញ្ជាតម្រូវឲ្យពិនិត្យមើលធាតុនៅក្នុងសមាសវត្ថុណាមួយបណ្តើរនិងតម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងបញ្ជា for នោះបណ្តើរ។ ហើយបើការអនុវត្តន៍ក្រុមបញ្ជានោះគ្មានបានជួបប្រទះនឹងបញ្ជា break ទេ ក្រុមបញ្ជានៅក្នុងបញ្ជា else ក៏នឹងត្រូវយកទៅអនុវត្តដែរ។ ផ្ទុយទៅវិញ បើការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងបញ្ជា for បានជួបប្រទះនឹងបញ្ជា break ក្រុមបញ្ជានៅក្នុងបញ្ជា else នឹងត្រូវរំលងចោល។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
កម្រងចម្រុះ = [100, 1.5, "នាមត្រកូល", True]
b = 0
for វត្ថុ in កម្រងចម្រុះ :
    print("ធាតុនៃកម្រងអថេរឈ្មោះ កម្រងចម្រុះ មានលេខរៀង", b, "គឺ", វត្ថុ)
    b += 1
    if វត្ថុ == 11 :
        break
else :
    print("ការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងបញ្ជា for គ្មានបានជួបប្រទះនឹងបញ្ជា break ទេ។")
```

for វត្ថុ in កម្រងចម្រុះ : គឺជាការប្រើបញ្ជា for តម្រូវឲ្យពនិត្យមើលធាតុនៃកម្រងអថេរឈ្មោះ កម្រងចម្រុះ បណ្តើរនិងតម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងបញ្ជា for នោះបណ្តើរ។

else : គឺជាការប្រើបញ្ជា else តម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងនោះ ក្នុងករណីការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងបញ្ជា for គ្មានបានជួបប្រទះនឹងបញ្ជា break ទេ។

ដោយការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងបញ្ជា for គ្មានបានជួបប្រទះនឹងបញ្ជា break ដូចនេះក្រុមបញ្ជានៅក្នុងបញ្ជា else ត្រូវបានយកទៅអនុវត្ត។

វដ្តកម្ម ក្នុងវដ្តកម្ម

while និង for គឺជាបញ្ជាដែលតម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងនោះសារចុះសារឡើង វិលចុះវិលឡើងជាច្រើនលើកច្រើនសារ គឺប្រៀបបាននឹងកងចក្រមួយដែលវិលគ្មានឈប់។ សកម្មភាពវិលជាប់នេះហៅថា *វដ្តកម្ម* (loop) ។

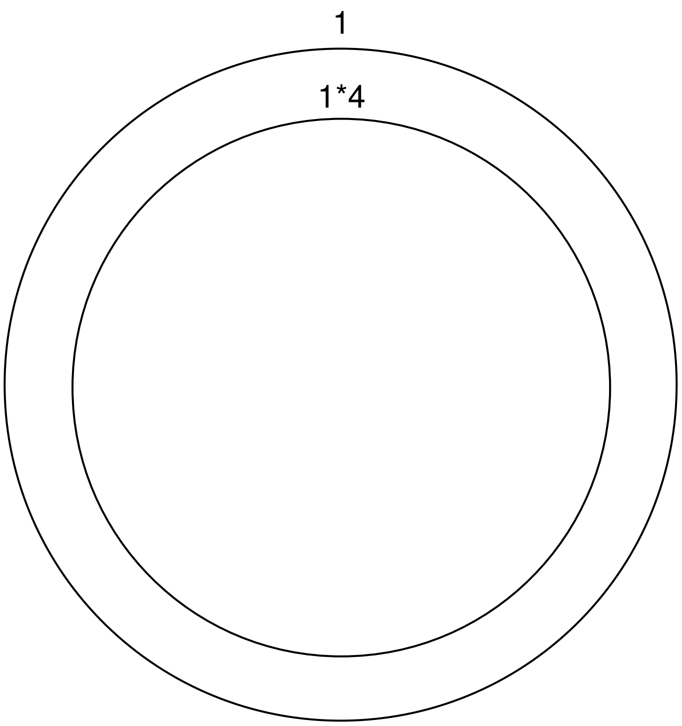
ក្រៅពីការបង្កើតវដ្តកម្មតែមួយដាច់តែឯង យើងអាចបង្កើតវដ្តកម្មមួយនៅក្នុងវដ្តកម្មមួយទៀតដោយសរសេរកម្មវិធីដូចខាងក្រោមនេះ៖

```
កម្រងចំនួនគត់ = [10, 20, 30, 40, 50]
កម្រងចំនួនពិត = [1.5, 2.33, 3.17, 4.32]
លេខរៀង = 0
for x in កម្រងចំនួនគត់ :
  print("ធាតុនៃកម្រងអថេរឈ្មោះ កម្រងចំនួនគត់ មានលេខរៀង", លេខរៀង, "គឺ", x)
  លេខ = 0
  for y in កម្រងចំនួនពិត :
    print("\t ធាតុនៃកម្រងអថេរឈ្មោះ កម្រងចំនួនពិត មានលេខរៀង", លេខ, "គឺ", y)
    លេខ += 1
  លេខរៀង += 1
```

for x in *កម្រងចំនួនគត់* : គឺជាការប្រើបញ្ជា for ដើម្បីបង្កើតវដ្តកម្មមួយ។

for y in *កម្រងចំនួនពិត* : គឺជាការប្រើបញ្ជា for មួយទៀតដើម្បីបង្កើតវដ្តកម្មមួយទៀតនៅខាងក្នុងវដ្តកម្មខាងដើម។

ក្នុងករណីវដ្តកម្មមួយនៅក្នុងវដ្តកម្មមួយទៀត ក្រុមបញ្ជាដែលជាវដ្តកម្មនៅខាងក្នុងត្រូវយកទៅអនុវត្តរហូតដល់លក្ខខ័ណ្ឌត្រូវបំពេញ រាល់លើកដែលក្រុមបញ្ជាដែលជាវដ្តកម្មនៅខាងក្រៅត្រូវយកទៅអនុវត្តម្តង។ ជាក់ស្តែង នៅក្នុងកម្មវិធីខាងលើនេះ រាល់លើកដែលធាតុនៃកម្រងអថេរឈ្មោះ កម្រងចំនួនគត់ ត្រូវបានយកមកពិនិត្យ និងក្រុមបញ្ជានៅក្នុងបញ្ជា for ដែលជាវដ្តកម្មនៅខាងក្រៅត្រូវយកទៅអនុវត្តម្តង ក្រុមបញ្ជានៅក្នុងបញ្ជា for ដែលជាវដ្តកម្មនៅខាងក្នុងត្រូវយកទៅអនុវត្តជាច្រើនលើកច្រើនសាររហូតដល់ធាតុនៃកម្រងអថេរឈ្មោះ កម្រងចំនួនពិត ត្រូវពិនិត្យអស់។



ក្បួន

ការបង្កើត ក្បួន

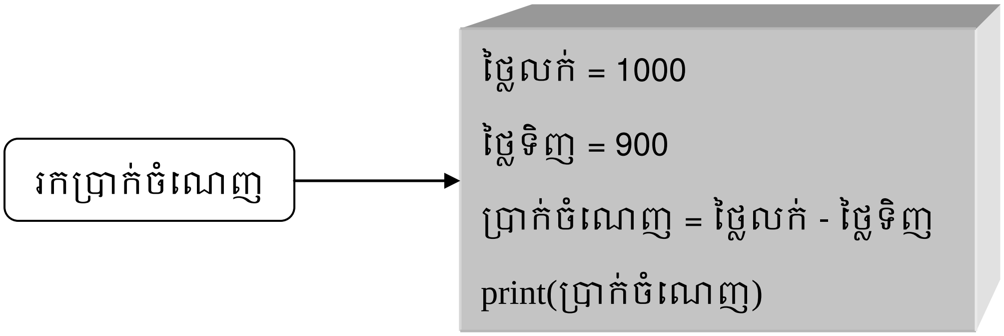
ក្បួន (function) គឺជាវត្ថុដែលជាកន្លែងមួយនៅក្នុងសតិរបស់កំព្យូទ័រប្រើសម្រាប់កត់ត្រាទុកនូវបញ្ជាមួយចំនួនដែលទាក់ទងគ្នាក្នុងការដោះស្រាយកូនបញ្ហាណាមួយ។ ដើម្បីបង្កើតក្បួន យើងត្រូវសរសេរកម្មវិធីដូចខាងក្រោមនេះ៖

```
def រកប្រាក់ចំណេញ() :
    ថ្លៃលក់ = 1000
    ថ្លៃទិញ = 900
    ប្រាក់ចំណេញ = ថ្លៃលក់ - ថ្លៃទិញ
    print(ប្រាក់ចំណេញ)
```

def *រកប្រាក់ចំណេញ*() : គឺជាការប្រើបញ្ជា def តម្រូវឲ្យបង្កើតវត្ថុមួយដែលជាក្បួនមានឈ្មោះថា រកប្រាក់ចំណេញ ដើម្បីកត់ត្រាទុកក្រុមបញ្ជានៅក្នុងបញ្ជា def នោះនៅកន្លែងណាមួយនៅក្នុងសតិរបស់កំព្យូទ័រ។

បន្ទាត់ដែលមានបញ្ជា def នៅក្នុងនោះហៅថា *ក្បាលក្បួន* (function header) ចំណែកឯក្រុមបញ្ជានៅក្នុងបញ្ជា def នោះត្រូវហៅថា *តួក្បួន* (function body) ។ ឈ្មោះរបស់ក្បួនក៏ដូចជាឈ្មោះនៃវត្ថុផ្សេងៗទៀតដែរ គឺត្រូវបង្កើតឡើងដោយគោរពទៅតាមវិធាននៃការបង្កើតឈ្មោះនៅក្នុងភាសា Python ។

នៅក្នុងកម្មវិធីខាងលើនេះ នៅពេលដែលបញ្ជា def ត្រូវបានយកទៅអនុវត្ត វត្ថុមួយមានឈ្មោះថា រកប្រាក់ចំណេញ ត្រូវបានបង្កើតឡើងសម្រាប់កត់ត្រាទុកក្រុមបញ្ជាដែលជាតួក្បួននោះ។ យើងត្រូវធ្វើការកត់សំគាល់ថា ក្រុមបញ្ជាដែលជាតួក្បួនមិនបានត្រូវយកទៅអនុវត្តទេ គឺវាគ្រាន់តែត្រូវបានកត់ត្រាទុកមួយអន្លើតែប៉ុណ្ណោះ។ ហេតុដូចនេះហើយបានជាយើងមិនឃើញមានអ្វីកើតឡើងទេនៅពេលដែលក្បួនឈ្មោះ រកប្រាក់ចំណេញ ត្រូវបានបង្កើតឡើង។ ដោយក្រុមបញ្ជានៅក្នុងក្បួនមិនទាន់ត្រូវយកទៅអនុវត្តនៅពេលក្បួនត្រូវបានបង្កើត ដូចនេះនៅពេលបង្កើតក្បួន បញ្ជាទាំងនោះមានលក្ខណៈជាគម្រោងការ។



ដោយក្បួនក៏ជាវត្ថុមួយដូចជាវត្ថុដទៃទៀតដែរ ដូចនេះយើងអាចយកឈ្មោះរបស់ក្បួនទៅភ្ជាប់នឹងវត្ថុផ្សេងៗទៀតបានតាមចិត្ត។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
def រកប្រាក់ចំណេញ() :
   ថ្លៃលក់ = 1000
   ថ្លៃទិញ = 900
   ប្រាក់ចំណេញ = ថ្លៃលក់ - ថ្លៃទិញ
   print(ប្រាក់ចំណេញ)

រកប្រាក់ចំណេញ = 1.33
```

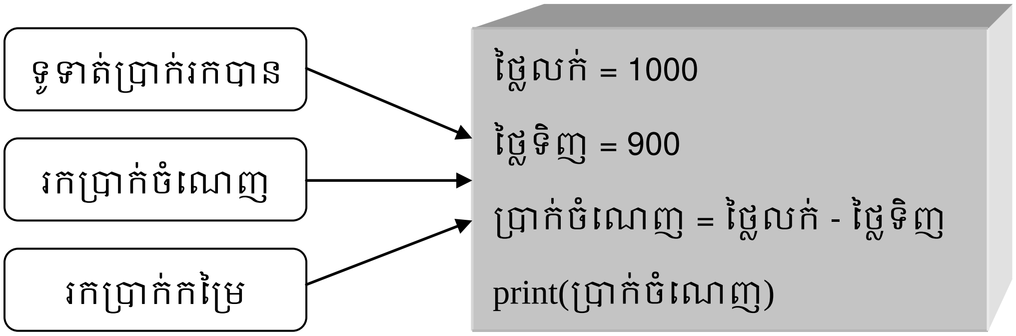
```
print(រកប្រាក់ចំណេញ)
```

រកប្រាក់ចំណេញ = 1.33 គឺជាបញ្ជាតម្រូវឲ្យយកឈ្មោះ រកប្រាក់ចំណេញ ដែលជាឈ្មោះរបស់ក្បួនមួយរួចទៅហើយនោះ ទៅភ្ជាប់នឹងវត្ថុដែលជាលេខ 1.33 វិញម្តង។ ជាផលវិបាកក្បួននោះក្លាយជាវត្ថុគ្មានឈ្មោះដែលត្រូវបានលុបចេញវិញពីក្នុងសតិរបស់កំព្យូទ័រដោយយន្តការបោសសំអាត។ ដូចនេះបញ្ជាទាំងឡាយដែលត្រូវបានកត់ត្រាទុកនៅក្នុងក្បួននោះក៏ត្រូវបានលុបចោលអស់ដែរ។

ដោយក្បួនក៏ជាវត្ថុមួយដែរ ដូចនេះយើងអាចភ្ជាប់ឈ្មោះជាច្រើនទៅនឹងវត្ថុដែលជាក្បួននោះ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
def រកប្រាក់ចំណេញ() :
    ថ្លៃលក់ = 1000
    ថ្លៃទិញ = 900
    ប្រាក់ចំណេញ = ថ្លៃលក់ - ថ្លៃទិញ
    print(ប្រាក់ចំណេញ)

ទូទាត់ប្រាក់រកបាន = រកប្រាក់កម្រៃ = រកប្រាក់ចំណេញ
print(ទូទាត់ប្រាក់រកបាន)
print(រកប្រាក់កម្រៃ)
print(រកប្រាក់ចំណេញ)
```

ទូទាត់ប្រាក់រកបាន = រកប្រាក់កម្រៃ = រកប្រាក់ចំណេញ គឺជាបញ្ជាតម្រូវឲ្យភ្ជាប់ឈ្មោះ ទូទាត់ប្រាក់រកបាន និង រកប្រាក់កម្រៃ ទៅនឹងក្បួនដែលមានឈ្មោះដើមថា រកប្រាក់ចំណេញ ។ ជាលទ្ធផល ក្បួនឈ្មោះ រកប្រាក់កម្រៃ មានឈ្មោះរហូតដល់ទៅបី។



ការយកក្បួនមកប្រើ

យើងបានដឹងរួចមកហើយថា ការបង្កើតក្បួនគឺជាការកត់ត្រាទុកនូវបញ្ជាមួយចំនួននៅកន្លែងណាមួយនៅក្នុងសតិរបស់កំព្យូទ័រ ហើយបញ្ជាទាំងនោះមិនទាន់ត្រូវបានយកទៅអនុវត្តឡើយ។ ដើម្បីឲ្យក្រុមបញ្ជានៅក្នុងក្បួនត្រូវយកទៅអនុវត្ត យើងត្រូវសរសេរកម្មវិធីដូចខាងក្រោមនេះ

```
def រកប្រាក់ចំណេញ() :
    ថ្លៃលក់ = 1000
    ថ្លៃទិញ = 900
    ប្រាក់ចំណេញ = ថ្លៃលក់ - ថ្លៃទិញ
    print(ប្រាក់ចំណេញ)

រកប្រាក់ចំណេញ()
```

រកប្រាក់ចំណេញ() គឺជាការយកក្បួនឈ្មោះ រកប្រាក់ចំណេញ មកប្រើដែលជាបញ្ជាតម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងក្បួននោះតាំងពីដើមរហូតដល់ចប់។

ដូចនេះយើងឃើញថា ដើម្បីឲ្យក្រុមបញ្ជាដែលជាតួក្បួនត្រូវយកទៅអនុវត្ត យើងចាំបាច់ត្រូវតែយកក្បួននោះមកប្រើតាមរបៀបដូចខាងលើនេះ។ ម្យ៉ាងទៀត *ការយកក្បួនមកប្រើ* (calling a

function) ក៏ដូចជាការយកបញ្ជាធម្មតាផ្សេងៗទៀតមកប្រើដែរ ពោលគឺជាការតម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងក្បួននោះពីលើចុះក្រោមនិងពីឆ្វេងទៅស្តាំតាំងពីដើមដល់ចប់។ ដូចនេះការយកក្បួមមកប្រើក៏ជាបញ្ជាមួយដូចជាបញ្ជាដទៃទៀតដែរ ពោលគឺជាបញ្ជាដែលតម្រូវឲ្យអនុវត្តក្រុមបញ្ជាដែលជាតួក្បួននោះ។

សរុបមកយើងឃើញថា ការបង្កើតក្បួនគឺជាការកត់ត្រាទុកក្រុមបញ្ជាដែលជាតួក្បួននៅកន្លែងណាមួយនៅក្នុងសតិរបស់កំព្យូទ័រ។ ហើយដើម្បីឲ្យក្រុមបញ្ជាដែលជាតួក្បួននោះត្រូវយកទៅអនុវត្ត យើងត្រូវយកក្បួននោះមកប្រើ។

ការយកក្បួនមកប្រើ អាចត្រូវធ្វើឡើងចំនួនប៉ុន្មានដងក៏បានដែរ ហើយគ្រប់ការយកក្បួនដដែលមកប្រើ ក្រុមបញ្ជាដែលជាតួក្បួននោះត្រូវយកទៅអនុវត្តសារជាថ្មីឡើងវិញជាដដែលៗ ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
def រកប្រាក់ចំណេញ() :
    ថ្លៃលក់ = 1000
    ថ្លៃទិញ = 900
    ប្រាក់ចំណេញ = ថ្លៃលក់ - ថ្លៃទិញ
    print(ប្រាក់ចំណេញ)

រកប្រាក់ចំណេញ()
រកប្រាក់ចំណេញ()
រកប្រាក់ចំណេញ()
```

នៅក្នុងកម្មវិធីខាងលើនេះយើងឃើញថា ក្បួនឈ្មោះ រកប្រាក់ចំណេញ ត្រូវបានយកទៅប្រើចំនួនបីដងបន្តបន្ទាប់គ្នា។ ដោយគ្រប់ការយកក្បួនទៅប្រើបណ្តាលឲ្យក្រុមបញ្ជានៅក្នុងនោះត្រូវយកទៅអនុវត្តជាដដែលៗ ដូចនេះយើងបានលទ្ធផលដដែលៗចំនួនបីដូចៗគ្នា។

ក្នុងករណីក្បួនមួយមានឈ្មោះជាច្រើន ក្បួននោះអាចត្រូវយកទៅប្រើតាមរយៈឈ្មោះណាមួយក៏បានដែរ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
def រកប្រាក់ចំណេញ() :
    ថ្លៃលក់ = 1000
    ថ្លៃទិញ = 900
    ប្រាក់ចំណេញ = ថ្លៃលក់ - ថ្លៃទិញ
    print(ប្រាក់ចំណេញ)

ទូទាត់ប្រាក់រកបាន = រកប្រាក់កម្រៃ = រកប្រាក់ចំណេញ
រកប្រាក់ចំណេញ()
ទូទាត់ប្រាក់រកបាន()
រកប្រាក់កម្រៃ()
```

រកប្រាក់ចំណេញ() គឺជាការយកក្បួនឈ្មោះ រកប្រាក់ចំណេញ មកប្រើតាមរយៈឈ្មោះដើមរបស់វា។

ទូទាត់ប្រាក់រកបាន() គឺជាការយកក្បួនឈ្មោះ រកប្រាក់ចំណេញ មកប្រើតាមរយៈឈ្មោះ ទូទាត់ប្រាក់រកបាន ។

រកប្រាក់កម្រៃ() គឺជាការយកក្បួនឈ្មោះ រកប្រាក់ចំណេញ មកប្រើតាមរយៈឈ្មោះ រកប្រាក់កម្រៃ ។

ដោយការយកក្បួនមកប្រើក៏ជាបញ្ជាមួយដូចជាបញ្ជាដទៃទៀតដែរ ដូចនេះយើងអាចយកក្បួនមួយឬច្រើនមកប្រើជាបញ្ជានៅក្នុងក្បួនមួយទៀតបានដោយគ្មានបញ្ហាអ្វីឡើយ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
def រកប្រាក់ចំណេញ() :
    ថ្លៃលក់ = 1000
    ថ្លៃទិញ = 900
    ប្រាក់ចំណេញ = ថ្លៃលក់ - ថ្លៃទិញ
    print(ប្រាក់ចំណេញ)

def បង្ហាញប្រាក់ចំណេញ() :
    print("ប្រាក់ចំណេញទាំងអស់មានដូចខាងក្រោមនេះ៖")
    រកប្រាក់ចំណេញ()

បង្ហាញប្រាក់ចំណេញ()
```

រកប្រាក់ចំណេញ() គឺជាការយកក្បួនឈ្មោះ រកប្រាក់ចំណេញ មកប្រើជាបញ្ជានៅក្នុងក្បួនឈ្មោះ បង្ហាញប្រាក់ចំណេញ ។

បង្ហាញប្រាក់ចំណេញ() គឺជាការយកក្បួនឈ្មោះ បង្ហាញប្រាក់ចំណេញ មកប្រើដែលជាប្រការធ្វើឲ្យបញ្ជាទាំងអស់នៅក្នុងនោះត្រូវយកទៅអនុវត្ត។ ប្រការនេះក៏ធ្វើឲ្យក្បួនឈ្មោះ រកប្រាក់ចំណេញ ក៏ត្រូវយកមកប្រើដែរ ព្រោះវាត្រូវបានយកមកប្រើជាបញ្ជាមួយនៅក្នុងនោះ ។

ប្រការដ៏សំខាន់មួយទៀតគឺថា បញ្ជាទាំងឡាយនៅក្នុងក្បួនត្រូវយកទៅអនុវត្តតែនៅពេលណាដែលក្បួនត្រូវយកទៅប្រើតែប៉ុណ្ណោះ។ ដូចនេះយើងអាចយកវត្ថុផ្សេងៗដែលមិនទាន់ត្រូវបានបង្កើតមកប្រើជាបញ្ជានៅក្នុងក្បួនបាន មុនពេលក្បួននោះត្រូវយកទៅប្រើ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
def បង្ហាញប្រាក់ចំណេញ() :
```

```
    print("ប្រាក់ចំណេញទាំងអស់មានដូចខាងក្រោមនេះ៖")
    រកប្រាក់ចំណេញ()

def រកប្រាក់ចំណេញ() :
    ប្រាក់ចំណេញ = ថ្លៃលក់ - ថ្លៃទិញ
    print(ប្រាក់ចំណេញ)

ថ្លៃលក់ = 1000
ថ្លៃទិញ = 900
បង្ហាញប្រាក់ចំណេញ()
```

រកប្រាក់ចំណេញ() គឺជាការយកក្បួនមិនទាន់ត្រូវបានបង្កើតឈ្មោះ រកប្រាក់ចំណេញ មកប្រើជាបញ្ជានៅក្នុងក្បួនឈ្មោះ បង្ហាញប្រាក់ចំណេញ ។

ដូចនេះនៅពេលដែលក្បួនឈ្មោះ បង្ហាញប្រាក់ចំណេញ ត្រូវបង្កើត បញ្ជាដែលជាការយកក្បួនឈ្មោះ រកប្រាក់ចំណេញ មកប្រើត្រូវបានកត់ត្រាទុកនៅក្នុងក្បួនឈ្មោះ បង្ហាញប្រាក់ចំណេញនោះ។ យើងសង្កេតឃើញថាក្បួនឈ្មោះ រកប្រាក់ចំណេញ មិនទាន់ត្រូវបានបង្កើតនៅឡើយទេ តែវាត្រូវបានយកទៅប្រើនៅក្នុងក្បួនឈ្មោះ បង្ហាញប្រាក់ចំណេញ ។ ប្រការនេះអាចធ្វើទៅបានពីព្រោះបញ្ជាដែលជាការយកក្បួនឈ្មោះ រកប្រាក់ចំណេញ មកប្រើគ្រាន់តែត្រូវបានកត់ត្រាទុកតែប៉ុណ្ណោះ គឺមិនទាន់ត្រូវយកទៅអនុវត្តឡើយ។ បានន័យថា វាស្ថិតនៅជាគម្រោងមួយនៅឡើយ។

ប្រាក់ចំណេញ = ថ្លៃលក់ - ថ្លៃទិញ គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកវត្ថុឈ្មោះ ថ្លៃលក់ និង ថ្លៃទិញ មកធ្វើប្រមាណវិធីដកវាងគ្នាដើម្បីបង្កើតវត្ថុមួយថ្មីទៀតមានឈ្មោះថា ប្រាក់ចំណេញ ។

យើងឃើញថាវត្ថុឈ្មោះ ថ្លៃលក់ និងវត្ថុឈ្មោះ ថ្លៃទិញ ជាវត្ថុដែលមិនទាន់ត្រូវបានបង្កើតនៅឡើយ នៅពេលដែលវត្ថុទាំងនោះត្រូវបានយកមកប្រើនៅក្នុងក្បួនឈ្មោះ រកប្រាក់ចំណេញ ។ ប្រការនេះអាចធ្វើទៅបាន ពីព្រោះការយកវត្ថុឈ្មោះ ថ្លៃលក់ និងវត្ថុឈ្មោះ ថ្លៃទិញ មកប្រើនៅក្នុងកន្សោមប្រមាណវិធីគឺគ្រាន់តែត្រូវបានកត់ត្រាទុកនិងមានលក្ខណៈជាគម្រោងតែប៉ុណ្ណោះ មិនមែនជាបញ្ជាតម្រូវឲ្យយកទៅអនុវត្តភ្លាមនោះទេ។ យើងគួររំលឹកឡើងវិញថា គ្រប់បញ្ជាទាំងអស់នៅក្នុងក្បួនមិនទាន់ត្រូវបានយកទៅអនុវត្តនៅឡើយទេ នៅពេលក្បួនត្រូវបានបង្កើត គឺវាគ្រាន់តែត្រូវបានកត់ត្រាទុកមួយអន្លើតែប៉ុណ្ណោះ។

បង្ហាញប្រាក់ចំណេញ() គឺជាការយកក្បួនឈ្មោះ បង្ហាញប្រាក់ចំណេញ មកប្រើដែលជាបញ្ជាតម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងនោះ។ នៅពេលដែលបញ្ជាទាំងនោះត្រូវបានយកទៅអនុវត្ត វត្ថុផ្សេងៗដែលត្រូវបានយកមកប្រើនៅក្នុងក្បួនឈ្មោះ បង្ហាញប្រាក់ចំណេញ ត្រូវតែត្រូវបានបង្កើតរួចហើយ បើពុំនោះសោតទេកំហុសនឹងកើតមានឡើង។

សរុបមក នៅពេលបង្កើតក្បួន យើងអាចយកវត្ថុផ្សេងៗដែលមិនទាន់ត្រូវបានបង្កើតមកប្រើនៅក្នុងក្បួនបាន តែនៅពេលដែលក្បួននោះត្រូវយកទៅប្រើ ចាំបាច់វត្ថុទាំងនោះត្រូវតែត្រូវបានបង្កើត បើពុំនោះសោតទេកំហុសនឹងកើតមានឡើង។

ដំណើង និងដំណាង

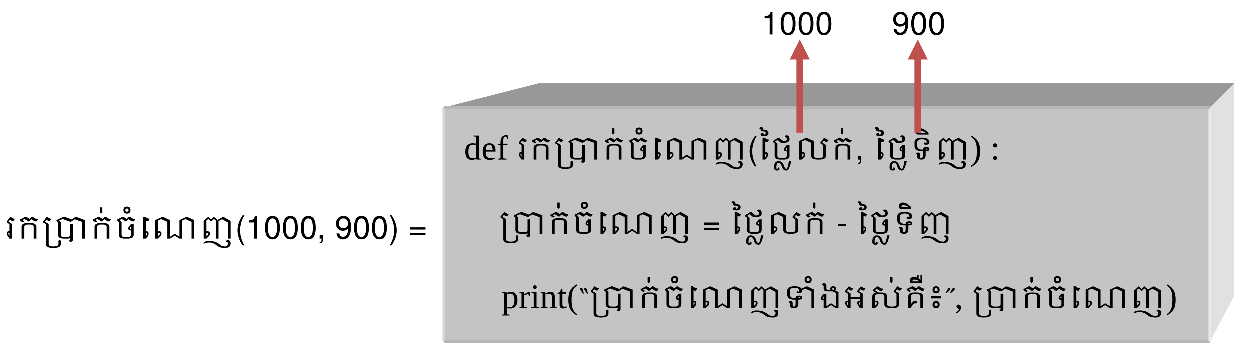
យើងអាចបង្កើតក្បួនឈ្មោះ រកប្រាក់ចំណេញ ដែលយើងបានឃើញកន្លងមកតាមរបៀបម៉្យាងទៀតដូចខាងក្រោមនេះ៖

```
def រកប្រាក់ចំណេញ(ថ្លៃលក់, ថ្លៃទិញ) :
    ប្រាក់ចំណេញ = ថ្លៃលក់ - ថ្លៃទិញ
```

```
    print("ប្រាក់ចំណេញទាំងអស់គឺ៖", ប្រាក់ចំណេញ)

រកប្រាក់ចំណេញ(1000, 900)
```

def *រកប្រាក់ចំណេញ(ថ្លៃលក់, ថ្លៃទិញ)* : គឺជាបញ្ជាតម្រូវឲ្យបង្កើតក្បួនឈ្មោះ រកប្រាក់ចំណេញ មួយដែលមានឈ្មោះ ថ្លៃលក់ និង ថ្លៃទិញ ជាឈ្មោះនៅក្នុងរង្វង់ក្រចកនៅក្បាលក្បួន។ ឈ្មោះ ថ្លៃលក់ និង ថ្លៃទិញ គឺជាឈ្មោះដែលមិនទាន់ត្រូវបានភ្ជាប់ទៅនឹងវត្ថុណាទាំងអស់ ដែលជាប្រការមិនអាចធ្វើទៅបានឡើយនៅខាងក្រៅក្បួន។ ឈ្មោះទាំងនោះត្រូវហៅថា *ដំណាង* (parameter) ។ ដំណាងដែលជាឈ្មោះនៅក្បាលក្បួនក៏ត្រូវចាត់ទុកថាស្ថិតនៅក្នុងក្បួនដែរ។ លើសពីនេះទៀត ដំណាងទាំងនោះត្រូវយកមកប្រើនៅក្នុងក្បួនធ្វើហាក់ដូចជាវាត្រូវបានភ្ជាប់ទៅនឹងវត្ថុណាមួយរួចទៅហើយ។ ប្រការនេះអាចត្រូវធ្វើទៅបានដោយសារបញ្ជានៅក្នុងក្បួនមិនទាន់ត្រូវយកទៅអនុវត្ត គឺវាគ្រាន់តែជាគម្រោងការមួយតែប៉ុណ្ណោះ។ ក៏ប៉ុន្តែ នៅពេលដែលក្បួនត្រូវយកទៅប្រើ ដែលជាបញ្ជាតម្រូវឲ្យយកក្រុមបញ្ជានៅក្នុងក្បួនទៅអនុវត្ត ដំណាងទាំងនោះចាំបាច់ត្រូវតែត្រូវបានភ្ជាប់ទៅនឹងវត្ថុណាមួយ បើពុំនោះសោតទេ កំហុសនឹងកើតមានឡើង។

រកប្រាក់ចំណេញ(1000, 900) គឺជាយកក្បួនឈ្មោះ រកប្រាក់ចំណេញ មកប្រើដោយតម្រូវឲ្យដំណាង ថ្លៃលក់ ភ្ជាប់ទៅនឹងលេខ 1000 និងដំណាង ថ្លៃទិញ ភ្ជាប់ទៅនឹងលេខ 900 ។ ដូចនេះនៅពេលដែលក្រុមបញ្ជានៅក្នុងក្បួនឈ្មោះ រកប្រាក់ចំណេញ ត្រូវយកទៅអនុវត្ត គ្រប់ការយកដំណាង ថ្លៃលក់ និង ថ្លៃទិញ ទៅប្រើ គឺជាការយកលេខ 1000 និងលេខ 900 ទៅប្រើរៀងគ្នា។ វត្ថុដែលយើងផ្តល់ឲ្យក្បួននៅពេលយកក្បួនមកប្រើ ត្រូវហៅថា *ដំណឹង* (argument) ដែលអាចជាវត្ថុប្រភេទណាក៏បានដែរ។



បញ្ជា return

return គឺជាបញ្ជាប្រើនៅក្នុងក្បួនសម្រាប់បញ្ចប់ការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងក្បួននិងបញ្ជូនវត្ថុណាមួយទៅកន្លែងដែលក្បួនត្រូវបានយកទៅប្រើ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
def រកប្រាក់ចំណេញ(ថ្លៃលក់, ថ្លៃទិញ) :
    ប្រាក់ចំណេញ = ថ្លៃលក់ - ថ្លៃទិញ
    return ប្រាក់ចំណេញ

វត្ថុក្បួនបញ្ជូនមក = រកប្រាក់ចំណេញ(1000, 900)
print(វត្ថុក្បួនបញ្ជូនមក)
```

return *ប្រាក់ចំណេញ* គឺជាការប្រើបញ្ជា return ដើម្បីបញ្ចប់ការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងក្បួនឈ្មោះ រកប្រាក់ចំណេញ និងបញ្ជូនវត្ថុឈ្មោះ ប្រាក់ចំណេញ ទៅកាន់កន្លែងដែលក្បួនឈ្មោះ រកប្រាក់ចំណេញ ត្រូវបានយកទៅប្រើ។

វត្ថុក្បួនបញ្ជូនមក = រកប្រាក់ចំណេញ(1000, 900) គឺជាការយកក្បួនឈ្មោះ រកប្រាក់ចំណេញ មកប្រើដោយផ្តល់ដំណឹងជាលេខ 1000 និងលេខ 900 ឲ្យទៅដំណាង ថ្លៃលក់ និង ថ្លៃទិញ រៀងគ្នា។ ជាលទ្ធផល ក្បួនឈ្មោះ រកប្រាក់ចំណេញ បញ្ជូនវត្ថុមួយដែលជាលេខ 100 ទៅកាន់កន្លែងដែលក្បួននេះត្រូវបានយកទៅប្រើ។ វត្ថុដែលក្បួនឈ្មោះ រកប្រាក់ចំណេញ បញ្ជូនមកនេះត្រូវបានភ្ជាប់ទៅនឹងឈ្មោះ វត្ថុក្បួនបញ្ជូនមក ។

គ្រប់ក្បួនទាំងអស់ដែលគ្មានបញ្ជា return នៅក្នុងនោះ បញ្ជូនមោឃៈវត្ថុ None ទៅកាន់កន្លែងដែលក្បួនត្រូវបានយកទៅប្រើ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
def រកប្រាក់ចំណេញ(ថ្លៃលក់, ថ្លៃទិញ) :
    ប្រាក់ចំណេញ = ថ្លៃលក់ - ថ្លៃទិញ

វត្ថុក្បួនបញ្ជូនមក = រកប្រាក់ចំណេញ(1000, 900)
print(វត្ថុក្បួនបញ្ជូនមក)
```

វត្ថុក្បួនបញ្ជូនមក = រកប្រាក់ចំណេញ(1000, 900) គឺជាការយកក្បួនឈ្មោះ រកប្រាក់ចំណេញ មកប្រើ និងភ្ជាប់ឈ្មោះ វត្ថុក្បួនបញ្ជូនមក ទៅនឹងវត្ថុដែលក្បួននោះបញ្ជូនមក។ ដោយនៅក្នុងក្បួននោះគ្មានការប្រើបញ្ជា return ដើម្បីបញ្ជូនវត្ថុណាមួយនោះ ដូចនេះក្បួនឈ្មោះ រកប្រាក់ចំណេញ បញ្ជូនមោឃៈវត្ថុ None ទៅកាន់កន្លែងដែលក្បួននោះត្រូវបានយកទៅប្រើ។ ជាទូទៅ យើងមិនសូវរវីរវល់នឹងមោឃៈវត្ថុដែលក្បួនផ្សេងៗបញ្ជូនមកនោះទេ។

ដំណឹងតាមលេខរៀង

ដំណឹងតាមលេខរៀង (positional argument) គឺជាដំណឹងទាំងឡាយណាដែលយើងផ្តល់ឲ្យទៅក្បួនដោយត្រូវតម្រៀបទៅតាមលេខរៀងនៃដំណាងនៅក្នុងក្បួន។ ដូចនេះការផ្តល់ដំណឹង

ឲ្យទៅក្បួនទាំងឡាយកន្លងមក គឺជាការផ្តល់ដំណឹងតាមលេខរៀង។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
def រកប្រាក់ចំណេញ(ថ្លៃលក់, ថ្លៃទិញ) :
    ប្រាក់ចំណេញ = ថ្លៃលក់ - ថ្លៃទិញ
    print("ប្រាក់ចំណេញទាំងអស់គឺ៖", ប្រាក់ចំណេញ)

រកប្រាក់ចំណេញ(1000, 900)
```

រកប្រាក់ចំណេញ(1000, 900) គឺជាការយកក្បួនឈ្មោះ រកប្រាក់ចំណេញ មកប្រើដោយផ្តល់ដំណឹងតាមលេខរៀងឲ្យវា។ បានន័យថា ដំណឹងជាលេខ 1000 គឺសម្រាប់ដំណាង ថ្លៃលក់ និងដំណឹងជាលេខ 900 គឺសម្រាប់ដំណាង ថ្លៃទិញ ។

ដំណឹងតាមដំណាង

ដំណឹងតាមដំណាង (keyword argument) គឺជាដំណឹងទាំងឡាយណាដែលត្រូវបានភ្ជាប់ទៅនឹងដំណាងផ្សេងៗមុនរួចជាស្រេចនៅពេលយកក្បួនមកប្រើ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
def រកប្រាក់ចំណេញ(ថ្លៃលក់, ថ្លៃទិញ) :
    ប្រាក់ចំណេញ = ថ្លៃលក់ - ថ្លៃទិញ
    print("ប្រាក់ចំណេញទាំងអស់គឺ៖", ប្រាក់ចំណេញ)

រកប្រាក់ចំណេញ(ថ្លៃទិញ=900, ថ្លៃលក់=1000)
```

រកប្រាក់ចំណេញ(ថ្លៃទិញ=900, ថ្លៃលក់=1000) គឺជាការយកក្បួនឈ្មោះ រកប្រាក់ចំណេញ មកប្រើដោយផ្តល់ដំណឹងតាមដំណាងឲ្យវា។ បានន័យថា ដំណឹងដែលជាលេខ 900 គឺសម្រាប់ដំណាង ថ្លៃទិញ និងដំណឹងដែលជាលេខ 1000 គឺសម្រាប់ដំណាង ថ្លៃលក់ ។

នៅក្នុងកម្មវិធីខាងលើនេះ យើងសង្កេតឃើញថា ការផ្តល់ដំណឹងតាមដំណាងឲ្យទៅក្បួនមិនចាំបាច់ត្រូវគោរពទៅតាមលេខរៀងនៃដំណាងឡើយ ព្រោះដំណាងនិមួយៗត្រូវបានភ្ជាប់រួចជាស្រេចទៅនឹងដំណឹងទាំងនោះ។

ដំណឹងមានស្រាប់

ដំណឹងមានស្រាប់ (default argument) គឺជាដំណឹងទាំងឡាយណាដែលមានមុនរួចជាស្រេចនៅពេលក្បួនត្រូវបានបង្កើត។ ហើយនៅពេលដែលក្បួនត្រូវយកទៅប្រើ បើក្បួនមិនបានទទួលដំណឹងឬទទួលដំណឹងមិនគ្រប់គ្រាន់ ក្បួននឹងយកដំណឹងមានស្រាប់ទាំងនោះមកប្រើ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
def រកប្រាក់ចំណេញ(ថ្លៃលក់=0, ថ្លៃទិញ=0) :
    ប្រាក់ចំណេញ = ថ្លៃលក់ - ថ្លៃទិញ
    print("ប្រាក់ចំណេញទាំងអស់គឺ៖", ប្រាក់ចំណេញ)

រកប្រាក់ចំណេញ()
រកប្រាក់ចំណេញ(1000)
រកប្រាក់ចំណេញ(ថ្លៃទិញ=900)
រកប្រាក់ចំណេញ(1000, 900)
```

def *រកប្រាក់ចំណេញ*(*ថ្លៃលក់*=0, *ថ្លៃទិញ*=0) : គឺជាបញ្ជាតម្រូវឲ្យបង្កើតក្បួនឈ្មោះ រកប្រាក់ចំណេញ មួយដែលមានដំណឹងមួយចំនួនរួចជាស្រេចសម្រាប់ដំណាងផ្សេងៗនៅក្នុងនោះ។ ពោលគឺដំណឹងសម្រាប់ដំណាង ថ្លៃលក់ គឺជាលេខ 0 និងដំណឹងសម្រាប់ដំណាង ថ្លៃទិញ ក៏ជាលេខ 0 ដែរ។ ដំណឹងទាំងអស់នោះត្រូវហៅថាដំណឹងមានស្រាប់។

រកប្រាក់ចំណេញ() គឺជាការយកក្បួនឈ្មោះ រកប្រាក់ចំណេញ មកប្រើដោយមិនផ្តល់ដំណឹងអ្វីទាំងអស់ឲ្យវា។ ក្នុងករណីនេះ ក្បួនឈ្មោះ រកប្រាក់ចំណេញ យកដំណឹងមានស្រាប់ជាប់នឹងដំណាងផ្សេងៗមកប្រើ។

រកប្រាក់ចំណេញ(1000) គឺជាការយកក្បួនឈ្មោះ រកប្រាក់ចំណេញ មកប្រើដោយផ្តល់ដំណឹងតែមួយសម្រាប់ដំណាងនៅខាងដើមគេដែលជាដំណាង ថ្លៃលក់ ។ ក្នុងករណីនេះ ក្បួនឈ្មោះ រកប្រាក់ចំណេញ យកដំណឹងមានស្រាប់ដែលជាលេខ 0 មកប្រើសម្រាប់ដំណាង ថ្លៃទិញ ។

រកប្រាក់ចំណេញ(*ថ្លៃទិញ*=900) គឺជាការយកក្បួនឈ្មោះ រកប្រាក់ចំណេញ មកប្រើដោយផ្តល់ដំណឹងតាមដំណាងតែមួយសម្រាប់ដំណាង ថ្លៃទិញ ។ ក្នុងករណីនេះ ក្បួនឈ្មោះ រកប្រាក់ចំណេញ យកដំណឹងមានស្រាប់ដែលជាលេខ 0 មកប្រើសម្រាប់ដំណាង ថ្លៃលក់ ។

រកប្រាក់ចំណេញ(1000, 900) គឺជាការយកក្បួនឈ្មោះ រកប្រាក់ចំណេញ មកប្រើដោយផ្តល់ដំណឹងមានចំនួនគ្រប់គ្រាន់ឲ្យទៅក្បួននោះ។ ក្នុងករណីនេះ ដំណឹងមានស្រាប់មិនត្រូវបានយកមកប្រើទេ គឺត្រូវទុកចោល។

យើងបានដឹងរួចមកហើយថា បញ្ជាផ្សេងៗនៅក្នុងក្បួនមិនត្រូវបានយកទៅអនុវត្តទេនៅពេលក្បួនត្រូវបានបង្កើត។ បញ្ជាទាំងនោះត្រូវយកទៅអនុវត្តតែនៅពេលណាដែលក្បួនត្រូវយកទៅ

ប្រើតែប៉ុណ្ណោះ។ ក៏ប៉ុន្តែចំពោះក្បួនដែលមានដំណឹងមានស្រាប់ បញ្ញាចាត់តាំងភ្ជាប់ដំណាងទៅនឹងដំណឹងទាំងឡាយត្រូវយកទៅអនុវត្តនៅពេលក្បួនត្រូវបានបង្កើត។ ដូចនេះនៅក្នុងកម្មវិធីខាងលើ បញ្ញាដែលមានទម្រង់ជា def *រកប្រាក់ចំណេញ(ថ្លៃលក់*=0, *ថ្លៃទិញ*=0) : គឺជាបញ្ញាតម្រូវឲ្យបង្កើតក្បួនឈ្មោះ រកប្រាក់ចំណេញ និងតម្រូវឲ្យយកបញ្ញាចាត់តាំង ថ្លៃលក់=0 និង ថ្លៃទិញ=0 ទៅអនុវត្តព្រមពេលជាមួយគ្នា។

ម៉្យាងទៀតយើងត្រូវធ្វើការកត់សំគាល់ផងដែរថា ដំណឹងរតាមដំណាងនិងដំណឹងមានស្រាប់មានទម្រង់ដូចគ្នាគឺ *ដំណាង* = *ដំណឹង* តែវាជារឿងពីរខុសគ្នាស្រឡះ។ ដំណឹងតាមដំណាងត្រូវផ្តល់ឲ្យទៅក្បួននៅពេលយកក្បួនមកប្រើ ចំណែកឯដំណឹងមានស្រាប់ត្រូវបានបង្កើតឡើងនៅពេលក្បួនត្រូវបានបង្កើត។ លើសពីនេះទៀត យើងអាចផ្តល់ដំណឹងតាមដំណាងឲ្យទៅក្បួនមានដំណឹងមានស្រាប់ដោយគ្មានបញ្ហាអ្វីឡើយ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
def រកប្រាក់ចំណេញ(ថ្លៃលក់=0, ថ្លៃទិញ=0) :
    ប្រាក់ចំណេញ = ថ្លៃលក់ - ថ្លៃទិញ
    print("ប្រាក់ចំណេញទាំងអស់គឺ៖", ប្រាក់ចំណេញ)

រកប្រាក់ចំណេញ(ថ្លៃលក់=1000, ថ្លៃទិញ=900)
```

def *រកប្រាក់ចំណេញ(ថ្លៃលក់*=0, *ថ្លៃទិញ*=0) : គឺជាបញ្ញាតម្រូវឲ្យបង្កើតក្បួនមួយមានឈ្មោះថា រកប្រាក់ចំណេញ ដែលមានដំណឹងមានស្រាប់រួចជាស្រេចសម្រាប់ដំណាងទាំងពីរនៅក្នុងនោះ។

រកប្រាក់ចំណេញ(ថ្លៃលក់=1000, *ថ្លៃទិញ*=900) គឺជាការយកក្បួនឈ្មោះ រកប្រាក់ចំណេញ ដែលជាក្បួនមានដំណឹងមានស្រាប់មកប្រើ ដោយផ្តល់ដំណឹងតាមដំណាងឲ្យទៅក្បួននោះ។

ការបំបែក ដំណឹង

ការបំបែកដំណឹង (unpacking argument) គឺជាការចម្លងយកសមាសវត្ថុផ្សេងៗមកបំបែកជាដំណឹងផ្តល់ឲ្យទៅក្បួនណាមួយ។ ទង្វើនេះត្រូវធ្វើឡើងនៅពេលយកក្បួនមកប្រើ។

យើងអាចចម្លងយកកម្រងផ្សេងៗមកបំបែកជាដំណឹងផ្តល់ឲ្យទៅក្បួនណាមួយដោយសរសេរកម្មវិធីដូចខាងក្រោមនេះ៖

```
កម្រងប្រាក់ = [1000, 900]
def រកប្រាក់ចំណេញ(ថ្លៃលក់=0, ថ្លៃទិញ=0) :
   ប្រាក់ចំណេញ = ថ្លៃលក់ - ថ្លៃទិញ
   print("ប្រាក់ចំណេញទាំងអស់គឺ៖", ប្រាក់ចំណេញ)

រកប្រាក់ចំណេញ(*កម្រងប្រាក់)
```

*រកប្រាក់ចំណេញ(*កម្រងប្រាក់)* គឺជាការចម្លងយកកម្រងអថេរឈ្មោះ កម្រងប្រាក់ មកបំបែកជាដំណឹងតាមលេខរៀងចំនួនពីរផ្តល់ឲ្យទៅក្បួនឈ្មោះ រកប្រាក់ចំណេញ ។

ដូចនេះយើងឃើញថា ដើម្បីចម្លងយកកម្រងណាមួយមកបំបែកជាដំណឹងផ្តល់ឲ្យទៅក្បួនណាមួយ យើងត្រូវប្រើសញ្ញាផ្កាយ (*) នៅខាងមុខកម្រងនោះ។

ក្រៅពីសមាសវត្ថុដែលជាកម្រង យើងក៏អាចចម្លងយកវចនានុក្រមណាមួយមកបំបែកជាដំណឹងតាមដំណាងសម្រាប់ផ្តល់ឲ្យទៅក្បួនណាមួយបានដែរ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
បញ្ជីប្រាក់ = {"ថ្លៃលក់":1000, "ថ្លៃទិញ":900}
def រកប្រាក់ចំណេញ(ថ្លៃលក់=0, ថ្លៃទិញ=0) :
   ប្រាក់ចំណេញ = ថ្លៃលក់ - ថ្លៃទិញ
```

```
print("ប្រាក់ចំណេញទាំងអស់គឺ៖", ប្រាក់ចំណេញ)

រកប្រាក់ចំណេញ(**បង្វិចប្រាក់)
```

*រកប្រាក់ចំណេញ(**បង្វិចប្រាក់)* គឺជាការចម្លងយករចនានុក្រមឈ្មោះ បង្វិចប្រាក់ មកបំបែកជាដំណឹងតាមដំណាងចំនួនពីរផ្តល់ឲ្យទៅក្បួនឈ្មោះ រកប្រាក់ចំណេញ ។

ដូចនេះយើងឃើញថា ដើម្បីចម្លងយករចនានុក្រមណាមួយមកបំបែកជាដំណឹងតាមដំណាងសម្រាប់ផ្តល់ឲ្យទៅក្បួនណាមួយ យើងត្រូវប្រើសញ្ញាផ្កាយពីរ (**) នៅខាងមុខរចនានុក្រមនោះ។ ម៉្យាងទៀត កូនសោរនៅក្នុងរចនានុក្រមត្រូវតែដូចគ្នាបេះបិទទៅនឹងដំណាងនៅក្នុងក្បួន។

ការប្រមូលផ្តុំដំណឹង

ការប្រមូលផ្តុំដំណឹង (collecting arguments) គឺជាការប្រមូលផ្តុំដំណឹងមានចំនួនមិនកំណត់មកបង្កើតជាសមាសវត្ថុណាមួយទុកនៅក្នុងក្បួនណាមួយ។

យើងអាចប្រមូលផ្តុំដំណឹងតាមលេខរៀងមានចំនួនមិនកំណត់មកបង្កើតជាកម្រងថេរមួយទុកនៅក្នុងក្បួនណាមួយដោយសរសេរកម្មវិធីដូចខាងក្រោមនេះ៖

```
def ប្រមូលផ្តុំ(*កម្រងប្រាក់) :
    print(កម្រងប្រាក់)

ប្រមូលផ្តុំ(1000, 900)
```

def *ប្រមូលផ្តុំ(*កម្រងប្រាក់)* : គឺជាបញ្ជាតម្រូវឲ្យបង្កើតក្បួនឈ្មោះ ប្រមូលផ្តុំ មួយដែលមានសមត្ថភាពអាចប្រមូលយកដំណឹងតាមលេខរៀងមានចំនួនមិនកំណត់មកបង្កើតជាកម្រងថេរ

មួយមានឈ្មោះថា កម្រងប្រាក់ ។ ហើយគឺដោយសារសញ្ញាផ្កាយមួយ (*) នៅមុខដំណាង កម្រងប្រាក់ នេះហើយដែលធ្វើឲ្យក្បួននេះមានសមត្ថភាពអាចប្រមូលយកដំណឹងតាមលេខរៀងមានចំនួនមិនកំណត់មកបង្កើតជាកម្រងថេរឈ្មោះ កម្រងប្រាក់ នេះ។

ប្រមូលផ្តុំ(1000, 900) គឺជាការយកក្បួនឈ្មោះ ប្រមូលផ្តុំ មកប្រើដោយផ្តល់ដំណឹងតាមលេខរៀងចំនួនពីរឲ្យទៅក្បួននោះ។ ដំណឹងទាំងនោះត្រូវបានប្រមូលផ្តុំបង្កើតជាកម្រងថេរមានឈ្មោះថា កម្រងប្រាក់ ទុកនៅក្នុងក្បួននោះ។

ដូចគ្នាដែរ យើងអាចប្រមូលផ្តុំដំណឹងតាមដំណាងមានចំនួនមិនកំណត់យកមកបង្កើតជាវចនានុក្រមមួយទុកនៅក្នុងក្បួនណាមួយដោយសរសេរកម្មវិធីដូចខាងក្រោមនេះ៖

```
def ប្រមូលផ្តុំ(**បង្វិចប្រាក់) :
    print(បង្វិចប្រាក់)

ប្រមូលផ្តុំ(ថ្ងៃលក់=1000, ថ្ងៃទិញ=900)
```

def *ប្រមូលផ្តុំ*(***បង្វិចប្រាក់*) : គឺជាបញ្ជាតម្រូវឲ្យបង្កើតក្បួនឈ្មោះ ប្រមូលផ្តុំ មួយដែលមានសមត្ថភាពអាចប្រមូលផ្តុំយកដំណឹងតាមដំណាងមានចំនួនមិនកំណត់មកបង្កើតជាវចនានុក្រមមួយមានឈ្មោះថា បង្វិចប្រាក់ ។ គឺដោយសារសញ្ញាផ្កាយពីរ (**) នៅខាងមុខដំណាង បង្វិចប្រាក់ នេះហើយដែលធ្វើឲ្យក្បួនឈ្មោះ ប្រមូលផ្តុំ មានសមត្ថភាពអាចប្រមូលផ្តុំដំណឹងតាមដំណាងមានចំនួនមិនកំណត់មកបង្កើតជាវចនានុក្រមទុកនៅក្នុងនោះ។

ប្រមូលផ្តុំ(*ថ្ងៃលក់*=1000, *ថ្ងៃទិញ*=900) គឺជាការយកក្បួនឈ្មោះ ប្រមូលផ្តុំ មកប្រើដោយផ្តល់ដំណឹងតាមដំណាងចំនួនពីរឲ្យទៅក្បួននោះ។ ដំណឹងទាំងនោះត្រូវបានប្រមូលផ្តុំបង្កើតជាវចនានុក្រមមួយមានឈ្មោះថា បង្វិចប្រាក់ ទុកនៅក្នុងក្បួននោះ។

យើងឃើញថា ការបំបែកដំណឹងនិងការប្រមូលផ្តុំដំណឹង គឺសុទ្ធតែត្រូវប្រើសញ្ញាផ្កាយមួយឬពីរនៅខាងមុខដំណឹងឬដំណាងណាមួយដូចគ្នា។ តែប្រការដែលខុសគ្នាគឺនៅត្រង់ថា ចំពោះការបំបែកដំណឹង យើងត្រូវប្រើសញ្ញាផ្កាយនៅខាងមុខដំណឹងនៅពេលយកក្បួនមកប្រើ តែចំពោះការប្រមូលផ្តុំដំណឹងវិញ យើងត្រូវប្រើសញ្ញាផ្កាយនៅខាងមុខដំណាងនៅពេលបង្កើតក្បួន។

លំដាប់ថ្នាក់នៃដំណាង និងដំណឹង

យើងអាចបង្កើតក្បួនមួយដែលនៅក្នុងនោះមានដំណាងធម្មតា ដំណាងមានផ្កាយមួយ និងដំណាងមានផ្កាយពីរ ដោយសរសេរកម្មវិធីដូចខាងក្រោមនេះ៖

```
def ប្រមូល(ប្រាក់ចំណេញ, *កម្រងប្រាក់, **បន្ទិចប្រាក់) :
    print(ប្រាក់ចំណេញ)
    print(កម្រងប្រាក់)
    print(បន្ទិចប្រាក់)

ប្រមូល(100, 1000, 900, ថ្លៃលក់=1000, ថ្លៃទិញ=900)
```

def ប្រមូល(ប្រាក់ចំណេញ, *កម្រងប្រាក់, **បន្ទិចប្រាក់) : គឺជាបញ្ជាតម្រូវឲ្យបង្កើតក្បួនឈ្មោះ ប្រមូល មួយដែលនៅក្នុងនោះមានដំណាងធម្មតា ដំណាងមានផ្កាយមួយ និងដំណាងមានផ្កាយពីរ។ ប្រការនេះធ្វើឲ្យក្បួនឈ្មោះ ប្រមូល នេះមានសមត្ថភាពអាចប្រមូលយកដំណឹងធម្មតាដែលជាដំណឹងតាមលេខរៀង និងដំណឹងតាមដំណាងមកបង្កើតជាវត្ថុទោលកម្រងថេរ និងវចនានុក្រមទុកនៅក្នុងក្បួននោះ។

អាស្រ័យទៅតាមច្បាប់នៅក្នុងភាសា Python នៅពេលបង្កើតក្បួន ដំណាងធម្មតាត្រូវនៅមុខដំណាងមានផ្កាយមួយដែលត្រូវនៅមុខដំណាងមានផ្កាយពីរ។

ប្រមូល(100, 1000, 900, *ថ្លៃលក់*=1000, *ថ្លៃទិញ*=900) គឺជាការយកក្បួនឈ្មោះ ប្រមូល មកប្រើដោយផ្តល់ដំណឹងតាមលេខរៀងនិងដំណឹងតាមដំណាងឲ្យទៅក្បួននោះ។ ដំណឹងតាមលេខរៀងគឺសម្រាប់ដំណាងធម្មតា តែបើដំណឹងនោះមានចំនួនលើសពីដំណាងធម្មតា ដំណឹងនៅសល់ទាំងប៉ុន្មាន គឺសម្រាប់ដំណាងមានផ្កាយមួយ។ ចំពោះដំណឹងតាមដំណឹងវិញ គឺសម្រាប់ដំណាងមានផ្កាយពីរ។ បន្ទាប់ពីបានទទួលដំណឹងរួចមក ក្បួនឈ្មោះ ប្រមូល ក៏ប្រមូលយកដំណឹងទាំងនោះមកបង្កើតជាវត្ថុទោលឈ្មោះ ប្រាក់ចំណេញ កម្រងថេរឈ្មោះ កម្រងប្រាក់ និងវចនានុក្រមឈ្មោះ បង្ខិចប្រាក់ ទុកនៅក្នុងក្បួននោះ។

អាស្រ័យទៅតាមច្បាប់នៅក្នុងភាសា Python នៅក្នុងការផ្តល់ដំណឹងឲ្យទៅក្បួន ដំណឹងធម្មតាដែលជាដំណឹងតាមលេខរៀង ត្រូវនៅខាងមុខដំណឹងតាមដំណាង។

យើងអាចយកក្បួនមួយមកប្រើដោយផ្តល់ឲ្យវានូវដំណឹងធម្មតានិងដំណឹងបានមកពីការចម្លងយកសមាសវត្ថុផ្សេងៗមកបំបែក។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
កម្រងប្រាក់ = [2000, 1500]
បង្ខិចប្រាក់ = {"ថ្លៃលក់":1000, "ថ្លៃទិញ":900}
def សាច់ប្រាក់(ប្រាក់សរុប, ប្រាក់នៅសល់, ប្រាក់ចំណាយ, ថ្លៃលក់=0, ថ្លៃទិញ=0) :
    print(ប្រាក់សរុប)
    print(ប្រាក់នៅសល់)
    print(ប្រាក់ចំណាយ)
    print(ថ្លៃលក់)
    print(ថ្លៃទិញ)
```

```
សាច់ប្រាក់(10000, *កម្រង់ប្រាក់, **បង្វិចប្រាក់)
```

*សាច់ប្រាក់(10000, *កម្រង់ប្រាក់, **បង្វិចប្រាក់)* គឺជាការយកក្បួនឈ្មោះ សាច់ប្រាក់ មកប្រើដោយផ្តល់ដំណឹងធម្មតាដែលជាលេខ 10000 ដំណឹងមានផ្កាយមួយឈ្មោះ *កម្រងប្រាក់ និងដំណឹងមានផ្កាយពីរ **បង្វិចប្រាក់ ឲ្យទៅក្បួននោះ។

អាស្រ័យទៅតាមទៅតាមច្បាប់នៅក្នុងភាសា Python ដំណឹងធម្មតាត្រូវនៅមុខដំណឹងមានផ្កាយមួយដែលត្រូវនៅមុខដំណឹងមានផ្កាយពីរ។

ដែនកំណត់

ដែនកំណត់ (scope) គឺជាទីកន្លែងផ្សេងៗនៅក្នុងសតិរបស់កំព្យូទ័រដែលមានព្រំដែនច្បាស់លាស់។

ដែនកំណត់សកល

ទីកន្លែងនៅខាងក្រៅក្បួននិងនៅក្នុងឯកសារដែលជាកម្មវិធីហៅថា *ដែនកំណត់សកល* (global scope) ។ ដើម្បីបង្កើតវត្ថុផ្សេងៗនៅក្នុងដែនកំណត់សកល យើងត្រូវសរសេរកម្មវិធីដូចខាងក្រោមនេះ៖

```
កម្រង់ប្រាក់ = [2000, 1500]
def រកប្រាក់ចំណេញ(ថ្លៃលក់=0, ថ្លៃទិញ=0) :
   ប្រាក់ចំណេញ = ថ្លៃលក់ - ថ្លៃទិញ
   print("ប្រាក់ចំណេញទាំងអស់គឺ៖", ប្រាក់ចំណេញ)

រកប្រាក់ចំណេញ(*កម្រង់ប្រាក់)
```

កម្រងប្រាក់ = [2000, 1500] គឺជាបញ្ជាតម្រូវឲ្យបង្កើតកម្រងអថេរឈ្មោះ កម្រងប្រាក់ មួយស្ថិតនៅក្នុងដែនកំណត់សកល។

ដែនកំណត់ដោយឡែក

ដែនកំណត់ដោយឡែក (local scope) គឺជាទីកន្លែងនៅក្នុងក្បួន។ ដើម្បីបង្កើតវត្ថុផ្សេងៗនៅក្នុងដែនកំណត់ដោយឡែក យើងត្រូវសរសេរកម្មវិធីដូចខាងក្រោមនេះ៖

```
def រកប្រាក់ចំណេញ(ថ្ងៃលក់=0, ថ្ងៃទិញ=0) :
    កម្រងប្រាក់ = [2000, 1500]
    ថ្ងៃលក់ = កម្រងប្រាក់[0]
    ថ្ងៃទិញ = កម្រងប្រាក់[1]
    ប្រាក់ចំណេញ = ថ្ងៃលក់ - ថ្ងៃទិញ
    print("ប្រាក់ចំណេញទាំងអស់គឺ៖", ប្រាក់ចំណេញ)

រកប្រាក់ចំណេញ()
```

កម្រងប្រាក់ = [2000, 1500] គឺជាបញ្ជាតម្រូវឲ្យបង្កើតកម្រងអថេរឈ្មោះ កម្រងប្រាក់ មួយស្ថិតនៅក្នុងដែនកំណត់ដោយឡែកនៅក្នុងក្បួនឈ្មោះ រកប្រាក់ចំណេញ ។

គ្រប់វត្ថុនៅក្នុងដែនកំណត់ដោយឡែកត្រូវបង្កើតឡើងនៅពេលដែលក្បួនត្រូវយកទៅប្រើ និងត្រូវលុបចោលវិញនៅពេលដែលក្បួនត្រូវបានប្រើរួចហើយ។ ជាក់ស្តែងនៅក្នុងកម្មវិធីខាងលើ កម្រងអថេរឈ្មោះ កម្រងប្រាក់ ត្រូវបានបង្កើតឡើងនៅក្នុងដែនកំណត់ដោយឡែកនៅពេលដែលក្បួនឈ្មោះ រកប្រាក់ចំណេញ ត្រូវបានយកទៅប្រើ ហើយត្រូវលុបចោលវិញនៅពេលដែលក្បួននោះត្រូវបានប្រើរួចហើយ។

ដែនកំណត់ចារឹកក្នុងនិងដែនកំណត់ចារឹកក្រៅ

នៅពេលដែលក្បួនមួយត្រូវបានបង្កើតឡើងនៅក្នុងក្បួនមួយទៀត ទីកន្លែងរបស់ក្បួននៅខាងក្នុងគឺជា *ដែនកំណត់ចារឹកក្នុង* (nested scope) និងទីកន្លែងរបស់ក្បួននៅខាងក្រៅគឺជា *ដែនកំណត់ចារឹងក្រៅ* (enclosing scope) ។ ដើម្បីបង្កើតវត្ថុផ្សេងៗនៅក្នុងដែនកំណត់ចារឹកក្នុង និងនៅក្នុងដែនកំណត់ចារឹកក្រៅ យើងត្រូវសរសេរកម្មវិធីដូចខាងក្រោមនេះ៖

```
def ក្បួនចារឹកក្រៅ() :
    កម្រងប្រាក់ = [2000, 1500]
    print("វត្ថុដែលត្រូវបានបង្កើតឡើងនៅក្នុងដែនកំណត់ចារឹកក្រៅគឺ៖", កម្រងប្រាក់)
    def ក្បួនចារឹកក្នុង() :
        បង្ហិចប្រាក់ = {"ថ្លៃលក់":1000, "ថ្លៃទិញ":900}
        print("វត្ថុដែលត្រូវបានបង្កើតឡើងនៅក្នុងដែនកំណត់ចារឹកក្នុងគឺ៖", បង្ហិចប្រាក់)
    ក្បួនចារឹកក្នុង()

ក្បួនចារឹកក្រៅ()
```

កម្រងប្រាក់ = [2000, 1500] គឺជាបញ្ជាតម្រូវឲ្យបង្កើតកម្រងអថេរមួយមានឈ្មោះថា កម្រងប្រាក់ ស្ថិតនៅក្នុងដែនកំណត់ចារឹកក្រៅនៃក្បួនឈ្មោះ ក្បួនចារឹកក្រៅ ។

បង្ហិចប្រាក់ = {"*ថ្លៃលក់*":1000, "*ថ្លៃទិញ*":900} គឺជាបញ្ជាតម្រូវឲ្យបង្កើតវចនានុក្រមឈ្មោះ បង្ហិចប្រាក់ មួយស្ថិតនៅក្នុងដែនកំណត់ចារឹកក្នុងនៃក្បួនឈ្មោះ ក្បួនចារឹកក្នុង ។

វត្ថុនៅក្នុងដែនកំណត់ចារឹកក្នុងនិងវត្ថុនៅក្នុងដែនកំណត់ចារឹកក្រៅ ក៏ដូចជាវត្ថុនៅក្នុងដែនកំណត់ដោយឡែកដែរ គឺវាត្រូវបានបង្កើតឡើងនៅពេលដែលក្បួនត្រូវបានយកទៅប្រើ និងត្រូវលុបចោលវិញនៅពេលដែលក្បួនត្រូវបានប្រើរួចហើយ។ ក៏ប៉ុន្តែបើសិនណាជាវត្ថុ

ទាំងនោះត្រូវបានបញ្ជូនទៅកាន់ដែនកំណត់នៅខាងក្រៅក្បួន វត្ថុទាំងនោះនឹងមិនត្រូវបាន លុបចោលឡើយនៅពេលដែលក្បួនត្រូវបានប្រើរួចហើយនោះ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
def រកប្រាក់ចំណេញ() :
    ថ្លៃលក់ = 1000
    ថ្លៃទិញ = 900
    ប្រាក់ចំណេញ = ថ្លៃលក់ - ថ្លៃទិញ
    return ប្រាក់ចំណេញ

ប្រាក់កម្រៃ = រកប្រាក់ចំណេញ()
print(ប្រាក់កម្រៃ)
```

ប្រាក់កម្រៃ = រកប្រាក់ចំណេញ() គឺជាបញ្ជាតម្រូវឲ្យភ្ជាប់ឈ្មោះ ប្រាក់កម្រៃ ទៅនឹងវត្ថុដែល ក្បួនឈ្មោះ រកប្រាក់ចំណេញ បញ្ជូនមក។ វត្ថុឈ្មោះ ប្រាក់កម្រៃ នេះគឺជាវត្ថុឈ្មោះ ប្រាក់ចំណេញ ដែលត្រូវបានបង្កើតឡើងនៅក្នុងក្បួនឈ្មោះ រកប្រាក់ចំណេញ ។ ដូចនេះវត្ថុ ឈ្មោះ ប្រាក់ចំណេញ មិនត្រូវបានលុបចោលឡើយទោះបីជាវាត្រូវបានបង្កើតនៅក្នុងក្បួនក៏ ដោយ ព្រោះវាត្រូវបានបញ្ជូនចេញផុតពីក្នុងក្បួនរួចទៅហើយ។

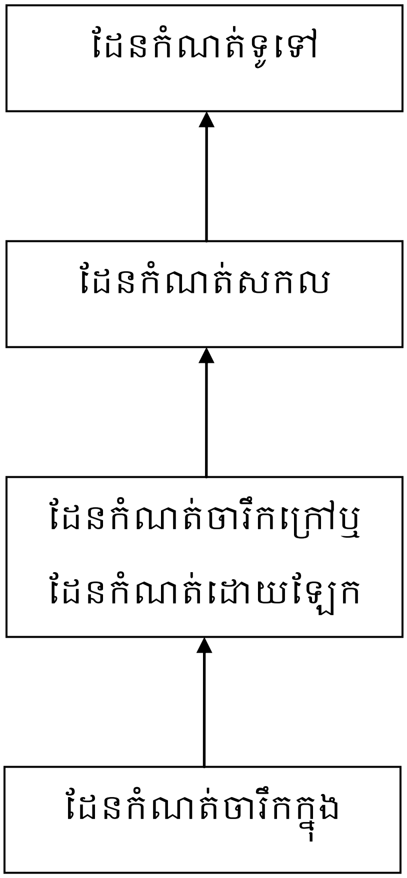
ដែនកំណត់ទូទៅ

ដែនកំណត់ទូទៅ (build-in scope) គឺជាទីកន្លែងមួយនៅក្នុងសតិរបស់កំព្យូទ័រដែលវត្ថុមួយ ចំនួនត្រូវបានបង្កើតឡើងរួចជាស្រេចទុកនៅក្នុងនោះ។ វត្ថុទាំងនោះត្រូវហៅថា *វត្ថុមានស្រាប់* (build-in object) ដែលអាចត្រូវយកទៅប្រើនៅក្នុងដែនកំណត់ណាក៏បានដែរ។

ដែនកំណត់ទូទៅ
ដែនកំណត់សកល
ដែនកំណត់ដោយឡែកឬចារឹកក្រៅ
ដែនកំណត់ចារឹកក្នុង

ការស្វែងរកវត្ថុ

នៅក្នុងភាសា Python នៅពេលដែលយើងយកវត្ថុណាមួយមកប្រើ ការស្វែងរកវត្ថុនោះត្រូវធ្វើឡើងតាមគំនូសបំព្រួញដូចខាងក្រោមនេះ៖



គឺបានន័យថាការស្វែងរកវត្ថុត្រូវធ្វើឡើងដោយចាប់ផ្តើមនៅក្នុងដែនកំណត់ដែលវត្ថុត្រូវបានយកទៅប្រើ រួចបានបន្តទៅដែនកំណត់ផ្សេងៗទៀតតាមសញ្ញាព្រួញដូចនៅក្នុងរូបខាងលើនេះរហូតដល់អស់ដែនកំណត់។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
ឃ្លាសកល = "ដែនកំណត់សកល"
def ក្បួនក្រៅ() :
   ឃ្លាចារឹកក្រៅ = "ដែនកំណត់ដោយឡែកឬចារឹកក្រៅ"
   def ក្បួនក្នុង() :
      ឃ្លាចារឹកក្នុង = "ដែនកំណត់ចារឹកក្នុង"
      print("ឃ្លាសកលត្រូវបានរកឃើញនៅក្នុង ", ឃ្លាសកល)
      print("ឃ្លាចារឹកក្រៅត្រូវបានរកឃើញនៅក្នុង ", ឃ្លាចារឹកក្រៅ)
      print("ឃ្លាចារឹកក្នុងត្រូវបានរកឃើញនៅក្នុង ", ឃ្លាចារឹកក្នុង)
   ក្បួនក្នុង()
ក្បួនក្រៅ()
```

print("ឃ្លាសកលត្រូវបានរកឃើញនៅក្នុង ", ឃ្លាសកល) គឺជាបញ្ជាដែលនៅក្នុងនោះវត្ថុឈ្មោះ ឃ្លាសកល ត្រូវបានយកមកប្រើនៅក្នុងដែនកំណត់ចារឹកក្នុង។ ដូចនេះការស្វែងរកវត្ថុឈ្មោះ ឃ្លាសកល នេះត្រូវចាប់ផ្តើមនៅក្នុងដែនកំណត់ចារឹកក្នុងនោះមុនគេ រួចបានឡើងទៅដែនកំណត់ចារឹកក្រៅ រួចបានឡើងទៅដែនកំណត់សកល និងចុងក្រោយបង្អស់ត្រូវឡើងទៅដែនកំណត់ទូទៅ។ វត្ថុឈ្មោះ ឃ្លាសកល ត្រូវបានរកឃើញនៅក្នុងដែនកំណត់សកល ព្រោះវាត្រូវបានបង្កើតនៅទីនោះ។

print("ឃ្លាចារឹកក្រៅត្រូវបានរកឃើញនៅក្នុង ", ឃ្លាចារឹកក្រៅ) គឺជាបញ្ជាដែលនៅក្នុងនោះវត្ថុឈ្មោះ ឃ្លាចារឹកក្រៅ ត្រូវបានយកមកប្រើនៅក្នុងដែនកំណត់ចារឹកក្នុង។ ដូចនេះការស្វែងរកវត្ថុឈ្មោះ ឃ្លាចារឹកក្រៅ នេះត្រូវចាប់ផ្តើមនៅក្នុងដែនកំណត់ចារឹកក្នុងនោះមុនគេ រួចបានឡើងទៅដែនកំណត់ចារឹកក្រៅ រួចបានឡើងទៅដែនកំណត់សកល និងចុងក្រោយបង្អស់ត្រូវ

ឡើងទៅដែនកំណត់ទូទៅ។ វត្ថុឈ្មោះ ឃ្លាចារឹកក្រៅ ត្រូវបានរកឃើញនៅក្នុងដែនកំណត់ចារឹកក្រៅ ព្រោះវាត្រូវបានបង្កើតនៅទីនោះ។

print("*ឃ្លាចារឹកក្នុងត្រូវបានរកឃើញនៅក្នុង* ", *ឃ្លាចារឹកក្នុង*) គឺជាបញ្ហាដែលនៅក្នុងនោះវត្ថុឈ្មោះ ឃ្លាចារឹកក្នុង ត្រូវបានយកមកប្រើនៅក្នុងដែនកំណត់ចារឹកក្នុង។ ដូចនេះការស្វែងរកវត្ថុឈ្មោះ ឃ្លាចារឹកក្នុង នេះត្រូវចាប់ផ្តើមនៅក្នុងដែនកំណត់ចារឹកក្នុងនោះមុនគេ រួចបានឡើងទៅដែនកំណត់ចារឹកក្រៅ រួចបានឡើងទៅដែនកំណត់សកល និងចុងក្រោយបង្អស់ត្រូវឡើងទៅដែនកំណត់ទូទៅ។ វត្ថុឈ្មោះ ឃ្លាចារឹកក្នុង ត្រូវបានរកឃើញនៅក្នុងដែនកំណត់ចារឹកក្នុង ព្រោះវាត្រូវបានបង្កើតនៅទីនោះ។

ដោយការស្វែងរកវត្ថុផ្សេងៗត្រូវប្រព្រឹត្តទៅដូចនៅក្នុងគំនូសបំព្រួញខាងលើ ដូចនេះយើងមិនអាចយកវត្ថុនៅក្នុងដែនកំណត់នៅខាងក្រោមមកប្រើនៅក្នុងដែនកំណត់នៅខាងលើបានឡើយ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
ឃ្លាសកល = "ដែនកំណត់សកល"
def ក្បួនក្រៅ() :
    ឃ្លាចារឹកក្រៅ = "ដែនកំណត់ដោយឡែកឬចារឹកក្រៅ"
    print(ឃ្លាចារឹកក្នុង)
    def ក្បួនក្នុង() :
        ឃ្លាចារឹកក្នុង = "ដែនកំណត់ចារឹកក្នុង"

    ក្បួនក្នុង()

ក្បួនក្រៅ()
print(ឃ្លាចារឹកក្រៅ)
```

print(*ឃ្លាចារឹកក្នុង*) គឺជាបញ្ជាតម្រូវឲ្យយកវត្ថុឈ្មោះ ឃ្លាចារឹកក្នុង នៅក្នុងដែនកំណត់ចារឹកក្នុង មកប្រើនៅក្នុងដែនកំណត់ចារឹកក្រៅ។ ប្រការនេះមិនអាចធ្វើទៅបានឡើយ កំហុសមួយបានកើតមានឡើង។

print(*ឃ្លាចារឹកក្រៅ*) គឺជាបញ្ជាតម្រូវឲ្យយកវត្ថុឈ្មោះ ឃ្លាចារឹកក្រៅ នៅក្នុងដែនកំណត់ចារឹកក្រៅមកប្រើនៅក្នុងដែនកំណត់សកល។ ប្រការនេះមិនអាចធ្វើទៅបានឡើយ កំហុសមួយបានកើតមានឡើង។

វត្ថុនៅក្នុងដែនកំណត់ផ្សេងៗគ្នាគឺជាវត្ថុខុសគ្នា ទោះបីជាវត្ថុទាំងនោះមានឈ្មោះដូចគ្នាក៏ដោយ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
object = 1000
print("នៅក្នុងដែនកំណត់សកល វត្ថុឈ្មោះ object គឺជា៖", object)
def ក្បួនក្រៅ() :
    object = True
    print("នៅក្នុងដែនកំណត់ចារឹកក្រៅ វត្ថុឈ្មោះ object គឺជា៖", object)
    def ក្បួនក្នុង() :
        object = "កម្រងអក្សរ"
        print("នៅក្នុងដែនកំណត់ចារឹកក្នុង វត្ថុឈ្មោះ object គឺជា៖", object)
    ក្បួនក្នុង()

ក្បួនក្រៅ()
```

object = 1000 គឺជាបញ្ជាតម្រូវឲ្យបង្កើតវត្ថុឈ្មោះ object មួយនៅក្នុងដែនកំណត់សកល។

object = True គឺជាបញ្ជាតម្រូវឲ្យបង្កើតវត្ថុឈ្មោះ object មួយទៀតនៅក្នុងដែនកំណត់ចារឹកក្រៅ។

object = *"កម្រងអក្សរ"* គឺជាបញ្ជាតម្រូវឲ្យបង្កើតវត្ថុឈ្មោះ object មួយនៅក្នុងដែនកំណត់ចារឹកក្នុង។

នៅក្នុងកម្មវិធីខាងលើ យើងសង្កេតឃើញថា មានវត្ថុមានឈ្មោះថា object ដូចគ្នាចំនួនបីត្រូវបានបង្កើតឡើងនៅក្នុងដែនកំណត់បីខុសៗគ្នា។ វត្ថុទាំងនោះគឺជាវត្ថុខុសៗគ្នា ព្រោះវាត្រូវបានបង្កើតឡើងនៅក្នុងដែនកំណត់ខុសៗគ្នា។ បានន័យថា វត្ថុដែលស្ថិតនៅក្នុងដែនកំណត់ខុសៗគ្នា គឺជាវត្ថុខុសៗគ្នា ទោះបីជាវត្ថុទាំងនោះមានឈ្មោះដូចគ្នាក៏ដោយ។

print("*នៅក្នុងដែនកំណត់សកល វត្ថុឈ្មោះ* object *គឺជា៖*", object) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកវត្ថុឈ្មោះ object មកប្រើនៅក្នុងដែនកំណត់សកល។ អាស្រ័យទៅតាមវិធាននៃការស្វែងរកវត្ថុនៅក្នុងភាសា Python វត្ថុឈ្មោះ object នៅក្នុងដែនកំណត់សកលត្រូវបានយកមកប្រើ។

print("*នៅក្នុងដែនកំណត់ចារឹកក្រៅ វត្ថុឈ្មោះ* object *គឺជា៖*", object) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកវត្ថុឈ្មោះ object មកប្រើនៅក្នុងដែនកំណត់ចារឹកក្រៅ។ អាស្រ័យទៅតាមវិធាននៃការស្វែងរកវត្ថុនៅក្នុងភាសា Python វត្ថុឈ្មោះ object នៅក្នុងដែនកំណត់ចារឹកក្រៅត្រូវបានយកមកប្រើ។

print("*នៅក្នុងដែនកំណត់ចារឹកក្នុង វត្ថុឈ្មោះ* object *គឺជា៖*", object) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកវត្ថុឈ្មោះ object មកប្រើនៅក្នុងដែនកំណត់ចារឹកក្នុង។ អាស្រ័យទៅតាមវិធាននៃការស្វែងរកវត្ថុនៅក្នុងភាសា Python វត្ថុឈ្មោះ object នៅក្នុងដែនកំណត់ចារឹកក្នុងត្រូវបានយកមកប្រើ។

យើងអាចយកវត្ថុនៅក្នុងដែនកំណត់សកលមកប្រើនៅក្នុងដែនកំណត់ដោយឡែកនិងឬនៅក្នុងដែនកំណត់ចារឹកក្នុងបានមែន តែយើងមិនអាចយកឈ្មោះរបស់វត្ថុទាំងនោះទៅភ្ជាប់នឹង

វត្ថុណាផ្សេងទៀតបានឡើយ។ ការប៉ុនប៉ងយកឈ្មោះរបស់វត្ថុទាំងនោះទៅភ្ជាប់នឹងវត្ថុផ្សេងទៀត គឺជាការបង្កើតវត្ថុមានឈ្មោះដូចគ្នានៅក្នុងដែនកំណត់ទាំងនោះ។ ក៏ប៉ុន្តែ បើយើងពិតជាចង់យកឈ្មោះរបស់វត្ថុនៅក្នុងដែនកំណត់សកលទៅភ្ជាប់នឹងវត្ថុផ្សេងទៀតនៅក្នុងដែនកំណត់ដោយឡែកនិងឬដែនកំណត់ចារឹកក្នុង យើងត្រូវប្រើបញ្ជា global នៅខាងមុខឈ្មោះទាំងនោះ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
object = 1000
object1 = [210, False, "ប្រាក់ចំណេញ"]
def ក្បួនក្រៅ() :
  global object
  object = True
  def ក្បួនក្នុង() :
    global object1
    object1 = "កម្រងអក្សរ"
  ក្បួនក្នុង()
ក្បួនក្រៅ()
print("វត្ថុឈ្មោះ object ថ្មីគឺ៖", object)
print("វត្ថុឈ្មោះ object1 ថ្មីគឺ៖", object1)
```

global object គឺជាការប្រើបញ្ជា global ដើម្បីអាចយកឈ្មោះ object របស់វត្ថុនៅក្នុងដែនកំណត់សកលទៅភ្ជាប់នឹងវត្ថុផ្សេងទៀតបាននៅក្នុងដែនកំណត់ដោយឡែកឬចារឹកក្រៅ។

global object1 គឺជាការប្រើបញ្ជា global ដើម្បីអាចយកឈ្មោះ object1 របស់វត្ថុនៅក្នុងដែនកំណត់សកលទៅភ្ជាប់នឹងវត្ថុផ្សេងទៀតបាននៅក្នុងដែនកំណត់ចារឹកក្នុង។

ចំពោះវត្ថុអាចដោះដូរបានដែលស្ថិតនៅក្នុងដែនកំណត់សកល យើងអាចយកវាមកដោះដូរធាតុនៅក្នុងដែនកំណត់ដោយឡែកនិងឬចារឹកក្រៅបានជាធម្មតាដោយមិនចាំបាច់ប្រើបញ្ជា global ។ ពីព្រោះការដោះដូរធាតុនៅក្នុងសមាសវត្ថុមិនមែនជាការយកឈ្មោះរបស់សមាសវត្ថុនោះទៅភ្ជាប់នឹងវត្ថុផ្សេងទៀតឡើយ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
object = [210, False, "ប្រាក់ចំណេញ"]
def ក្បួនក្រៅ() :
    object[0] = True
    def ក្បួនក្នុង() :
        object[1] = "កម្រងអក្សរ"
    ក្បួនក្នុង()

ក្បួនក្រៅ()
print("វត្ថុឈ្មោះ object ថ្មីគឺ៖", object)
```

object[0] = True គឺជាបញ្ជាតម្រូវឲ្យយកកម្រងអថេរឈ្មោះ object នៅក្នុងដែនកំណត់សកលមកដោះដូរធាតុមានលេខរៀង 0 នៅក្នុងដែនកំណត់ចារឹកក្រៅ។

object[1] = "កម្រងអក្សរ" គឺជាបញ្ជាតម្រូវឲ្យយកកម្រងអថេរឈ្មោះ object នៅក្នុងដែនកំណត់សកលមកដោះដូរធាតុមានលេខរៀង 1 នៅក្នុងដែនកំណត់ចារឹកក្នុង។

ដូចគ្នាដែរ យើងអាចយកវត្ថុនៅក្នុងដែនកំណត់ចារឹកក្រៅមកប្រើនៅក្នុងដែនកំណត់ចារឹកក្នុងបានមែន តែយើងមិនអាចយកឈ្មោះរបស់វត្ថុទាំងនោះទៅភ្ជាប់នឹងវត្ថុផ្សេងទៀតបានឡើយ។ ការប៉ុនប៉ងយកឈ្មោះរបស់វត្ថុទាំងនោះទៅភ្ជាប់នឹងវត្ថុផ្សេង គឺជាការបង្កើតវត្ថុមានឈ្មោះដូចគ្នានៅក្នុងដែនកំណត់ចុងក្រោយនេះ។ ក៏ប៉ុន្តែ បើយើងពិតជាចង់ឈ្មោះរបស់វត្ថុនៅក្នុង

ដែនកំណត់ចារឹកក្រៅទៅភ្ជាប់នឹងវត្ថុនៅក្នុងដែនកំណត់ចារឹកក្នុងមែន យើងត្រូវប្រើបញ្ជា nonlocal ដោយធ្វើដូចខាងក្រោមនេះ៖

```
def ក្បួនក្រៅ() :
    object = 1000
    def ក្បួនក្នុង() :
        nonlocal object
        object = "កម្រងអក្សរ"
    ក្បួនក្នុង()
    print("វត្ថុឈ្មោះ object ថ្មីគឺ៖", object)

ក្បួនក្រៅ()
```

nonlocal object គឺជាការប្រើបញ្ជា nonlocal ដើម្បីអាចយកឈ្មោះ object របស់វត្ថុនៅក្នុងដែនកំណត់ចារឹកក្រៅទៅភ្ជាប់នឹងវត្ថុផ្សេងទៀតបាននៅក្នុងដែនកំណត់ចារឹកក្នុង។

ចំពោះវត្ថុដែលស្ថិតនៅក្នុងដែនកំណត់ដោយឡែកដែលជាក្បួនខុសៗគ្នា គឺជាវត្ថុខុសៗគ្នា ទោះបីជាវត្ថុទាំងនោះមានឈ្មោះដូចគ្នាក៏ដោយ។ ពីព្រោះក្បួននិមួយៗគឺជាដែនកំណត់ដោយឡែកខុសៗគ្នា។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
def រកប្រាក់ចំណេញ(ថ្លៃលក់, ថ្លៃទិញ) :
    ប្រាក់ចំណេញ = ថ្លៃលក់ - ថ្លៃទិញ
    return ប្រាក់ចំណេញ

def សរុបទឹកប្រាក់(ថ្លៃលក់, ថ្លៃទិញ) :
    ទឹកប្រាក់ = ថ្លៃលក់ + ថ្លៃទិញ + រកប្រាក់ចំណេញ(ថ្លៃលក់=ថ្លៃលក់, ថ្លៃទិញ=ថ្លៃទិញ)
    print(ទឹកប្រាក់)
```

សរុបទឹកប្រាក់(1000, 900)

រកប្រាក់ចំណេញ(ថ្លៃលក់=ថ្លៃលក់, ថ្លៃទិញ=ថ្លៃទិញ) គឺជាការយកក្បួនឈ្មោះ រកប្រាក់ចំណេញ មកប្រើជាបញ្ជានៅក្នុងក្បួនឈ្មោះ សរុបទឹកប្រាក់ ។ ដំណាង ថ្លៃលក់ និង ថ្លៃទិញ របស់ក្បួនឈ្មោះ សរុបទឹកប្រាក់ ត្រូវបានផ្តល់ជាដំណឹងរៀងគ្នាឲ្យទៅដំណាង ថ្លៃលក់ និង ថ្លៃទិញ នៅក្នុងក្បួនឈ្មោះ រកប្រាក់ចំណេញ ។

ដូចនេះយើងឃើញថា ក្បួនឈ្មោះ រកប្រាក់ចំណេញ និងក្បួនឈ្មោះ សរុបទឹកប្រាក់ សុទ្ធតែមានដំណាងជាឈ្មោះ ថ្លៃលក់ និង ថ្លៃទិញ ដូចគ្នា តែឈ្មោះទាំងនោះជាឈ្មោះរបស់វត្ថុខុសៗគ្នា ព្រោះវាស្ថិតនៅក្នុងដែនកំណត់ដោយឡែកពីរផ្សេងគ្នា។

សរុបមក ឈ្មោះរបស់វត្ថុនៅក្នុងភាសា Python ក៏ដូចជាឈ្មោះរបស់មនុស្សយើងដែរ។ ពោលគឺមនុស្សឈ្មោះ វុធ នៅក្នុងគ្រួសារ ក ខុសពីមនុស្សឈ្មោះ វុធ នៅក្នុងគ្រួសារ ខ ទោះបីជាមនុស្សទាំងពីរនាក់នោះមានឈ្មោះដូចគ្នាក៏ដោយ។

ក្បួនមានស្រាប់

ក្បួនមានស្រាប់ (built-in function) គឺជាក្បួនទាំងឡាយដែលត្រូវបានបង្កើតឡើងរួចជាស្រេចទុកនៅក្នុងដែនកំណត់ទូទៅ។ យើងអាចយកក្បួនមានស្រាប់ទៅប្រើនៅក្នុងដែនកំណត់ណាក៏បានដែរ។ ក្បួនមានស្រាប់សំខាន់ៗមានដូចតទៅនេះ៖

abs(…)

abs(លេខ)-> លេខ

គឺជាក្បួនមានស្រាប់ប្រើសម្រាប់រកតម្លៃដាច់ខាតនៃចំនួនណាមួយ។

```
a = -136
```

```
print("តម្លៃដាច់ខាតរបស់ a គឺ៖", abs(a))
```

divmod(…)

divmod(លេខ, លេខ) -> (ផលចែក, សំណល់)

គឺជាក្បួនមានស្រាប់ប្រើសម្រាប់គណនារកផលចែកនិងសំណល់រវាងចំនួនពីរ។

```
a = 10.5
b = 3.14
print(divmod(a, b))
```

input(…)

input([prompt])-> កម្រងអក្សរ

គឺជាក្បួនមានស្រាប់ប្រើសម្រាប់ចម្លងយកកម្រងអក្សរទាំងឡាយណាដែលត្រូវបានសរសេរនៅលើបង្អួចបឋម។

```
ឃ្លា = input("ចូរសរសេរអ្វីមួយនៅទីនេះ ")
print(ឃ្លា)
```

len(…)

len(សមាសវត្ថុ)-> ចំនួនគត់

គឺជាក្បួនមានស្រាប់ប្រើសម្រាប់រាប់ធាតុឬកូនសោរនៅក្នុងសមាសវត្ថុណាមួយ។

```
កម្រងចម្រុះ = [100, 1.5, "ប្រាក់ចំណេញ", True]
សំណុំចម្រុះ = {"ឈ្មោះ":"កុសល", "អាយុ":35, "ភេទ":"ប្រុស"}
print(len(កម្រងចម្រុះ))
print(len(សំណុំចម្រុះ))
```

max(...)

max(សមាសវត្ថុ)-> ធាតុឬកូនសោរធំជាងគេ

max(a, b, c)-> វត្ថុធំជាងគេ

គឺជាក្បួនមានស្រាប់ប្រើសម្រាប់រកធាតុឬកូនសោរឬវត្ថុដែលធំជាងគេបំផុត។

```
កម្រងថេរ = [100, 1.5, 3.14, 1000]
សំណុំថេរ = {"ឈ្មោះ":"កុសល", "អាយុ":35, "ភេទ":"ប្រុស"}
print(max(កម្រងថេរ))
print(max(សំណុំថេរ))
print(max(1, 0.5, 15))
```

min(...)

min(សមាសវត្ថុ)-> ធាតុឬកូនសោរតូចជាងគេ

min(a, b, c)-> វត្ថុតូចជាងគេ

គឺជាក្បួនមានស្រាប់ប្រើសម្រាប់រកធាតុឬកូនសោរដែលតូចជាងគេបំផុត។

```
កម្រងថេរ = [100, 1.5, 3.14, 1000]
សំណុំថេរ = {"ឈ្មោះ":"កុសល", "អាយុ":35, "ភេទ":"ប្រុស"}
print(min(កម្រងថេរ))
print(min(សំណុំថេរ))
print(min(1, 0.5, 15))
```

pow(...)

pow(លេខ, លេខ)-> លេខ

គឺជាក្បួនមានស្រាប់ដែលសមមូលទៅនឹងប្រមាណវិធីស្វ័យគុនដោយប្រើប្រមាណសញ្ញា ** ។

```
print(pow(3, 3))
```

print(…)

print(វត្ថុ, …, វត្ថុ)

គឺជាក្បួនមានស្រាប់ប្រើសម្រាប់សរសេរវត្ថុផ្សេងៗនៅលើបង្អួចបឋម។

```
print(100, "ប្រាក់ចំណេញ", True)
```

round(…)

round(ចំនួនពិត)-> ចំនួនគត់

គឺជាក្បួនមានស្រាប់ប្រើសម្រាប់កែចំនួនពិតឲ្យទៅជាចំនួនគត់។

```
ថ្លៃលក់ = 1000.33
print(round(ថ្លៃលក់))
```

sorted(…)

sorted(សមាសវត្ថុ)-> សមាសវត្ថុ

គឺជាក្បួនមានស្រាប់ប្រើសម្រាប់ចម្លងយកកម្រងឬកូនសោរនៅក្នុងវចនានុក្រមណាមួយមកបង្កើតជាកម្រងអថេរថ្មីមួយទៀតមានធាតុដែលត្រូវបានតម្រៀបតាមលំដាប់ថ្នាក់ពីតូចទៅធំ។

```
កម្រងតម្លៃ = [100, 0.5, 1.34, 200]
សំណុំចម្រុះ = {"c":100, "b":"Python", "a":True}
```

```
print(sorted(កម្រងតម្លៃ))
print(sorted(សំណុំចម្រុះ))
```

sum(...)

sum(សមាសវត្ថុ)-> ផលបូកនៃធាតុទាំងអស់

គឺជាក្បួនមានស្រាប់ប្រើសម្រាប់ចម្លងយកធាតុឬកូនសោរទាំងអស់នៅក្នុងសមាសវត្ថុណាមួយមកបូកបញ្ចូលគ្នាបង្កើតជាលេខថ្មីមួយទៀត។

```
កម្រងតម្លៃ = (100, 0.5, 1.34, 200)
សំណុំចម្រុះ = {1:100, 2:"Python", 3:True}
print(sum(កម្រងតម្លៃ))
print(sum(សំណុំចម្រុះ))
```

ថ្នាក់

ការបង្កើត ថ្នាក់

ថ្នាក់ (class) គឺជាវត្ថុដែលជាកន្លែងមួយសម្រាប់បង្កើតវត្ថុផ្សេងៗទៀតទុកសម្រាប់យកទៅប្រើនៅពេលក្រោយទៀត។ ដើម្បីបង្កើតថ្នាក់ យើងត្រូវសរសេរកម្មវិធីដូចខាងក្រោមនេះ៖

```
class ក្រឡាផ្ទៃ() :
  ពី = 3.14
  def ក្រឡាផ្ទៃចតុកោណកែង(សិស្ស, បណ្តោយ, ទទឹង) :
    ផ្ទៃក្រឡា = បណ្តោយ * ទទឹង
    print("ក្រឡាផ្ទៃចតុកោណកែងគឺ៖", ផ្ទៃក្រឡា)
```

class *ក្រឡាផ្ទៃ*() : គឺជាការប្រើបញ្ជា class តម្រូវឲ្យបង្កើតវត្ថុដែលជាថ្នាក់មានឈ្មោះថា ក្រឡាផ្ទៃ ។

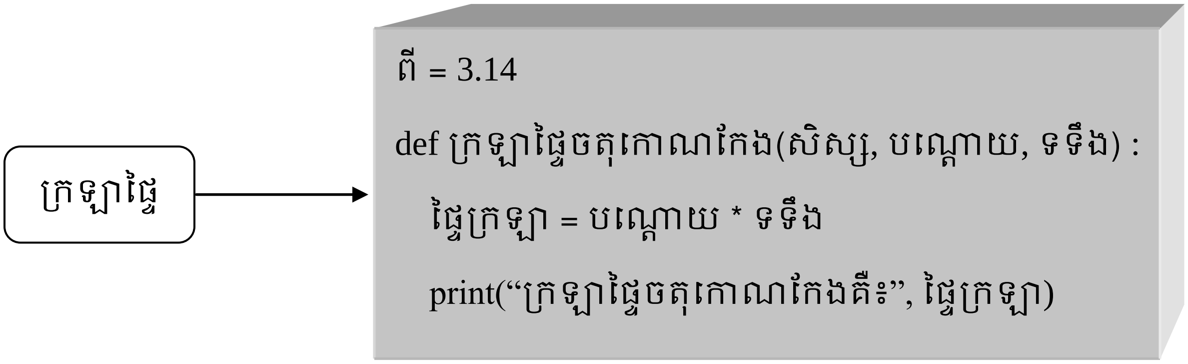
ក្រោយពីថ្នាក់ឈ្មោះ ក្រឡាផ្ទៃ ត្រូវបានបង្កើតឡើងនៅក្នុងសតិរបស់កំព្យូទ័ររួចមក ក្រុមបញ្ជានៅក្នុងថ្នាក់នោះត្រូវបានយកទៅអនុវត្តភ្លាមមួយរំពេច ដែលជាប្រការធ្វើឲ្យចំនួនពិតឈ្មោះ ពី និងក្បួនឈ្មោះ ក្រឡាផ្ទៃចតុកោណកែង ក៏ត្រូវបានបង្កើតទុកនៅក្នុងថ្នាក់នោះដែរ។

ដូចនេះថ្នាក់ខុសពីក្បួននៅត្រង់ថា ក្រុមបញ្ជានៅក្នុងថ្នាក់ត្រូវយកទៅអនុវត្តភ្លាមមួយរំពេចបន្ទាប់ពីថ្នាក់ត្រូវបានបង្កើតរួចហើយ។ ចំណែកឯក្រុមបញ្ជានៅក្នុងក្បួនវិញ មិនទាន់ត្រូវយកទៅអនុវត្តភ្លាមទេនៅពេលដែលក្បួនត្រូវបានបង្កើឡើងនោះ។ ក្រុមបញ្ជានៅក្នុងក្បួនត្រូវយកទៅអនុវត្តតែនៅពេលណាដែលក្បួនត្រូវយកទៅប្រើតែប៉ុណ្ណោះ។

ក្រុមបញ្ជានៅក្នុងថ្នាក់អាចជាបញ្ជាប្រភេទណាក៏បានដែរ វាអាចជាបញ្ជាតម្រូវឲ្យបង្កើតវត្ថុផ្សេងៗទៀត ឬជាបញ្ជាតម្រូវឲ្យអនុវត្តបញ្ជាមួយចំនួនទៀត។ ក៏ប៉ុន្តែ ភាគច្រើនគេនិយមប្រើប្រភេទនៃបញ្ជាដែលតម្រូវឲ្យបង្កើតវត្ថុដែលជាទិន្នន័យនិងឬក្បួនមួយចំនួនទុកនៅក្នុងថ្នាក់សម្រាប់យកទៅប្រើនៅពេលក្រោយៗទៀត។

គ្រប់វត្ថុទាំងឡាយដែលត្រូវបានបង្កើតទុកនៅក្នុងថ្នាក់ ត្រូវចាត់ទុកថាជា *សម្បត្តិថ្នាក់* (class attribute) ដែលអាចជាក្បួននិងវត្ថុដែលជាទិន្នន័យផ្សេងៗ។ ក្បួននៅក្នុងថ្នាក់ត្រូវហៅថា *វិធី* (method) ចំណែកឯវត្ថុដែលជាទិន្នន័យនៅក្នុងថ្នាក់ត្រូវហៅថា *ទិន្នន័យគំរូ* (data attribute) ។

ដូចនេះនៅក្នុងថ្នាក់ឈ្មោះ ក្រឡាផ្ទៃ វត្ថុឈ្មោះ ពី គឺជាទិន្នន័យគំរូ និងក្បួនឈ្មោះ ក្រឡាផ្ទៃចតុកោណកែង គឺជាវិធី។ ម៉្យាទៀតយើងត្រូវធ្វើការកត់សំគាល់ផងដែរថា វិធីដែលជាក្បួននៅក្នុងថ្នាក់ ត្រូវតែមានដំណាងលើសមួយ ទោះបីជាយើងមិនត្រូវការយកវាទៅប្រើការក៏ដោយ។ យើងនឹងដឹងពីតួនាទីនៃដំណាងទីមួយនេះនៅពេលបន្តិចទៀតនេះ។



ដើម្បីឲ្យដឹងថាតើនៅក្នុងថ្នាក់ឈ្មោះ ក្រឡាផ្ទៃ មានអ្វីខ្លះនោះ យើងត្រូវធ្វើដូចខាងក្រោមនេះ៖

```
class ក្រឡាផ្ទៃ() :
    ពី = 3.14
    def ក្រឡាផ្ទៃចតុកោណកែង(សិស្ស, បណ្តោយ, ទទឹង) :
```

```
        ផ្ទៃក្រឡា = បណ្តោយ * ទទឹង
        print("ក្រឡាផ្ទៃចតុកោណកែងគឺ៖", ផ្ទៃក្រឡា)

help(ក្រឡាផ្ទៃ)
```

help(ក្រឡាផ្ទៃ) គឺជាបញ្ជាតម្រូវឲយកថ្នាក់ឈ្មោះ ក្រឡាផ្ទៃ មកពិនិត្យមើលថាតើមានអ្វីខ្លះនៅក្នុងនោះ។ ជាលទ្ធផល យើងឃើញថានៅក្នុងថ្នាក់ឈ្មោះ ក្រឡាផ្ទៃ មានទិន្នន័យគំរូឈ្មោះ ពី មួយនិងវិធីឈ្មោះ ក្រឡាផ្ទៃចតុកោណកែង មួយ។

ដោយថ្នាក់ក៏ជាវត្ថុមួយដូចជាវត្ថុដទៃទៀតដែរ ដូចនេះយើងអាចឈ្មោះរបស់ថ្នាក់ទៅភ្ជាប់នឹងវត្ថុដទៃទៀតបានដោយគ្មានបញ្ហាអ្វីឡើយ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
class ក្រឡាផ្ទៃ() :
    ពី = 3.14
    def ក្រឡាផ្ទៃចតុកោណកែង(សិស្ស, បណ្តោយ, ទទឹង) :
        ផ្ទៃក្រឡា = បណ្តោយ * ទទឹង
        print("ក្រឡាផ្ទៃចតុកោណកែងគឺ៖", ផ្ទៃក្រឡា)

ក្រឡាផ្ទៃ = 1000
print(ក្រឡាផ្ទៃ)
```

ក្រឡាផ្ទៃ = 1000 គឺជាបញ្ជាតម្រូវឲ្យយកឈ្មោះរបស់ថ្នាក់ដែលមានឈ្មោះថា ក្រឡាផ្ទៃ ទៅភ្ជាប់នឹងចំនួនគត់លេខ 1000 ។ កត្តានេះធ្វើឲ្យថ្នាក់នោះក្លាយជាវត្ថុគ្មានឈ្មោះនិងត្រូវបានលុបចេញពីក្នុងសតិរបស់កំព្យូទ័រដោយយន្តការបោសសម្អាត។

ដោយថ្នាក់ក៏ជាវត្ថុមួយដូចជាវត្ថុដទៃទៀតដែរ ដូចនេះយើងអាចយកឈ្មោះជាច្រើនទៅភ្ជាប់នឹងថ្នាក់ណាមួយដូចជាការយកឈ្មោះជាច្រើនទៅភ្ជាប់នឹងវត្ថុដទៃទៀតដែរ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
class ក្រឡាផ្ទៃ() :
    ពី = 3.14
    def ក្រឡាផ្ទៃចតុកោណកែង(សិស្ស, បណ្តោយ, ទទឹង) :
        ផ្ទៃក្រឡា = បណ្តោយ * ទទឹង
        print("ក្រឡាផ្ទៃចតុកោណកែងគឺ៖", ផ្ទៃក្រឡា)

surface = area = ក្រឡាផ្ទៃ
print(surface)
print(area)
print(ក្រឡាផ្ទៃ)
```

surface = area = *ក្រឡាផ្ទៃ* គឺជាបញ្ជាតម្រូវឲ្យភ្ជាប់ឈ្មោះ surface និង area ទៅនឹងវត្ថុដែលជាថ្នាក់មានឈ្មោះថា ក្រឡាផ្ទៃ ។

ការយកថ្នាក់មកប្រើ

ការយកថ្នាក់មកប្រើ (calling a class) គឺជាទង្វើមួយស្រដៀងនឹងការយកក្បួនមកប្រើដែរ គឺត្រូវធ្វើឡើងដូចខាងក្រោមនេះ៖

```
class ក្រឡាផ្ទៃ() :
    ពី = 3.14
    def ក្រឡាផ្ទៃចតុកោណកែង(សិស្ស, បណ្តោយ, ទទឹង) :
        ផ្ទៃក្រឡា = បណ្តោយ * ទទឹង
```

```
        print("ក្រឡាផ្ទៃចតុកោណកែង៖", ផ្ទៃក្រឡា)

    លទ្ធផល = ក្រឡាផ្ទៃ()
    print(លទ្ធផល)
```

លទ្ធផល = ក្រឡាផ្ទៃ() គឺជាការយកថ្នាក់ឈ្មោះ ក្រឡាផ្ទៃ មកប្រើដោយភ្ជាប់ឈ្មោះ លទ្ធផល ទៅនឹងវត្ថុដែលត្រូវបានបង្កើតឡើងដោយសារការយកថ្នាក់នោះមកប្រើ។

ដូចនេះយើងឃើញថា ការយកថ្នាក់មកប្រើ ពុំមែនជាការយកសម្បត្តិទាំងឡាយដែលមាននៅក្នុងថ្នាក់មកប្រើឡើយ ផ្ទុយទៅវិញ វាគឺជាការបង្កើតវត្ថុម៉្យាងនៅក្នុងសតិរបស់កំព្យូទ័រ។

print(លទ្ធផល) គឺជាបញ្ជាតម្រូវឲ្យសរសេរវត្ថុឈ្មោះ លទ្ធផល ដែលជាវត្ថុបានមកពីការយកថ្នាក់ឈ្មោះ ក្រឡាផ្ទៃ មកប្រើ។

ដូចនេះការយកថ្នាក់ឈ្មោះ ក្រឡាផ្ទៃ មកប្រើបណ្តាលឲ្យវត្ថុម៉្យាងត្រូវបង្កើតឡើង។ វត្ថុនោះត្រូវហៅថា *សិស្ស* (instance) ។ ជាទូទៅ គ្រប់ការយកថ្នាក់ណាមួយមកប្រើ បណ្តាលឲ្យសិស្សម្នាក់ត្រូវបានបង្កើតឡើង។ យើងអាចនិយាយបានម៉្យាងទៀតថា ការយកថ្នាក់មកប្រើ គឺជាការបង្កើតសិស្សនៃថ្នាក់នោះ (instantiation) ។

ក្រៅពីការបង្កើតសិស្សម្នាក់ចេញពីថ្នាក់ណាមួយ យើងបង្កើតសិស្សជាច្រើនរាប់មិនអស់ចេញពីថ្នាក់នោះដោយយកថ្នាក់នោះមកប្រើជាច្រើនលើកច្រើនសារ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ

```
class ក្រឡាផ្ទៃ() :
  ពី = 3.14
  def ក្រឡាផ្ទៃចតុកោណកែង(សិស្ស, បណ្តោយ, ទទឹង) :
    ផ្ទៃក្រឡា = បណ្តោយ * ទទឹង
    print("ក្រឡាផ្ទៃចតុកោណកែង៖", ផ្ទៃក្រឡា)
```

```
សិស្សក = ក្រឡាផ្ទៃ()
សិស្សខ = ក្រឡាផ្ទៃ()
សិស្សគ = ក្រឡាផ្ទៃ()
print(សិស្សក)
print(សិស្សខ)
print(សិស្សគ)
```

សិស្សក = ក្រឡាផ្ទៃ() គឺជាការយកថ្នាក់ឈ្មោះ ក្រឡាផ្ទៃ មកប្រើដើម្បីបង្កើតសិស្សឈ្មោះ សិស្សក ម្នាក់។

សិស្សខ = ក្រឡាផ្ទៃ() គឺជាការយកថ្នាក់ឈ្មោះ ក្រឡាផ្ទៃ មកប្រើដើម្បីបង្កើតសិស្សឈ្មោះ សិស្សខ ម្នាក់ទៀត។

សិស្សគ = ក្រឡាផ្ទៃ() គឺជាការយកថ្នាក់ឈ្មោះ ក្រឡាផ្ទៃ មកប្រើដើម្បីបង្កើតសិស្សឈ្មោះ សិស្សគ ម្នាក់ទៀត។

ការយកសម្បត្តិថ្នាក់មកប្រើ

យើងបានដឹងរួចមកហើយថា ការយកថ្នាក់មកប្រើមិនមែនជាការយកសម្បត្តិនៅក្នុងថ្នាក់មកប្រើនោះទេ វាគឺជាការបង្កើតសិស្សនៃថ្នាក់នោះ។ ដើម្បីយកសម្បត្តិនៅក្នុងថ្នាក់មកប្រើ យើងត្រូវធ្វើដូចខាងក្រោមនេះ៖

```
class ក្រឡាផ្ទៃ() :
    ពី = 3.14
    def ក្រឡាផ្ទៃចតុកោណកែង(សិស្ស, បណ្តោយ, ទទឹង) :
```

```
        ផ្ទៃក្រឡា = បណ្តោយ * ទទឹង
        print("ក្រឡាផ្ទៃចតុកោណកែងគឺៈ", ផ្ទៃក្រឡា)

សិស្សក = ក្រឡាផ្ទៃ()
print(ក្រឡាផ្ទៃ.ពី)
print(សិស្សក.ពី)
ក្រឡាផ្ទៃ.ក្រឡាផ្ទៃចតុកោណកែង(សិស្សក, 25, 5)
សិស្សក.ក្រឡាផ្ទៃចតុកោណកែង(25, 5)
```

print(ក្រឡាផ្ទៃ.ពី) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកទិន្នន័យគំរូឈ្មោះ ពី មកប្រើតាមរយៈថ្នាក់ឈ្មោះ ក្រឡាផ្ទៃ ដែលជាថ្នាក់របស់ទិន្នន័យគំរូនោះ។

print(សិស្សក.ពី) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកទិន្នន័យគំរូឈ្មោះ ពី មកប្រើតាមរយៈសិស្សឈ្មោះ សិស្សក ដែលជាសិស្សនៃថ្នាក់របស់ទិន្នន័យគំរូនោះ។

ក្រឡាផ្ទៃ.ក្រឡាផ្ទៃចតុកោណកែង(សិស្សក, 25, 5) គឺជាបញ្ជាតម្រូវឲ្យយកវិធីឈ្មោះ ក្រឡាផ្ទៃចតុកោណកែង មកប្រើតាមរយៈថ្នាក់ឈ្មោះ ក្រឡាផ្ទៃ ដែលជាថ្នាក់របស់វិធីនោះ។ ដំណឹងដែលត្រូវផ្តល់ឲ្យទៅវិធីនោះគឺជាសិស្សឈ្មោះ សិស្សក លេខ 25 និងលេខ 5 ។

សិស្សក.ក្រឡាផ្ទៃចតុកោណកែង(25, 5) គឺជាបញ្ជាតម្រូវឲ្យយកវិធីឈ្មោះ ក្រឡាផ្ទៃចតុកោណកែង មកប្រើតាមរយៈសិស្សឈ្មោះ សិស្សក ដែលជាសិស្សនៃថ្នាក់របស់វិធីនោះ។

ដូចនេះយើងឃើញថា ដើម្បីអាចយកវិធីនៅក្នុងថ្នាក់ណាមួយមកប្រើបាន យើងចាំបាច់ត្រូវតែបង្កើតសិស្សនៃថ្នាក់នោះជាមុនសិន រួចសឹមយកសម្បត្តិទាំងនោះមកប្រើតាមរយៈថ្នាក់ឬតាមរយៈសិស្សរបស់ថ្នាក់នោះ។ បានន័យថាយើងមិនអាចយកសម្បត្តិនៅក្នុងថ្នាក់មកប្រើ

ដោយផ្ទាល់បានឡើយ។ មួយវិញទៀត ចំពោះការយកវិធីមកប្រើតាមរយៈថ្នាក់របស់វា យើងចាំបាច់ត្រូវផ្តល់ដំណឹងជាសិស្សណាម្នាក់សម្រាប់ដំណាងទីមួយ។ តែចំពោះការយកវិធីមកប្រើតាមរយៈសិស្សនៃថ្នាក់របស់វិធីនោះវិញ យើងមិនចាំបាច់ផ្តល់ដំណឹងជាសិស្សណាម្នាក់សម្រាប់ដំណាងទីមួយឡើយ។

ការបន្ថែមវត្ថុចូលទៅក្នុងថ្នាក់

ក្រៅពីការយកសម្បត្តិនៅក្នុងថ្នាក់មកប្រើ យើងអាចបន្ថែមវត្ថុផ្សេងៗទៀតចូលទៅក្នុងថ្នាក់ថែមពីលើសម្បត្តិមាននៅក្នុងថ្នាក់។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
def ផ្ទៃវង្វង់(សិស្ស, កាំ) :
    S = កាំ * កាំ * 3.14
    print("ក្រឡាផ្ទៃរង្វង់គឺ៖", S)

កម្រងវិមាត្រ = [25, 5]

class ក្រឡាផ្ទៃ() :
    ពី = 3.14
    def ក្រឡាផ្ទៃចតុកោណកែង(សិស្ស, បណ្តោយ, ទទឹង) :
        ផ្ទៃក្រឡា = បណ្តោយ * ទទឹង
        print("ក្រឡាផ្ទៃចតុកោណកែងគឺ៖", ផ្ទៃក្រឡា)

ក្រឡាផ្ទៃ.ក្រឡាផ្ទៃរង្វង់ = ផ្ទៃរង្វង់
ក្រឡាផ្ទៃ.វិមាត្រ = កម្រងវិមាត្រ
help(ក្រឡាផ្ទៃ)
```

ក្រឡាផ្ទៃ.ក្រឡាផ្ទៃរង្វង់ = ផ្ទៃរង្វង់ គឺជាបញ្ជាតម្រូវឲ្យបន្ថែមក្បូនឈ្មោះ ផ្ទៃរង្វង់ មួយទៀតចូលទៅក្នុងថ្នាក់ឈ្មោះ ក្រឡាផ្ទៃ ។ ក្បូនឈ្មោះ ផ្ទៃរង្វង់ ក្លាយទៅជាវិធីឈ្មោះ ក្រឡាផ្ទៃរង្វង់ នៅក្នុងថ្នាក់ឈ្មោះ ក្រឡាផ្ទៃ ។

ក្រឡាផ្ទៃ.វិមាត្រ = កម្រងវិមាត្រ គឺជាបញ្ជាតម្រូវឲ្យបន្ថែមកម្រងអថេរឈ្មោះ កម្រងវិមាត្រ មួយទៀតចូលទៅក្នុងថ្នាក់ឈ្មោះ ក្រឡាផ្ទៃ ។ កម្រងអថេរឈ្មោះ កម្រងវិមាត្រ ក្លាយទៅជាទិន្នន័យគំរូឈ្មោះ វិមាត្រ នៅក្នុងថ្នាក់ឈ្មោះ ក្រឡាផ្ទៃ ។

ស្ថាបនិក

ស្ថាបនិក (constructor) គឺជាវិធីដែលមានឈ្មោះថា __init__ (សញ្ញា _ ជាប់គ្នាពីរនៅពីមុខនិងនៅពីក្រោយឈ្មោះ init) ។ នៅក្នុងភាសា Python បើសិនជាយើងបង្កើតវិធីមួយមានឈ្មោះថា __init__ នៅក្នុងថ្នាក់ណាមួយ វិធីនោះនឹងត្រូវយកទៅប្រើជាស្វ័យប្រវត្តិនៅពេលដែលយើងយកថ្នាក់នោះទៅប្រើដើម្បីបង្កើតសិស្ស។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
class ក្រឡាផ្ទៃ() :
  ពី = 3.14
  def __init__(សិស្ស) :
    print("ស្ថាបនិកត្រូវបានយកទៅប្រើ។")

  def ក្រឡាផ្ទៃចតុកោណកែង(សិស្ស, បណ្តោយ, ទទឹង) :
    ផ្ទៃក្រឡា = បណ្តោយ * ទទឹង
    print("ក្រឡាផ្ទៃចតុកោណកែងគឺ៖", ផ្ទៃក្រឡា)

សិស្សក = ក្រឡាផ្ទៃ()
```

def __init__(*សិស្ស*) : គឺជាបញ្ជាតម្រូវឲ្យបង្កើតស្ថាបនិកមួយដែលជាវិធីមានឈ្មោះថា __init__ ។

សិស្សក = ក្រឡាផ្ទៃ() គឺជាការយកថ្នាក់ឈ្មោះ ក្រឡាផ្ទៃ មកប្រើដើម្បីបង្កើតសិស្សឈ្មោះ សិស្សក ម្នាក់។ ប្រការនេះធ្វើឲ្យស្ថាបនិកដែលជាវិធីមានឈ្មោះថា __init__ នៅក្នុងថ្នាក់នោះត្រូវយកមកប្រើជាស្វ័យប្រវត្តិ។

ដូចនេះយើងឃើញថា ការយកថ្នាក់មកប្រើគឺជាបញ្ជាតម្រូវឲ្យធ្វើការងារពីរគឺ៖ បង្កើតសិស្ស និងយកស្ថាបនិកមកប្រើបើសិនជាមាន។ ហើយយើងនឹងបានដឹងពីសារៈសំខាន់របស់ស្ថាបនិកនៅពេលបន្តិចទៀតនេះ។

វិធី

វិធីក៏ជាក្បួនដូចជាក្បួនដទៃទៀតនៅខាងក្រៅថ្នាក់ដែរ តែវិធីខុសពីក្បួនទាំងនោះនៅត្រង់ថា ដំណាងនៅក្នុងវិធីត្រូវតែមានចំនួនលើសចំនួនដំណឹងមួយជានិច្ច។ បានន័យថា បើវិធីត្រូវការដំណឹងចំនួន n ដំណាងនៅក្នុងវិធីនោះត្រូវតែមានចំនួន n + 1 ហើយដំណាងដែលលើសមួយនោះ ត្រូវតែជាដំណាងដែលនៅខាងដើមគេបំផុត។ ការទាមទារឲ្យមានដំណាងលើសមួយនៅក្នុងវិធីទាំងឡាយ គឺដោយហេតុថា នៅពេលដែលវិធីត្រូវយកទៅប្រើតាមរយៈសិស្ស វិធីនឹងទទួលបានដំណឹងដែលជាសិស្សនោះជាស្វ័យប្រវត្តិបន្ថែមទៅលើដំណឹងផ្សេងទៀត។ ហើយដំណឹងដែលជាសិស្សនោះនឹងត្រូវផ្តល់ឲ្យទៅដំណាងនៅខាងដើមគេបំផុត។ ដូចនេះបើនៅក្នុងវិធីគ្មានដំណាងលើសមួយសម្រាប់ទទួលយកដំណឹងដែលជាសិស្សនោះទេ ដំណាងនៅខាងដើមគេបំផុតនឹងត្រូវយកមកប្រើសម្រាប់ទទួលយកសិស្សនោះ។ ប្រការនេះនឹងធ្វើឲ្យកំហុសស្តីពីការមានដំណាងមិនគ្រប់គ្រាន់នឹងកើតមានឡើង។ ហេតុដូចនេះហើយបានជា

ចាំបាច់យើងត្រូវបង្កើតដំណាងលើសមួយជានិច្ចនៅក្នុងវិធីផ្សេងៗដើម្បីចាំទទួលយកដំណឹង ដែលជាសិស្សនោះ។ ជាទូទៅ គេច្រើនប្រើឈ្មោះ self ជាដំណាងដែលចាំទទួលយកសិស្ស តែយើងក៏អាចប្រើឈ្មោះដទៃទៀតបានដែរ។ ជាភាសាខ្មែរ យើងគួរតែប្រើពាក្យថា សិស្ស ជាដំណាងទីមួយនោះ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
class ក្រឡាផ្ទៃ() :
    ពី = 3.14
    def __init__(សិស្ស) :
        print("ស្ថាបនិកត្រូវបានយកទៅប្រើ។")

    def ក្រឡាផ្ទៃចតុកោណកែង(សិស្ស, បណ្តោយ, ទទឹង) :
        ផ្ទៃក្រឡា = បណ្តោយ * ទទឹង
        print("ក្រឡាផ្ទៃចតុកោណកែងគឺ៖", ផ្ទៃក្រឡា)

សិស្សក = ក្រឡាផ្ទៃ()
សិស្សក.ក្រឡាផ្ទៃចតុកោណកែង(25, 5)
```

def __init__(សិស្ស) : គឺជាបញ្ជាតម្រូវឲ្យបង្កើតស្ថាបនិកដែលនៅក្នុងនោះមានពាក្យថា សិស្ស ជាដំណាងទីមួយសម្រាប់ទទួលយកដំណឹងដែលជាសិស្សដែលនឹងត្រូវបង្កើតឡើងនៅពេលដែលថ្នាក់ត្រូវយកទៅប្រើ។

def ក្រឡាផ្ទៃចតុកោណកែង(សិស្ស, បណ្តោយ, ទទឹង) : គឺជាបញ្ជាតម្រូវឲ្យបង្កើតវិធីឈ្មោះ ក្រឡាផ្ទៃចតុកោណកែង មួយដែលនៅក្នុងនោះមានពាក្យថា សិស្ស ជាដំណាងទីមួយសម្រាប់ ដំណឹងដែលនឹងជាសិស្សណាម្នាក់។

សិស្សក = ក្រឡាផ្ទៃ() គឺជាការយកថ្នាក់ឈ្មោះ ក្រឡាផ្ទៃ មកប្រើដើម្បីបង្កើតសិស្សឈ្មោះ សិស្សក ម្នាក់។ ប្រការនេះធ្វើឲ្យស្ថាបនិកនៃថ្នាក់នោះត្រូវយកមកប្រើជាស្វ័យប្រវត្តិ។ ហើយនៅពេលដែលស្ថាបនិកត្រូវយកមកប្រើ ដំណឹងដែលជាសិស្សទើបនឹងត្រូវបានបង្កើតមានឈ្មោះថា សិស្សក នោះត្រូវបានផ្តល់ឲ្យស្ថាបនិកសម្រាប់ដំណាង សិស្ស ជាស្វ័យប្រវត្តិ។

សិស្សក.ក្រឡាផ្ទៃចតុកោណកែង(25, 5) គឺជាការយកវិធីឈ្មោះ ក្រឡាផ្ទៃចតុកោណកែង មកប្រើតាមរយៈសិស្សឈ្មោះ សិស្សក ។ ប្រការនេះធ្វើឲ្យដំណឹងដែលជាសិស្សឈ្មោះ សិស្សក ត្រូវបានផ្តល់ជាស្វ័យប្រវត្តិបន្ថែមលើដំណឹងដែលជាលេខ 25 និងលេខ 5 ឲ្យទៅវិធីនោះ។

ដូចនេះយើងឃើញថា នៅពេលដែលវិធីត្រូវយកទៅប្រើតាមរយៈសិស្សណាមួយ ដំណឹងដែលជាសិស្សនោះ ត្រូវផ្តល់ឲ្យទៅវិធីនោះជាស្វ័យប្រវត្តិបន្ថែមទៅលើដំណឹងផ្សេងៗទៀតបើសិនជាមាន។ ក៏ប៉ុន្តែ បន្ទាប់ពីនោះមក សិស្សនោះត្រូវយកទៅប្រើឬអត់គឺគ្មានបញ្ហាអ្វីទាំងអស់ គឺវាប្រៀបបានទៅនឹងវត្ថុមួយដែលត្រូវបានទទួលយកហើយរួចទុកចោល វានឹងមិនបង្កឲ្យមានបញ្ហាអ្វីឡើយ។ បញ្ហាគឺនៅត្រង់ថា បើនៅក្នុងវិធីគ្មានដំណាងដែលជាពាក្យថា សិស្ស សម្រាប់ទទួលយកវត្ថុដែលជាសិស្សនោះទេ កំហុសស្តីពីការមានដំណាងមិនគ្រប់គ្រាន់នឹងកើតមានឡើង។

ម៉្យាងទៀត យើងត្រូវធ្វើការកត់សំគាល់ថា ការផ្តល់ដំណឹងជាសិស្សជាស្វ័យប្រវត្តិសម្រាប់ដំណាងទីមួយនៅក្នុងវិធី អាចប្រព្រឹត្តទៅបានតែនៅពេលណាដែលវិធីត្រូវយកទៅប្រើតាមរយសិស្សតែប៉ុណ្ណោះ។ ក្នុងករណីវិធីត្រូវយកទៅប្រើតាមរយៈតាមរយៈថ្នាក់របស់វា ដំណឹងសម្រាប់ដំណាងទីមួយនឹងមិនត្រូវបានផ្តល់ឲ្យទៅវិធីនោះដោយស្វ័យប្រត្តិឡើយ។ ដូចនេះនៅពេលដែលយើងយកវិធីផ្សេងៗមកប្រើតាមរយៈថ្នាក់របស់វា យើងចាំបាច់ត្រូវតែផ្តល់ដំណឹង

ណាមួយសម្រាប់ដំណាងទីមួយនៅក្នុងវិធីនោះ បើពុំនោះសោតទេ កំហុសនឹងកើតមានឡើង ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
class ក្រឡាផ្ទៃ() :
  ពី = 3.14
  def __init__(សិស្ស) :
    print("ស្ថាបនិកត្រូវបានយកទៅប្រើ។")

  def ក្រឡាផ្ទៃចតុកោណកែង(សិស្ស, បណ្តោយ, ទទឹង) :
    ផ្ទៃក្រឡា = បណ្តោយ * ទទឹង
    print("ក្រឡាផ្ទៃចតុកោណកែងគឺ៖", ផ្ទៃក្រឡា)

សិស្សក = ក្រឡាផ្ទៃ()
ក្រឡាផ្ទៃ.ក្រឡាផ្ទៃចតុកោណកែង(សិស្សក, 25, 5)
ក្រឡាផ្ទៃ.ក្រឡាផ្ទៃចតុកោណកែង(True, 25, 5)
```

ក្រឡាផ្ទៃ.ក្រឡាផ្ទៃចតុកោណកែង(សិស្សក, 25, 5) គឺជាការវិធីឈ្មោះ ក្រឡាផ្ទៃចតុកោណកែង មកប្រើតាមរយៈថ្នាក់របស់វា។ ដំណឹងដែលជាសិស្សឈ្មោះ សិស្សក ដំណឹងដែលជាលេខ 25 និងដំណឹងដែលជាលេខ 5 ត្រូវផ្តល់ឲ្យទៅវិធីនោះរៀងគ្នា។

ក្រឡាផ្ទៃ.ក្រឡាផ្ទៃចតុកោណកែង(True, 25, 5) គឺជាការយកវិធីឈ្មោះ ក្រឡាផ្ទៃចតុកោណកែង មកប្រើតាមរយៈថ្នាក់របស់វា។ ដំណឹងដែលជាតក្កវត្ត True ដំណឹងដែលជាលេខ 25 និងដំណឹងដែលជាលេខ 5 ត្រូវបានផ្តល់ឲ្យទៅដំណាងផ្សេងៗនៅក្នុងវិធីនោះរៀងគ្នា។

ក្រៅពីការយកទិន្នន័យគម្រូនិងឬវិធីមួយចំនួនទៅប្រើនៅខាងក្រៅថ្នាក់ យើងក៏អាចយកវត្ថុទាំងនោះមកប្រើនៅក្នុងវិធីផ្សេងៗទៀតនៅក្នុងថ្នាក់ជាមួយគ្នាបានដែរ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
class ក្រឡាផ្ទៃ() :
    ពី = 3.14
    def __init__(សិស្ស) :
        print("ស្ថាបនិកត្រូវបានយកទៅប្រើ។")

    def ក្រឡាផ្ទៃចតុកោណកែង(សិស្ស, បណ្តោយ, ទទឹង) :
        ផ្ទៃក្រឡា = បណ្តោយ * ទទឹង
        return ផ្ទៃក្រឡា

    def បង្ហាញលទ្ធផល(សិស្ស, បណ្តោយ, ទទឹង) :
        print("ពី គឺជាលេខ៖", ក្រឡាផ្ទៃ.ពី)
        print("ក្រឡាផ្ទៃចតុកោណកែងគឺ៖", សិស្ស.ក្រឡាផ្ទៃចតុកោណកែង(បណ្តោយ, ទទឹង))

សិស្សក = ក្រឡាផ្ទៃ()
សិស្សក.បង្ហាញលទ្ធផល(25, 5)
```

ក្រឡាផ្ទៃ.ពី គឺជាការយកទិន្នន័យគម្រូឈ្មោះ ពី មកប្រើនៅក្នុងវិធីឈ្មោះ បង្ហាញលទ្ធផល ដែលជាវិធីស្ថិតនៅក្នុងថ្នាក់ជាមួយនឹងទិន្នន័យគម្រូនោះ។ ដូចនេះការយកទិន្នន័យគម្រូមកប្រើអាចត្រូវធ្វើឡើងតាមរយៈថ្នាក់របស់វា។

សិស្ស.ក្រឡាផ្ទៃចតុកោណកែង(បណ្តោយ, ទទឹង) គឺជាការយកវិធីឈ្មោះ ក្រឡាផ្ទៃចតុកោណកែង មកប្រើនៅក្នុងវិធីឈ្មោះ បង្ហាញលទ្ធផល ដែលជាវិធីនៅក្នុងថ្នាក់

ជាមួយគ្នា។ ដូចនេះការយកវិធីមួយទៅប្រើនៅក្នុងវិធីមួយទៀតអាចត្រូវធ្វើឡើយតាមសិស្ស នៃថ្នាក់របស់វិធីទាំងនោះ។

ចំពោះស្ថាបនិកវិញ វាក៏ជាវិធីមួយដូចជាវិធីដទៃទៀតដែរ ដូចនេះស្ថាបនិកអាចទទួលយក ដំណឹងផ្សេងៗក្រៅពីសិស្សបាន។ ដោយស្ថាបនិកត្រូវយកទៅប្រើនៅពេលដែលថ្នាក់ត្រូវយក ទៅប្រើ ដូចនេះការផ្តល់ដំណឹងឲ្យទៅស្ថាបនិកគឺត្រូវធ្វើឡើងនៅពេលដែលថ្នាក់ត្រូវយកទៅ ប្រើ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
class ក្រឡាផ្ទៃ() :
    ពី = 3.14
    def __init__(សិស្ស, បណ្តោយ, ទទឹង) :
        សិស្ស.បង្ហាញលទ្ធផល(បណ្តោយ, ទទឹង)

    def ក្រឡាផ្ទៃចតុកោណកែង(សិស្ស, បណ្តោយ, ទទឹង) :
        ផ្ទៃក្រឡា = បណ្តោយ * ទទឹង
        return ផ្ទៃក្រឡា

    def បង្ហាញលទ្ធផល(សិស្ស, បណ្តោយ, ទទឹង) :
        print("ពី គឺជាលេខ៖", ក្រឡាផ្ទៃ.ពី)
        print("ក្រឡាផ្ទៃចតុកោណកែងគឺ៖", សិស្ស.ក្រឡាផ្ទៃចតុកោណកែង(បណ្តោយ, ទទឹង))

សិស្សក = ក្រឡាផ្ទៃ(25, 5)
```

def __init__(សិស្ស, បណ្តោយ, ទទឹង) : គឺជាបញ្ជាតម្រូវឲ្យបង្កើតស្ថាបនិកមួយដែលមាន ដំណាង បណ្តោយ និង ទទឹង សម្រាប់ទទួលយកដំណឹងដែលមិនមែនជាសិស្ស។

សិស្សក = ក្រឡាផ្ទៃ(25, 5) គឺជាការយកថ្នាក់ឈ្មោះ ក្រឡាផ្ទៃ មកប្រើដោយផ្តល់វត្ថុចំនួនពីរជាដំណឹងសម្រាប់ដំណាង បណ្តោយ និង ទទឹង នៅក្នុងស្ថាបនិកនៃថ្នាក់នោះ។

ទិន្នន័យគម្រូ

នៅក្នុងថ្នាក់ក៏ដូចជានៅខាងក្រៅថ្នាក់ដែរ ការយកទិន្នន័យគម្រូទៅប្រើគឺត្រូវធ្វើឡើងតាមរយៈថ្នាក់ឬតាមរយៈសិស្សនៃថ្នាក់របស់ទិន្នន័យគម្រូនោះ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
class ក្រឡាផ្ទៃ() :
  ពី = 3.14
  def __init__(សិស្ស, បណ្តោយ, ទទឹង) :
    សិស្ស.បង្ហាញលទ្ធផល(បណ្តោយ, ទទឹង)

  def ក្រឡាផ្ទៃចតុកោណកែង(សិស្ស, បណ្តោយ, ទទឹង) :
    ផ្ទៃក្រឡា = បណ្តោយ * ទទឹង
    return ផ្ទៃក្រឡា

  def បង្ហាញលទ្ធផល(សិស្ស, បណ្តោយ, ទទឹង) :
    print("ពី គឺជាលេខ៖", សិស្ស.ពី)
    print("ក្រឡាផ្ទៃចតុកោណកែងគឺ៖", សិស្ស.ក្រឡាផ្ទៃចតុកោណកែង(បណ្តោយ, ទទឹង))

សិស្សក = ក្រឡាផ្ទៃ(25, 5)
print(ក្រឡាផ្ទៃ.ពី)
```

print("*ពី គឺជាលេខ៖*", *សិស្ស.ពី*) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកទិន្នន័យគម្រូឈ្មោះ ពី មកប្រើ។ ការយកទិន្នន័យគម្រូនេះមកប្រើត្រូវធ្វើឡើងតាមរយៈសិស្សនៃថ្នាក់របស់ទិន្នន័យនោះ។

print(*ក្រឡាផ្ទៃ.ពី*) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកទិន្នន័យគម្រូឈ្មោះ ពី មកប្រើ។ ការយកទិន្នន័យនេះមកប្រើគឺត្រូវធ្វើឡើងតាមរយៈថ្នាក់របស់ទិន្នន័យនោះ។

ក្រៅពីការយកទិន្នន័យគម្រូនៃថ្នាក់ណាមួយមកប្រើ យើងក៏អាចយកទិន្នន័យនោះមកដូរថ្មីបានដែរ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
class ក្រឡាផ្ទៃ() :
  ពី = 3.14
  def __init__(សិស្ស, បណ្តោយ, ទទឹង) :
    សិស្ស.បង្ហាញលទ្ធផល(បណ្តោយ, ទទឹង)

  def ក្រឡាផ្ទៃចតុកោណកែង(សិស្ស, បណ្តោយ, ទទឹង) :
    ផ្ទៃក្រឡា = បណ្តោយ * ទទឹង
    return ផ្ទៃក្រឡា

  def បង្ហាញលទ្ធផល(សិស្ស, បណ្តោយ, ទទឹង) :
    print("ពី គឺជាលេខ៖", ក្រឡាផ្ទៃ.ពី)
    print("ក្រឡាផ្ទៃចតុកោណកែងគឺ៖", សិស្ស.ក្រឡាផ្ទៃចតុកោណកែង(បណ្តោយ, ទទឹង))

ក្រឡាផ្ទៃ.ពី = 3.1415
print("ទិន្នន័យគម្រូឈ្មោះ ពី ថ្មីគឺ៖", ក្រឡាផ្ទៃ.ពី)
```

ក្រឡាផ្ទៃ.ពី = 3.1415 គឺជាបញ្ជាតម្រូវឲ្យយកទិន្នន័យគម្រូឈ្មោះ ពី មកដូរថ្មីតាមរយៈថ្នាក់របស់វា។ ទិន្នន័យគម្រូឈ្មោះ ពី នេះត្រូវបានដូរពីរពីលេខ 3.14 ឲ្យទៅជាលេខ 3.1415 វិញ។

យើងត្រូវធ្វើការកត់សំគាល់ថា ការយកទិន្នន័យគម្រូមកដោះដូរថ្មីគឺត្រូវតែធ្វើឡើងតាមរយៈថ្នាក់របស់វា។ គឺយើងមិនអាចយកទិន្នន័យគម្រូនោះមកធ្វើការដោះដូរតាមរយៈសិស្សនៃថ្នាក់របស់វាបានឡើយ។ យើងនឹងដឹងថាមកពីហេតុអ្វីនៅពេលបន្តិចទៀតនេះ។

សិស្ស

យើងបានដឹងរួចមកហើយថាសិស្សគឺជាវត្ថុដែលជាលទ្ធផលបានមកពីការយកថ្នាក់មកប្រើ។ ដូចនេះសិស្សមានលក្ខណៈដូចជាវត្ថុដទៃទៀតដែរ បានន័យថានៅកន្លែងណាដែលវត្ថុផ្សេងៗអាចត្រូវយកទៅប្រើបាន សិស្សក៏អាចត្រូវយកទៅប្រើបានដូចគ្នា។ មួយវិញទៀត សិស្សគឺជាវត្ថុទទេស្អាតមួយក្រោយពីត្រូវបានបង្កើតឡើង តែវាមានទំនាក់ទំនងយ៉ាងជិទ្ធស្និទ្ធជាមួយនឹងថ្នាក់របស់វា ហើយគឺដោយសារសិស្សនេះហើយដែលយើងអាចវិធីផ្សេងៗនៅក្នុងថ្នាក់ទៅប្រើនៅក្នុងដែនកំណត់ដទៃទៀតបាន។

យើងគួរតែរំលឹកឡើងវិញថា នៅពេលដែលយើងយកវិធីណាមួយមកប្រើតាមរយៈសិស្សណាម្នាក់ សិស្សនោះនឹងត្រូវផ្តល់ជាដំណឹងឲ្យទៅដំណាងទីមួយនៅក្នុងវិធីនោះជាស្វ័យប្រវត្តិ។ យន្តការនេះគឺជាច្បាប់នៅក្នុងភាសា Python ។

ប្រការដែលសំខាន់មួយទៀតគឺថា សិស្សគឺជាវត្ថុដែលជាកន្លែងទទេស្អាតមួយនៅក្នុងសតិរបស់កំព្យូទ័រ ដូចនេះយើងអាចបង្កើតវត្ថុផ្សេងៗទៀតទុកនៅទីនោះដើម្បីកុំឲ្យច្រឡំទៅនឹងវត្ថុដទៃទៀតដែលមានឈ្មោះដូចគ្នា។ ដើម្បីបង្កើតវត្ថុផ្សេងៗទុកនៅក្នុងសិស្ស យើងត្រូវសរសេរកម្មវិធីដូចខាងក្រោមនេះ៖

```
class ក្រឡាផ្ទៃ() :
    ពី = 3.14
    def __init__(សិស្ស) :
        print("ស្ថាបនិកបានត្រូវយកទៅប្រើ។")

    def ក្រឡាផ្ទៃចតុកោណកែង(សិស្ស, បណ្តោយ, ទទឹង) :
        ផ្ទៃក្រឡា = បណ្តោយ * ទទឹង
        return ផ្ទៃក្រឡា

សិស្សក = ក្រឡាផ្ទៃ()
សិស្សក.បណ្តោយ = 25
សិស្សក.ទទឹង = 5
print(សិស្សក.បណ្តោយ)
print(សិស្សក.ទទឹង)
```

សិស្សក.បណ្តោយ = 25 គឺជាបញ្ជាតម្រូវឲ្យបង្កើតចំនួនគត់លេខ 25 មួយមានឈ្មោះថា បណ្តោយ ទុកនៅក្នុងសិស្សឈ្មោះ សិស្សក ។

សិស្សក.ទទឹង = 5 គឺជាបញ្ជាតម្រូវឲ្យបង្កើតចំនួនគត់លេខ 5 មួយទៀតមានឈ្មោះថា ទទឹង ទុកនៅក្នុងសិស្សឈ្មោះ សិស្សក ដដែល។

print(សិស្សក.បណ្តោយ) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកវត្ថុឈ្មោះ បណ្តោយ ដែលស្ថិតនៅក្នុងសិស្សឈ្មោះ សិស្សក មកប្រើការ។

print(សិស្សក.ទទឹង) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកវត្ថុឈ្មោះ ទទឹង ដែលស្ថិតនៅក្នុងសិស្សឈ្មោះ សិស្សក មកប្រើការ។

សរុបមកយើងឃើញថា នៅក្នុងសិស្សឈ្មោះ សិស្សក មានវត្ថុឈ្មោះ បណ្ដោយ និងវត្ថុឈ្មោះ ទទឹង ត្រូវបានបង្កើតឡើងនិងទុកនៅទីនោះ។ យើងហៅវត្ថុដែលមាននៅក្នុងសិស្សថាជា *សម្បត្តិសិស្ស* (instance attribute) ដែលអាចជាវត្ថុប្រភេទណាក៏បានដែរ។ ក៏ប៉ុន្តែភាគច្រើន គេនិយមបង្កើតវត្ថុដែលជាទិន្នន័យផ្សេងៗទុកនៅក្នុងសិស្សដើម្បីកុំឲ្យច្រឡំទៅនឹងវត្ថុផ្សេងៗ ទៀតដែលមានឈ្មោះដូចគ្នា។

ការបង្កើតសម្បត្តិសិស្សអាចត្រូវធ្វើតាមរបៀបម៉្យាងទៀតដូចខាងក្រោមនេះ៖

```
class ក្រឡាផ្ទៃ() :
    ពី = 3.14
    def __init__(សិស្ស, បណ្ដោយ, ទទឹង) :
        សិស្ស.បណ្ដោយ = បណ្ដោយ
        សិស្ស.ទទឹង = ទទឹង

    def ក្រឡាផ្ទៃចតុកោណកែង(សិស្ស, បណ្ដោយ, ទទឹង) :
        ផ្ទៃក្រឡា = បណ្ដោយ * ទទឹង
        return ផ្ទៃក្រឡា

សិស្សក = ក្រឡាផ្ទៃ(25, 5)
print(សិស្សក.បណ្ដោយ)
print(សិស្សក.ទទឹង)
```

សិស្ស.បណ្ដោយ = បណ្ដោយ គឺជាបញ្ជាតម្រូវឲ្យបង្កើតសម្បត្តិឈ្មោះ បណ្ដោយ មួយទុកនៅក្នុងសិស្សដែលមានឈ្មោះថា សិស្ស នៅក្នុងស្ថាបនិក។

សិស្ស.ទទឹង = ទទឹង គឺជាបញ្ជាតម្រូវឲ្យបង្កើតសម្បត្តិឈ្មោះ ទទឹង មួយទៀតទុកនៅក្នុងសិស្សដែលមានឈ្មោះជា សិស្ស នៅក្នុងស្ថាបនិក។

សិស្សក = ក្រឡាផ្ទៃ(25, 5) គឺជាការយកថ្នាក់ឈ្មោះ ក្រឡាផ្ទៃ មកប្រើដើម្បីបង្កើតសិស្សឈ្មោះ សិស្សក ម្នាក់។ កត្តានេះធ្វើឲ្យស្ថាបនិកនៅក្នុងថ្នាក់នោះត្រូវបានយកមកប្រើជាស្វ័យប្រវត្តិ ដែលជាប្រការបណ្តាលឲ្យសិស្សឈ្មោះ សិស្សក ដែលទើបនឹងត្រូវបានបង្កើតនោះត្រូវបានផ្តល់ជាដំណឹងឲ្យទៅស្ថាបនិកសម្រាប់ដំណាង សិស្ស និងដំណឹងជាលេខ 25 និងលេខ 5 ត្រូវបានផ្តល់ជាដំណឹងសម្រាប់ដំណាង បណ្តោយ និង ទទឹង រៀងគ្នា។ ហើយបន្ទាប់មកទៀត ក្រុមបញ្ជានៅក្នុងស្ថាបនិកនោះក៍ត្រូវបានយកទៅអនុវត្តដែរ ដែលជាប្រការធ្វើឲ្យសម្បត្តិឈ្មោះ បណ្តោយ និង ទទឹង ត្រូវបានបង្កើតឡើងនិងទុកនៅក្នុងសិស្សឈ្មោះ សិស្សក នោះ។ សម្បត្តិសិស្សទាំងនោះគឺជាដំណឹងដែលត្រូវបានផ្តល់ឲ្យទៅស្ថាបនិកនៅពេលដែលថ្នាក់ឈ្មោះ ក្រឡាផ្ទៃ ត្រូវបានយកមកប្រើ។

យើងឃើញថា ការបង្កើតសម្បត្តិសិស្សអាចត្រូវធ្វើទៅបាននៅខាងក្រៅថ្នាក់ក៏បាននិងឬនៅខាងក្នុងថ្នាក់ក៏បាន។ ក៏ប៉ុន្តែការបង្កើតសម្បត្តិសិស្សនៅខាងក្នុងថ្នាក់គឺត្រូវធ្វើឡើងនៅក្នុងវិធីណាមួយ ហើយជាទូទៅគេច្រើនប្រើស្ថាបនិកជាកន្លែងសម្រាប់បង្កើតសម្បត្តិសិស្សផ្សេងៗ។ គឺហេតុដូចនេះហើយបានជាវិធីឈ្មោះ __init__ ត្រូវហៅថាស្ថាបនិកព្រោះវាមាននាទីជាអ្នកបង្កើតសម្បត្តិសិស្សទាំងពួង។

ក្រោយពីសម្បត្តិសិស្សត្រូវបានបង្កើតរួចហើយ យើងអាចយកវត្ថុទាំងនោះទៅធ្វើការដោះដូរថ្មីបានគ្រប់ពេលវេលា។ ហើយការដោះដូរនេះអាចត្រូវធ្វើឡើងនៅខាងក្នុងថ្នាក់និងឬនៅខាងក្រៅថ្នាក់បានដូចគ្នា។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
class ក្រឡាផ្ទៃ() :
```

```
    ពី = 3.14
    def __init__(សិស្ស, បណ្តោយ, ទទឹង) :
        សិស្ស.បណ្តោយ = បណ្តោយ
        សិស្ស.ទទឹង = ទទឹង

    def ដូរសម្បត្តិសិស្ស(សិស្ស, បណ្តោយថ្មី) :
        សិស្ស.បណ្តោយ = បណ្តោយថ្មី

    def ក្រឡាផ្ទៃចតុកោណកែង(សិស្ស, បណ្តោយ, ទទឹង) :
        ផ្ទៃក្រឡា = បណ្តោយ * ទទឹង
        return ផ្ទៃក្រឡា

សិស្សក = ក្រឡាផ្ទៃ(25, 5)
សិស្សក.ដូរសម្បត្តិសិស្ស(50)
សិស្សក.ទទឹង = 10
print(សិស្សក.បណ្តោយ)
print(សិស្សក.ទទឹង)
```

សិស្ស.បណ្តោយ = បណ្តោយថ្មី គឺជាបញ្ជាតម្រូវឲ្យធ្វើការដោះដូរសម្បត្តិឈ្មោះ បណ្តោយ របស់សិស្សមានឈ្មោះថា សិស្ស នៅក្នុងវិធីឈ្មោះ ដូរសម្បត្តិសិស្ស ។

សិស្សក.ដូរសម្បត្តិសិស្ស(50) គឺជាការយកវិធីឈ្មោះ ដូរសម្បត្តិសិស្ស មកប្រើតាមរយៈសិស្សឈ្មោះ សិស្សក ដោយផ្តល់ដំណឹងជាលេខ 50 ឲ្យទៅវិធីនោះ។ ប្រការនេះបានធ្វើឲ្យសម្បត្តិឈ្មោះ បណ្តោយ របស់សិស្សឈ្មោះ សិស្សក ត្រូវបានប្តូរទៅជាលេខ 50 វិញ។

សិស្សក.ទទឹង = 10 គឺជាបញ្ជាតម្រូវឲ្យធ្វើការដោះដូរសម្បត្តិឈ្មោះ ទទឹង របស់សិស្សឈ្មោះ សិស្សក ឲ្យទៅជាលេខ 10 វិញ។

ការបង្កើតសិស្សក៏ជាការបង្កើតដែនកំណត់មួយដែរ ដូចនេះសម្បត្តិសិស្សមិនអាចត្រូវច្រឡំជាមួយនឹងសម្បត្តិថ្នាក់និងឬសម្បត្តិសិស្សផ្សេងៗទៀតបានឡើយ ទោះបីជាវត្ថុទាំងនោះមានឈ្មោះដូចគ្នាក៏ដោយ។ ហើយនៅពេលដែលយើងយកសម្បត្តិទាំងនោះមកប្រើតាមរយៈសិស្សណាមួយ ការស្វែងរកសម្បត្តិទាំងនោះត្រូវធ្វើឡើងនៅក្នុងសិស្សនោះមុន រួចបានឡើងទៅថ្នាក់របស់សិស្សនោះជាបន្តបន្ទាប់។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
class ក្រឡាផ្ទៃ() :
    ពី = 3.14
    def __init__(សិស្ស, ពី) :
        សិស្ស.ពី = ពី

    def ក្រឡាផ្ទៃចតុកោណកែង(សិស្ស, បណ្តោយ, ទទឹង) :
        ផ្ទៃក្រឡា = បណ្តោយ * ទទឹង
        return ផ្ទៃក្រឡា

សិស្សក = ក្រឡាផ្ទៃ(3.1415)
សិស្សខ = ក្រឡាផ្ទៃ(3.141592)
print(សិស្សក.ពី)
print(សិស្សខ.ពី)
```

ពី = 3.14 គឺជាបញ្ជាតម្រូវឲ្យបង្កើតទិន្នន័យគម្រូឈ្មោះ ពី មួយដែលជាសម្បត្តិនៅក្នុងថ្នាក់ឈ្មោះ ក្រឡាផ្ទៃ ។

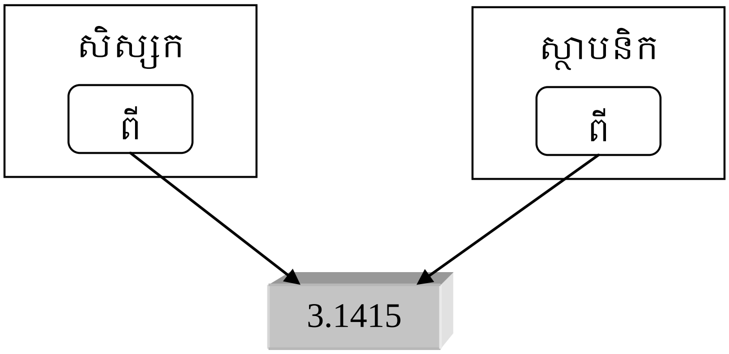
def __init__(*សិស្ស, ពី*) : គឺជាបញ្ជាតម្រូវឲ្យបង្កើតស្ថាបនិកមួយដែលនៅក្នុងនោះមានដំណាងមួយជាឈ្មោះ ពី ។

សិស្ស.ពី = *ពី* គឺជាបញ្ជាតម្រូវឲ្យបង្កើតសម្បត្តិឈ្មោះ ពី មួយទុកនៅក្នុងសិស្សដែលមានឈ្មោះថា សិស្ស នៅក្នុងស្ថាបនិក។

សិស្សក = *ក្រឡាផ្ទៃ*(3.1415) គឺជាការយកថ្នាក់ឈ្មោះ ក្រឡាផ្ទៃ មកប្រើដើម្បីបង្កើតសិស្សឈ្មោះ សិស្សក ម្នាក់។ នៅពេលដែលថ្នាក់ឈ្មោះ ក្រឡាផ្ទៃ ត្រូវយកមកប្រើ ស្ថាបនិកនៅក្នុងថ្នាក់នោះក៏ត្រូវយកមកប្រើជាស្វ័យប្រវត្តិដែរ។ ហើយសិស្សឈ្មោះ សិស្សក ដែលទើបនឹងត្រូវបង្កើតនោះ ក៏ត្រូវបានផ្តល់ជាដំណឹងសម្រាប់ដំណាងទីមួយនៅក្នុងស្ថាបនិកនោះជាស្វ័យប្រវត្តិ។ ចំណែកឯដំណឹងដែលជាលេខ 3.1415 វិញ ត្រូវបានផ្តល់ឲ្យទៅស្ថាបនិកសម្រាប់ដំណាង ពី ។ ដូចនេះវត្ថុដែលជាលេខ 3.1415 និងមានឈ្មោះជា ពី មួយត្រូវបានបង្កើតឡើងនៅក្នុងស្ថាបនិក។ ហើយនៅពេលដែល បញ្ជា សិស្ស.ពី = ពី ត្រូវយកទៅអនុវត្តឈ្មោះ ពី មួយទៀតស្ថិតនៅក្នុងដែនកំណត់ដែលជាសិស្សឈ្មោះ សិស្សក ក៏ត្រូវបានភ្ជាប់ទៅនឹងលេខ 3.1415 នោះដែរ។

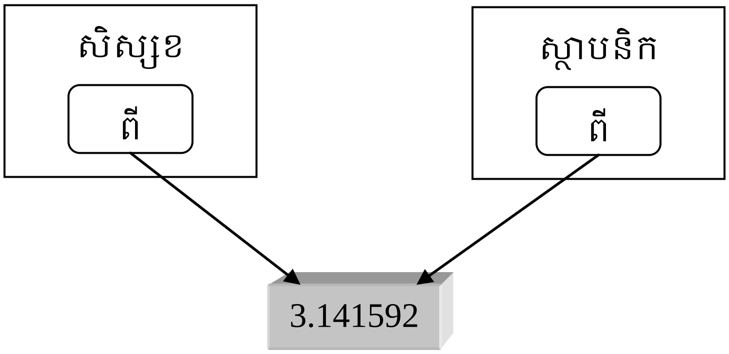
ឈ្មោះ ពី នៅក្នុងស្ថាបនិក និងឈ្មោះ ពី នៅក្នុងដែនកំណត់ដែលជាសិស្សឈ្មោះ សិស្សក គឺជាឈ្មោះពីរផ្សេងគ្នា ព្រោះវាស្ថិតនៅក្នុងដែនកំណត់ពីរផ្សេងគ្នា។ ម្យ៉ាងទៀតឈ្មោះ ពី ដែលជាដំណាងនៅក្នុងស្ថាបនិកត្រូវបានលុបចោលវិញនៅពេលដែលស្ថាបនិកត្រូវយកទៅប្រើរួចហើយ។ តែចំណែកឯឈ្មោះ ពី នៅក្នុងដែនកំណត់ដែលជាសិស្សឈ្មោះ សិស្សក វិញ ត្រូវស្ថិតនៅរហូតដល់សិស្សនោះត្រូវលុបចោល។

សិស្សក.ពី = ពី

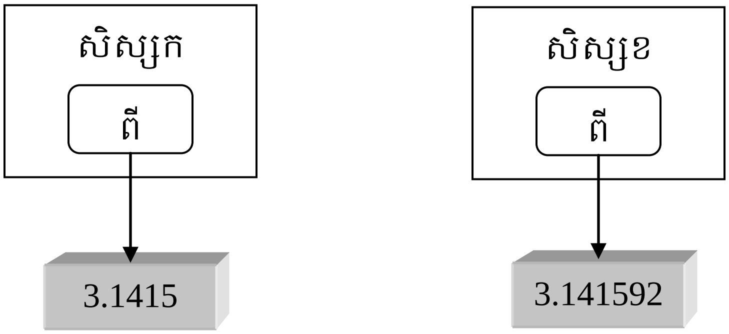


សិស្សខ = ក្រឡាផ្ទៃ(3.141592) គឺជាការយកថ្នាក់ឈ្មោះ ក្រឡាផ្ទៃ មកប្រើដើម្បីបង្កើតសិស្ស ឈ្មោះ សិស្សខ ម្នាក់ទៀត និងបង្កើតសម្បត្តិឈ្មោះ ពី មួយទៀតទុកនៅក្នុងសិស្សឈ្មោះ សិស្សខ នោះ។

សិស្សខ.ពី = ពី



ដូចនេះយើងឃើញថាសិស្សឈ្មោះ សិស្សក និងសិស្សឈ្មោះ សិស្សខ សុទ្ធតែមានសម្បត្តិ ឈ្មោះ ពី ដូចគ្នា តែវត្ថុទាំងពីរនោះជាវត្ថុខុសគ្នា ព្រោះវាស្ថិតនៅក្នុងដែនកំណត់ដែលជា សិស្សពីរផ្សេងគ្នា។



print(*សិស្សក.ពី*) គឺជាបញ្ហាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកវត្ថុឈ្មោះ ពី មកប្រើតាមរយៈសិស្សឈ្មោះ សិស្សក ។ យោងទៅតាមច្បាប់នៅក្នុងភាសា Python ការស្វែងរកវត្ថុឈ្មោះ ពី នោះត្រូវធ្វើឡើងនៅក្នុងសិស្សនោះមុនរួចបានឡើងទៅថ្នាក់របស់សិស្សនោះជាក្រោយ។ អាស្រ័យហេតុនេះ វត្ថុឈ្មោះ ពី ដែលជាសម្បត្តិរបស់សិស្សឈ្មោះ សិស្សក ត្រូវបានយកមកប្រើ ព្រោះវាត្រូវបានរកឃើញនៅក្នុងសិស្សនោះមុនគេ។

print(*សិស្សខ.ពី*) គឺជាបញ្ហាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកវត្ថុឈ្មោះ ពី មកប្រើតាមរយៈសិស្សឈ្មោះ សិស្សខ ។ យោងទៅតាមច្បាប់នៅក្នុងភាសា Python ការស្វែងរកវត្ថុឈ្មោះ ពី នោះត្រូវធ្វើឡើងនៅក្នុងសិស្សនោះមុនរួចបានឡើងទៅថ្នាក់របស់សិស្សនោះជាក្រោយ។ អាស្រ័យហេតុនេះ វត្ថុឈ្មោះ ពី ដែលជាសម្បត្តិរបស់សិស្សឈ្មោះ សិស្សខ ត្រូវបានយកមកប្រើ ព្រោះវាត្រូវបានរកឃើញនៅក្នុងសិស្សនោះមុនគេ។

សរុបមក នៅពេលដែលកម្មវិធីខាងលើនេះមានដំណើរការ ឈ្មោះ ពី ចំនួនបួនត្រូវបានបង្កើតឡើងនៅក្នុងដែនកំណត់បួនខុសៗគ្នា។ ឈ្មោះទាំងនោះមានដូចខាងក្រោមនេះ៖

- ឈ្មោះ ពី របស់ទិន្នន័យគម្រូ 3.14 ត្រូវបានបង្កើតឡើងនៅក្នុងថ្នាក់ឈ្មោះ ក្រឡាផ្ទៃ ។

- ឈ្មោះ ពី មួយទៀតដែលជាដំណាងនៅក្នុងស្ថាបនិកត្រូវបានបង្កើតឡើងដើម្បីភ្ជាប់ទៅនឹងដំណឹងដែលជាលេខត្រូវបានផ្តល់ឲ្យទៅស្ថាបនិក។ ឈ្មោះ ពី នេះត្រូវលុបចោលវិញនៅពេលដែលស្ថាបនិកត្រូវបានយកទៅប្រើរួចហើយ។
- ឈ្មោះ ពី មួយទៀតត្រូវបានបង្កើតឡើងនៅក្នុងសិស្សឈ្មោះ សិស្សក និងត្រូវបានភ្ជាប់ទៅនឹងលេខ 3.1415 ដែលមានឈ្មោះ ពី ដូចគ្នានៅស្ថាបនិក។
- ឈ្មោះ ពី មួយទៀតត្រូវបានបង្កើតឡើងនៅក្នុងសិស្សឈ្មោះ សិស្សខ និងត្រូវបានភ្ជាប់ទៅនឹងលេខ 3.141592 ដែលមានឈ្មោះ ពី ដូចគ្នានៅក្នុងស្ថាបនិក។

ដោយហេតុថាយើងអាចបង្កើតសម្បត្តិមានឈ្មោះដូចគ្នានៅក្នុងថ្នាក់និងនៅក្នុងសិស្ស ដូចនេះការប៉ុនប៉ងយកសម្បត្តិក្នុងថ្នាក់មកធ្វើការដោះដូរតាមរយៈសិស្ស គឺជាការបង្កើតសម្បត្តិសិស្សថ្មីមានឈ្មោះដូចទៅនឹងសម្បត្តិនៅក្នុងថ្នាក់នោះទៅវិញទេ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
class ក្រឡាផ្ទៃ() :
  ពី = 3.14
  def __init__(សិស្ស) :
    print("ស្ថាបនិកត្រូវបានយកទៅប្រើ។")

  def ក្រឡាផ្ទៃចតុកោណកែង(សិស្ស, បណ្តោយ, ទទឹង) :
    ផ្ទៃក្រឡា = បណ្តោយ * ទទឹង
    return ផ្ទៃក្រឡា

សិស្សក = ក្រឡាផ្ទៃ()
សិស្សក.ពី = 3.1415
print(ក្រឡាផ្ទៃ.ពី)
print(សិស្សក.ពី)
```

សិស្ស.ពី = 3.1415 គឺជាការប៉ុនប៉ងយកទិន្នន័យគម្រូនៅក្នុងថ្នាក់ឈ្មោះ ក្រឡាផ្ទៃ មកធ្វើការដោះដូរតាមរយៈសិស្សឈ្មោះ សិស្សក ។ ក៏ប៉ុន្តែទង្វើនេះគឺជាការបង្កើតសម្បត្តិឈ្មោះ ពី មួយទៀតទុកនៅក្នុងសិស្សឈ្មោះ សិស្សក នោះវិញទេ គឺមិនមែនជាការយកទិន្នន័យគម្រូឈ្មោះ ពី មកធ្វើការដោះដូរឡើយ។

ដូចនេះគ្រប់ការយកទិន្នន័យគម្រូមកធ្វើការដោះដូរ ត្រូវតែត្រូវធ្វើឡើងតាមរយៈថ្នាក់របស់វត្ថុនោះ។

មួយវិញទៀត ទិន្នន័យគម្រូនិងវិធីផ្សេងៗនៅក្នុងថ្នាក់ គឺហាក់ដូចជាសម្បត្តិសាធារណៈមួយចំនួនដែលអាចត្រូវយកទៅប្រើតាមរយៈសិស្សណាមួយក៏បានដែរ ឲ្យតែសិស្សទាំងនោះជាសិស្សនៃថ្នាក់ដែលមានសម្បត្តិទាំងនោះ។ ក៏ប៉ុន្តែចំពោះសម្បត្តិសិស្សវិញ វាគឺជាវត្ថុផ្ទាល់ខ្លួនរបស់សិស្សម្នាក់ៗ ទោះបីជាវត្ថុទាំងនោះមានឈ្មោះដូចគ្នាក៏ដោយ។ ហើយគ្រប់ការសម្បត្តិរបស់សិស្សណាមួយទៅប្រើ គឺត្រូវតែត្រូវធ្វើឡើងតាមរយៈសិស្សនោះ។

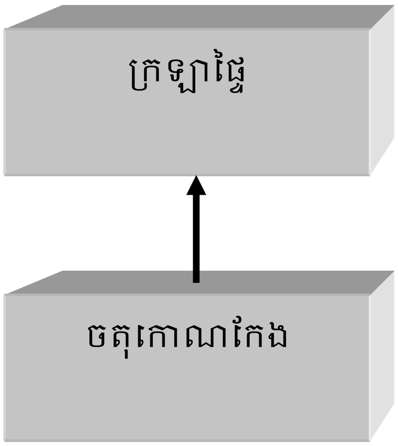
ការបន្តថ្នាក់

ការបន្តថ្នាក់ (inheritance) គឺជាការបង្កើតថ្នាក់មួយបន្តភ្ជាប់ទៅនឹងថ្នាក់ផ្សេងៗទៀត។ ដើម្បីបង្កើតថ្នាក់មួយបន្តភ្ជាប់ទៅនឹងថ្នាក់មួយទៀត យើងត្រូវធ្វើដូចខាងក្រោមនេះ៖

```
class ក្រឡាផ្ទៃ() :
  ពី = 3.14
  def __init__(សិស្ស) :
    print("ស្ថាបនិកនៃថ្នាក់ឈ្មោះ ក្រឡាផ្ទៃ ។")

  def ផ្ទៃក្រឡា(សិស្ស) :
```

```
        print("ផ្ទៃក្រឡា")

class ចតុកោណកែង(ក្រឡាផ្ទៃ) :
    def ផ្ទៃចតុកោណកែង(សិស្ស, បណ្តោយ=0, ទទឹង=0) :
        ផ្ទៃ = បណ្តោយ * ទទឹង
        print("ផ្ទៃក្រឡាចតុកោណកែងគឺ៖", ផ្ទៃ)
```

class *ចតុកោណកែង(ក្រឡាផ្ទៃ)* : គឺជាបង្កើតថ្នាក់ឈ្មោះ ចតុកោណកែង មួយបន្តភ្ជាប់ទៅនឹងថ្នាក់ឈ្មោះ ក្រឡាផ្ទៃ ។ ក្នុងករណីនេះ ថ្នាក់ឈ្មោះ ក្រឡាផ្ទៃ ត្រូវហៅថា *ថ្នាក់មេ* (superclass) និងថ្នាក់ឈ្មោះ ចតុកោណកែង ត្រូវហៅថា *ថ្នាក់រង* (subclass) ។



ផលប្រយោជន៍នៃការបន្តថ្នាក់គឺថា តាមរយៈសិស្សនៃថ្នាក់រងនិងឬថ្នាក់រង យើងអាចយកសម្បត្តិនៅក្នុងថ្នាក់មេនិងថ្នាក់រងទាំងអស់មកប្រើការ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
class ក្រឡាផ្ទៃ() :
    ពី = 3.14
    def __init__(សិស្ស) :
```

```
        print("ស្ថាបនិកនៃថ្នាក់ឈ្មោះ ក្រឡាផ្ទៃ ។")

    def ផ្ទៃក្រឡា(សិស្ស) :
        print("ផ្ទៃក្រឡា")

class ចតុកោណកែង(ក្រឡាផ្ទៃ) :
    def ផ្ទៃចតុកោណកែង(សិស្ស, បណ្តោយ=0, ទទឹង=0) :
        ផ្ទៃ = បណ្តោយ * ទទឹង
        print("ផ្ទៃក្រឡាចតុកោណកែងគឺៈ", ផ្ទៃ)

ចតុសិស្ស = ចតុកោណកែង()
print(ចតុកោណកែង.ពី)
ចតុសិស្ស.ផ្ទៃចតុកោណកែង()
ចតុសិស្ស.ផ្ទៃក្រឡា()
```

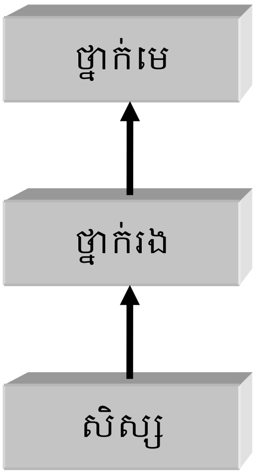
ចតុសិស្ស = ចតុកោណកែង() គឺជាបញ្ជាតម្រូវឲ្យបង្កើតសិស្សឈ្មោះ ចតុសិស្ស ម្នាក់ដែលជាសិស្សនៃថ្នាក់រងឈ្មោះ ចតុកោណកែង ។ ប្រការនេះបណ្តាលឲ្យស្ថាបនិកត្រូវយកមកប្រើជាស្វ័យប្រវត្តិ។ ការស្វែងរកវិធីដែលជាស្ថាបនិកត្រូវធ្វើឡើងនៅក្នុងសិស្សនោះមុន រួចបានឡើងទៅថ្នាក់របស់សិស្សនោះដែលជាថ្នាក់ឈ្មោះ ចតុកោណកែង រួចបានឡើងជាបន្តទៅទៀតទៅថ្នាក់មេនៃថ្នាក់របស់សិស្សនោះ។ វិធីឈ្មោះ __init__ ដែលជាស្ថាបនិកត្រូវបានរកឃើញនៅក្នុងថ្នាក់មេឈ្មោះ ក្រឡាផ្ទៃ ។

print(ចតុកោណកែង.ពី) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកទិន្នន័យគម្រូឈ្មោះ ពី មកប្រើតាមរយៈថ្នាក់ឈ្មោះ ចតុកោណកែង ។ ការស្វែងរកវត្ថុនោះត្រូវធ្វើឡើងនៅក្នុងថ្នាក់ឈ្មោះ ចតុកោណកែង នោះមុន រួចបានឡើងទៅថ្នាក់មេរបស់ថ្នាក់នោះ។ ទិន្នន័យគម្រូឈ្មោះ ពី ត្រូវបានរកឃើញនៅក្នុងថ្នាក់មេឈ្មោះ ក្រឡាផ្ទៃ ។

ចតុសិស្ស.ផ្ទៃចតុកោណកែង() គឺជាការយកវិធីឈ្មោះ ចតុកោណកែង មកប្រើតាមរយៈសិស្សឈ្មោះ ចតុសិស្ស ។ ការស្វែងរកវិធីនេះត្រូវធ្វើឡើងនៅក្នុងសិស្សនោះមុន រួចបានឡើងទៅថ្នាក់របស់សិស្សនោះ រួចបានឡើងទៅថ្នាក់មេនៃថ្នាក់របស់សិស្សនោះ។ វិធីឈ្មោះ ផ្ទៃចតុកោណកែង ត្រូវបានរកឃើញនៅក្នុងថ្នាក់របស់សិស្សនោះដែលជាថ្នាក់ឈ្មោះ ចតុកោណកែង ។

ចតុសិស្ស.ផ្ទៃក្រឡា() គឺជាការយកវិធីឈ្មោះ ក្រឡាផ្ទៃ មកប្រើតាមរយៈសិស្សឈ្មោះ ចតុសិស្ស ។ ការស្វែងរកវិធីនោះត្រូវធ្វើឡើងនៅក្នុងសិស្សនោះមុន រួចបានឡើងទៅថ្នាក់សិស្សនោះ រួចបានទៅថ្នាក់មេនៃថ្នាក់របស់សិស្សនោះ។ វិធីឈ្មោះ ផ្ទៃក្រឡា ត្រូវបានរកឃើញនៅក្នុងថ្នាក់ឈ្មោះ ក្រឡាផ្ទៃ ដែលជាថ្នាក់មេនៃថ្នាក់របស់សិស្សនោះ។

សរុបមក នៅពេលដែលសម្បត្តិណាមួយត្រូវបានយកមកប្រើតាមរយៈសិស្សណាម្នាក់ ការស្វែងរកវត្ថុនោះត្រូវធ្វើឡើងនៅក្នុងសិស្សនោះមុន រួចបានឡើងទៅថ្នាក់របស់សិស្សនោះ រួចបានឡើងទៅថ្នាក់មេនៃថ្នាក់របស់សិស្សនោះបើសិនជាមាន។ តែបើសម្បត្តិនោះត្រូវបានយកមកប្រើតាមរយៈថ្នាក់ណាមួយវិញ ការស្វែងរកវត្ថុនោះត្រូវធ្វើឡើងនៅក្នុងថ្នាក់នោះមុន រួចបានឡើងទៅថ្នាក់មេរបស់ថ្នាក់នោះបើសិនជាមាន។



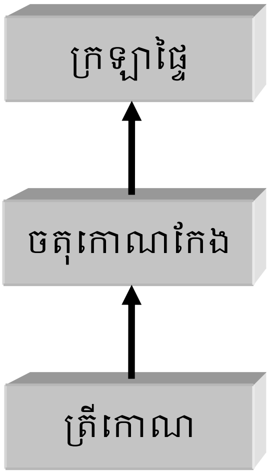
នៅក្នុងការបន្តថ្នាក់ យើងអាចយកថ្នាក់ចំនួនប៉ុន្មានក៏បានដែរមកបន្តគ្នា។ ពោលគឺយើងអាចបង្កើតថ្នាក់មួយបន្តភ្ជាប់ទៅនឹងថ្នាក់មួយទៀតដែលត្រូវបានបន្តភ្ជាប់ទៅនឹងថ្នាក់មួយទៀតដែលត្រូវបានបន្តភ្ជាប់ទៅនឹងថ្នាក់មួយទៀត...

ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
class ក្រឡាផ្ទៃ() :
  ពី = 3.14
  def __init__(សិស្ស) :
    print("ស្ថាបនិកនៃថ្នាក់ឈ្មោះ ក្រឡាផ្ទៃ ។")

  def ផ្ទៃក្រឡា(សិស្ស) :
    print("ផ្ទៃក្រឡា")

class ចតុកោណកែង(ក្រឡាផ្ទៃ) :
  def ផ្ទៃចតុកោណកែង(សិស្ស, បណ្តោយ=0, ទទឹង=0) :
    ផ្ទៃ = បណ្តោយ * ទទឹង
    print("ផ្ទៃក្រឡាចតុកោណកែងគឺ៖", ផ្ទៃ)

class ត្រីកោណ(ចតុកោណកែង) :
  def ផ្ទៃត្រីកោណ(សិស្ស, បាត, កំពស់) :
    ផ្ទៃ = បាត * កំពស់ / 2
    print("ក្រឡាផ្ទៃត្រីកោណគឺ៖", ផ្ទៃ)
```

class *ចតុកោណកែង(ក្រឡាផ្ទៃ)* : គឺជាការបង្កើតថ្នាក់ឈ្មោះ ចតុកោណកែង មួយបន្តភ្ជាប់ទៅនឹងថ្នាក់ឈ្មោះ ក្រឡាផ្ទៃ ។

class *ត្រីកោណ(ចតុកោណកែង)* : គឺជាការបង្កើតថ្នាក់ឈ្មោះ ត្រីកោណ មួយបន្តភ្ជាប់ទៅនឹង ថ្នាក់ឈ្មោះ ចតុកោណកែង ។



ក្នុងករណីមានថ្នាក់ជាច្រើនតភ្ជាប់គ្នាពីមួយទៅមួយ នៅពេលដែលសម្បត្តិណាមួយត្រូវបាន យកមកប្រើ ការស្វែងរកសម្បត្តិនោះត្រូវធ្វើឡើងជាដំបូងនៅក្នុងសិស្សឬថ្នាក់ដែលតាមរយៈវា សម្បត្តិនោះត្រូវយកមកប្រើ រួចបានឡើងទៅថ្នាក់លើជាបន្តបន្ទាប់។ ពិនិត្យកម្មវិធីខាងក្រោម នេះ៖

```
class ក្រឡាផ្ទៃ() :
    ពី = 3.14
    def __init__(សិស្ស) :
        print("ស្ថាបនិកនៃថ្នាក់ឈ្មោះ ក្រឡាផ្ទៃ ។")

    def ផ្ទៃក្រឡា(សិស្ស) :
        print("ផ្ទៃក្រឡា")

class ចតុកោណកែង(ក្រឡាផ្ទៃ) :
```

```
    def ផ្ទៃចតុកោណកែង(សិស្ស, បណ្ដោយ=0, ទទឹង=0) :
        ផ្ទៃ = បណ្ដោយ * ទទឹង
        print("ផ្ទៃក្រឡាចតុកោណកែងគឺ៖", ផ្ទៃ)

class ត្រីកោណ(ចតុកោណកែង) :
    def ផ្ទៃត្រីកោណ(សិស្ស, បាត, កំពស់) :
        ផ្ទៃ = បាត * កំពស់ / 2
        print("ក្រឡាផ្ទៃត្រីកោណគឺ៖", ផ្ទៃ)

សិស្សក = ត្រីកោណ()
សិស្សក.ផ្ទៃត្រីកោណ(25, 5)
សិស្សក.ផ្ទៃចតុកោណកែង(25, 5)
print(ត្រីកោណ.ពី)
```

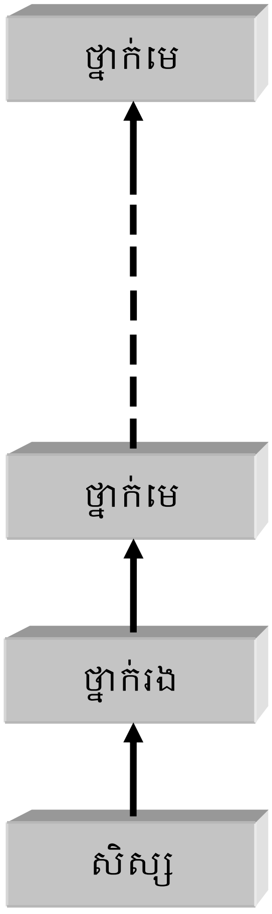
សិស្សក = ត្រីកោណ() គឺជាការយកថ្នាក់ឈ្មោះ ត្រីកោណ មកប្រើដើម្បីបង្កើតសិស្សឈ្មោះ សិស្សក ម្នាក់។ ប្រការនេះបណ្ដាលឲ្យស្ថាបនិកត្រូវយកមកប្រើជាស្វ័យប្រវត្តិ។ ការស្វែងរកស្ថាបនិកត្រូវធ្វើឡើងនៅក្នុងសិស្សឈ្មោះ សិស្សក នោះមុន រួចបានឡើងទៅថ្នាក់របស់សិស្សនោះ រួចបានឡើងទៅថ្នាក់មេនៃថ្នាក់របស់សិស្សនោះជាបន្តបន្ទាប់។ ស្ថាបនិកដែលជាវិធីឈ្មោះ __init__ ត្រូវបានរកឃើញនៅក្នុងថ្នាក់ឈ្មោះ ក្រឡាផ្ទៃ ដែលជាថ្នាក់មេនៅខាងលើគេបំផុត។

សិស្សក.ផ្ទៃត្រីកោណ(25, 5) គឺជាការយកវិធីឈ្មោះ ផ្ទៃត្រីកោណ មកប្រើតាមរយៈសិស្សឈ្មោះ សិស្សក ។ ប្រការនេះបណ្ដាលឲ្យការស្វែងរកវិធីនោះត្រូវធ្វើឡើងជាដំបូងនៅក្នុងសិស្សឈ្មោះ សិស្សក នោះមុន រួចបានឡើងទៅថ្នាក់របស់សិស្សនោះ រួចបានឡើងទៅថ្នាក់

មេនៃថ្នាក់របស់សិស្សនោះជាបន្តបន្ទាប់។ វិធីឈ្មោះ ផ្ទៃត្រីកោណ ត្រូវបានរកឃើញនៅក្នុងថ្នាក់ឈ្មោះ ត្រីកោណ ដែលជាថ្នាក់របស់សិស្សនោះ។

សិស្សក.ផ្ទៃចតុកោណកែង(25, 5) គឺជាការយកវិធីឈ្មោះ ផ្ទៃចតុកោណកែង មកប្រើតាមរយៈសិស្សឈ្មោះ សិស្សក ។ ប្រការនេះបណ្តាលឲ្យការស្វែងរកវិធីនោះត្រូវធ្វើឡើងជាដំបូងនៅក្នុងសិស្សនោះមុន រួចបានឡើងទៅថ្នាក់របស់សិស្សនោះ រួចបានឡើងទៅថ្នាក់មេនៃថ្នាក់របស់សិស្សនោះជាបន្តបន្ទាប់។ វិធីឈ្មោះ ផ្ទៃចតុកោណកែង ត្រូវបានរកឃើញនៅក្នុងថ្នាក់ឈ្មោះ ចតុកោណកែង ដែលជាថ្នាក់មេនៃថ្នាក់ឈ្មោះ ត្រីកោណ ។

ការស្វែងរកសម្បត្តិផ្សេងៗអាចត្រូវតាងដោយគំនូសបំព្រួញដូចខាងក្រោមនេះ៖



អាស្រ័យទៅតាមវិធាននៃការស្វែងរកសម្បត្តិដូចខាងលើ យើងមិនអាចយកសម្បត្តិនៅក្នុងថ្នាក់រងមកប្រើតាមរយៈថ្នាក់មេឬសិស្សនៃថ្នាក់មេបានឡើយ។ ការប៉ុនប៉ងធ្វើដូចនេះនឹងបង្កឲ្យមានកំហុសកើតមានឡើង។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
class ក្រឡាផ្ទៃ() :
    ពី = 3.14
    def __init__(សិស្ស) :
        print("ស្ថាបនិកនៃថ្នាក់ឈ្មោះ ក្រឡាផ្ទៃ ។")

    def ផ្ទៃក្រឡា(សិស្ស) :
        print("ផ្ទៃក្រឡា")

class ចតុកោណកែង(ក្រឡាផ្ទៃ) :
    def ផ្ទៃចតុកោណកែង(សិស្ស, បណ្តោយ=0, ទទឹង=0) :
        ផ្ទៃ = បណ្តោយ * ទទឹង
        print("ផ្ទៃក្រឡាចតុកោណកែងគឺ៖", ផ្ទៃ)

class ត្រីកោណ(ចតុកោណកែង) :
    def ផ្ទៃត្រីកោណ(សិស្ស, បាត, កំពស់) :
        ផ្ទៃ = បាត * កំពស់ / 2
        print("ក្រឡាផ្ទៃត្រីកោណគឺ៖", ផ្ទៃ)

សិស្សក = ចតុកោណកែង()
សិស្សក.ផ្ទៃត្រីកោណ(25, 5)
ចតុកោណកែង.ផ្ទៃត្រីកោណ(សិស្ស, 25, 5)
```

សិស្សក.ផ្ទៃត្រីកោណ(25, 5) គឺជាការសាកល្បងយកវិធីឈ្មោះ ផ្ទៃត្រីកោណ មកប្រើតាមរយៈសិស្សឈ្មោះ សិស្សក ដែលជាសិស្សនៃថ្នាក់ឈ្មោះ ចតុកោណកែង ។ យោងទៅតាមវិធាននៃការស្វែងរកសម្បត្តិ ការស្វែងរកវិធីឈ្មោះ ផ្ទៃត្រីកោណ ត្រូវធ្វើឡើងជាដំបូងនៅក្នុងសិស្សឈ្មោះ សិស្សក នោះមុន រួចបានឡើងទៅថ្នាក់របស់សិស្សនោះ រួចបានឡើងទៅថ្នាក់មេនៃថ្នាក់របស់សិស្សនោះជាបន្តបន្ទាប់។ ដោយវិធីឈ្មោះ ផ្ទៃត្រីកោណ ស្ថិតនៅក្នុងថ្នាក់ឈ្មោះ ត្រីកោណ ដែលជាថ្នាក់រងនៃថ្នាក់ឈ្មោះ ចតុកោណកែង ដូចនេះវិធីនោះមិនអាចត្រូវរកឃើញឡើយ។ ប្រការនេះបណ្តាលឲ្យកំហុសមួយបានកើតមានឡើង។

ចតុកោណកែង.ផ្ទៃត្រីកោណ(*សិស្ស*, 25, 5) គឺជាការសាកល្បងយកវិធីឈ្មោះ ផ្ទៃត្រីកោណ មកប្រើតាមរយៈថ្នាក់ឈ្មោះ ចតុកោណកែង ។ យោងទៅតាមវិធាននៃការស្វែងរកសម្បត្តិ ការស្វែងរកវិធីឈ្មោះ ផ្ទៃត្រីកោណ ត្រូវធ្វើឡើងជាដំបូងនៅក្នុងថ្នាក់នោះមុន រួចបានឡើងទៅថ្នាក់មេរបស់ថ្នាក់នោះជាបន្តបន្ទាប់។ ដោយវិធីឈ្មោះ ផ្ទៃត្រីកោណ ស្ថិតនៅក្នុងថ្នាក់ឈ្មោះ ត្រីកោណ ដែលជាថ្នាក់រងនៃថ្នាក់ឈ្មោះ ចតុកោណកែង ដូចនេះវិធីនោះមិនអាចត្រូវរកឃើញឡើយ។

សរុបមកយើងឃើញថា នៅពេលដែលសម្បត្តិណាមួយត្រូវយកមកប្រើតាមរយៈសិស្សឬថ្នាក់ណាមួយ ការស្វែងរកវត្ថុពីរប្រភេទត្រូវធ្វើឡើងក្នុងពេលជាមួយគ្នា។ ជាដំបូងគឺការស្វែងរកវត្ថុដែលជាសិស្សឬថ្នាក់ដែលតាមរយៈវាសម្បត្តិនោះត្រូវបានយកមកប្រើ។ ការស្វែងរកវត្ថុដែលជាសិស្សឬថ្នាក់នោះ ត្រូវធ្វើឡើងដោយអនុលោមទៅតាមវិធាននៃការស្វែងរកវត្ថុនៅក្នុងដែនកំណត់ផ្សេងៗ។ ជាបន្ទាប់មកទៀតគឺការស្វែងរកវត្ថុដែលជាសម្បត្តិត្រូវយកមកប្រើ។ ការស្វែងរកសម្បត្តិនោះត្រូវធ្វើឡើងដោយអនុលោមទៅតាមវិធាននៃការស្វែងរកសម្បត្តិនៅក្នុងសិស្សនិងថ្នាក់ផ្សេងៗ។

ជាក់ស្តែងការសរសេរ៖ *សិស្សក.ផ្ទៃត្រីកោណ*(25, 5) នៅក្នុងកម្មវិធីខាងលើ គឺជាការយកសម្បត្តិឈ្មោះ ផ្ទៃត្រីកោណ មកប្រើតាមរយៈសិស្សឈ្មោះ សិស្សក ។ ប្រការនេះធ្វើឲ្យការស្វែងរកវត្ថុពីរប្រភេទត្រូវធ្វើឡើងក្នុងពេលជាមួយគ្នា។ មុនដំបូងគឺការស្វែងរកវត្ថុដែលជាសិស្សឈ្មោះ សិស្សក ។ ការស្វែងរកសិស្សនោះត្រូវធ្វើឡើងយោងទៅតាមវិធាននៃការស្វែងរកវត្ថុនៅក្នុងដែនកំណត់ផ្សេងៗ។ បន្ទាប់មកទៀត គឺជាការស្វែងវត្ថុដែលជាសម្បត្តិឈ្មោះ ផ្ទៃត្រីកោណ ។ ការស្វែងរកសម្បត្តិនោះត្រូវធ្វើឡើងដោយអនុលោមទៅតាមវិធាននៃការស្វែងរកសម្បត្តិនៅក្នុងសិស្សនិងថ្នាក់ផ្សេងៗ។

សម្បត្តិឈ្មោះដូចគ្នា

ដោយថ្នាក់ក៏ជាដែនកំណត់មួយដែរ ដូចនេះសម្បត្តិនៅក្នុងថ្នាក់និមួយៗមិនអាចត្រូវច្រឡំទៅនឹងសម្បត្តិនៅក្នុងថ្នាក់ផ្សេងៗទៀតបានឡើយ ទោះបីជាវត្ថុទាំងនោះមានឈ្មោះដូចគ្នាក៏ដោយ។ ម៉្យាងទៀត អត្ថប្រយោជន៍ជាសារវ័ន្តនៃការបង្កើតថ្នាក់ គឺដើម្បីអាចបង្កើតវត្ថុខុសៗគ្នាដែលមានឈ្មោះដូចគ្នា ព្រោះវត្ថុមានឈ្មោះដូចគ្នាតែស្ថិតនៅក្នុងថ្នាក់ខុសគ្នា គឺជាវត្ថុខុសគ្នា។ បើយើងមិនបង្កើតថ្នាក់ដែលជាកន្លែងសម្រាប់បង្កើតវត្ថុផ្សេងៗទៀតនោះទេ បញ្ហាឈ្មោះជាន់គ្នានឹងកើតមានឡើង។

នៅក្នុងករណីមានការបន្តថ្នាក់ដែលមានសម្បត្តិមានឈ្មោះដូចគ្នា នៅពេលដែលសម្បត្តិទាំងនោះត្រូវយកមកប្រើ សម្បត្តិដែលត្រូវរកឃើញមុនគេនឹងត្រូវយកមកប្រើ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
class ក្រឡាផ្ទៃ() :
    ពី = 3.14
    def __init__(សិស្ស, *វិមាត្រ) :
```

```
        សិស្ស.វិមាត្រ = វិមាត្រ

    def ផ្ទៃក្រឡា(សិស្ស) :
        print("ផ្ទៃក្រឡា")

class ចតុកោណកែង(ក្រឡាផ្ទៃ) :
    def __init__(សិស្ស, បណ្ដោយ, ទទឹង) :
        សិស្ស.បណ្ដោយ = បណ្ដោយ
        សិស្ស.ទទឹង = ទទឹង

    def ផ្ទៃក្រឡា(សិស្ស) :
        ផ្ទៃ = សិស្ស.បណ្ដោយ * សិស្ស.ទទឹង
        print("ផ្ទៃក្រឡាចតុកោណកែងគឺ៖", ផ្ទៃ)

class ត្រីកោណ(ចតុកោណកែង) :
    def __init__(សិស្ស, បាត, កំពស់) :
        សិស្ស.បាត = បាត
        សិស្ស.កំពស់ = កំពស់

    def ផ្ទៃក្រឡា(សិស្ស) :
        ផ្ទៃ = សិស្ស.បាត * សិស្ស.កំពស់ / 2
        print("ក្រឡាផ្ទៃត្រីកោណគឺ៖", ផ្ទៃ)

សិស្សត្រីកោណ = ត្រីកោណ(25, 5)
សិស្សចតុកោណ = ចតុកោណកែង(25, 5)
សិស្សត្រីកោណ.ផ្ទៃក្រឡា()
សិស្សចតុកោណ.ផ្ទៃក្រឡា()
```

សិស្សត្រីកោណ = ត្រីកោណ(25, 5) គឺជាការបង្កើតសិស្សឈ្មោះ សិស្សត្រីកោណ ម្នាក់ដែលជាសិស្សនៃថ្នាក់ឈ្មោះ ត្រីកោណ ។ ប្រការនេះធ្វើឲ្យស្ថាបនិកត្រូវយកមកប្រើ ហើយសិស្សឈ្មោះ សិស្សត្រីកោណ លេខ 25 និងលេខ 5 ត្រូវផ្តល់ឲ្យទៅស្ថាបនិកនោះជាស្វ័យប្រវត្តិ។

ការស្វែងរកស្ថាបនិកដែលជាវិធីមានឈ្មោះថា __init__ ត្រូវធ្វើឡើងនៅក្នុងសិស្សឈ្មោះ សិស្សត្រីកោណ នោះមុន រួចបានឡើងទៅថ្នាក់របស់សិស្សនោះ រួចបានឡើងទៅថ្នាក់មេនៃថ្នាក់របស់សិស្សនោះជាបន្តបន្ទាប់។ ដោយហេតុថាស្ថាបនិកនៅក្នុងថ្នាក់រងឈ្មោះ ត្រីកោណ ត្រូវបានរកឃើញមុនគេ ដូចនេះគឺជាស្ថាបនិកនៅក្នុងថ្នាក់ឈ្មោះ ត្រីកោណ នោះហើយដែលត្រូវយកទៅប្រើជាស្វ័យប្រវត្តិ។

សិស្សចតុកោណ = ចតុកោណកែង(25, 5) គឺជាការបង្កើតសិស្សឈ្មោះ សិស្សចតុកោណ ម្នាក់ដែលជាសិស្សនៃថ្នាក់ឈ្មោះ ចតុកោណកែង ។ ប្រការនេះធ្វើឲ្យស្ថាបនិកត្រូវយកមកប្រើ ហើយសិស្សឈ្មោះ សិស្សចតុកោណ លេខ 25 និងលេខ 5 ត្រូវផ្តល់ឲ្យទៅស្ថាបនិកនោះជាស្វ័យប្រវត្តិ។

ការស្វែងរកស្ថាបនិកដែលជាវិធីមានឈ្មោះថា __init__ ត្រូវធ្វើឡើងនៅក្នុងសិស្សឈ្មោះ សិស្សចតុកោណ នោះមុន រួចបានឡើងទៅថ្នាក់របស់សិស្សនោះ រួចបានឡើងទៅថ្នាក់មេនៃថ្នាក់របស់សិស្សនោះជាបន្តបន្ទាប់។ ដោយហេតុថាស្ថាបនិកនៅក្នុងថ្នាក់ឈ្មោះ ចតុកោណកែង ត្រូវបានរកឃើញមុនគេ ដូចនេះគឺជាស្ថាបនិកនៅក្នុងថ្នាក់ ចតុកោណកែង នោះហើយដែលត្រូវយកទៅប្រើជាស្វ័យប្រវត្តិ។

សិស្សត្រីកោណ.ផ្ទៃក្រឡា() គឺជាការយកវិធីឈ្មោះ ផ្ទៃក្រឡា មកប្រើតាមរយៈសិស្សឈ្មោះ សិស្សត្រីកោណ ។ ការស្វែងរកវិធីនោះត្រូវធ្វើឡើងជាដំបូងនៅក្នុងសិស្សនោះមុន រួចបានឡើងទៅថ្នាក់របស់សិស្សនោះ រួចបានឡើងទៅថ្នាក់មេនៃថ្នាក់របស់សិស្សនោះជាបន្តបន្ទាប់។

ដោយហេតុថាវិធីឈ្មោះ ផ្ទៃក្រឡា នៅក្នុងថ្នាក់ឈ្មោះ ត្រីកោណ ត្រូវបានរកឃើញមុនគេ ដូចនេះគឺជាវិធីឈ្មោះ ផ្ទៃក្រឡា នៅក្នុងថ្នាក់នេះហើយដែលត្រូវយកទៅប្រើ។

សិស្សចតុកោណ.ផ្ទៃក្រឡា() គឺជាការយកវិធីឈ្មោះ ផ្ទៃក្រឡា មកប្រើតាមរយៈសិស្សឈ្មោះ សិស្សចតុកោណ ។ ការស្វែងរកវិធីនោះត្រូវធ្វើឡើងជាដំបូងនៅក្នុងសិស្សនោះមុន រួចបាន ឡើងទៅថ្នាក់របស់សិស្សនោះ រួចបានឡើងទៅថ្នាក់មេនៃថ្នាក់របស់សិស្សនោះជាបន្តបន្ទាប់។ ដោយហេតុថាវិធីឈ្មោះ ផ្ទៃក្រឡា នៅក្នុងថ្នាក់ឈ្មោះ ចតុកោណកែង ត្រូវបានរកឃើញមុនគេ ដូចនេះគឺជាវិធីឈ្មោះ ផ្ទៃក្រឡា នៅក្នុងថ្នាក់នេះហើយដែលត្រូវយកទៅប្រើ។

នៅពេលដែលយើងបង្កើតវិធីនៅក្នុងថ្នាក់រងមានឈ្មោះដូចវិធីនៅក្នុងថ្នាក់មេ គេនិយាយថា យើងបង្កើតវិធីនៅក្នុងថ្នាក់រង *បាំង* (override) វិធីនៅក្នុងថ្នាក់មេ។ ក្នុងករណីនេះ វិធីនៅក្នុង ថ្នាក់រងត្រូវហៅថា *វិធីបាំងគេ* (overriding method) និងវិធីនៅក្នុងថ្នាក់មេត្រូវហៅថា *វិធីត្រូវគេបាំង* (overridden method) ។

មួយវិញទៀត ក្នុងករណីដែលថ្នាក់មេនិងថ្នាក់រងមានសម្បត្តិមានឈ្មោះដូចគ្នា នៅពេលដែល យើងយកសម្បត្តិឈ្មោះដូចគ្នានោះទៅប្រើ សម្បត្តិដែលត្រូវយកទៅប្រើគឺអាស្រ័យទៅលើ សិស្សដែលតាមរយៈវាសម្បត្តិនោះត្រូវយកទៅប្រើ។ ជាក់ស្តែងនៅក្នុងកម្មវិធីខាងលើ ការយក វិធីឈ្មោះ ផ្ទៃក្រឡា នៃថ្នាក់ណាមួយមកប្រើគឺអាស្រ័យទៅលើសិស្សដែលតាមរយៈវាវិធីនោះ ត្រូវយកទៅប្រើ។ គឺថាបើសិស្សជាសិស្សនៃថ្នាក់ឈ្មោះ ត្រីកោណ គឺវិធីឈ្មោះ ផ្ទៃក្រឡា នៅ ក្នុងថ្នាក់ឈ្មោះ ត្រីកោណ នោះដែលត្រូវយកទៅប្រើ។ តែបើសិស្សជាសិស្សនៃថ្នាក់ឈ្មោះ ចតុកោណកែង វិញ គឺជាវិធីឈ្មោះ ផ្ទៃក្រឡា នៅក្នុងថ្នាក់ឈ្មោះ ចតុកោណកែង នោះដែល ត្រូវយកទៅប្រើ។ ដូចនេះវិធីឈ្មោះ ផ្ទៃក្រឡា អាចប្រែក្រឡាជាវិធីនៃថ្នាក់ណាមួយក៏បានដែរ គឺអាស្រ័យទៅលើសិស្សដែលតាមរយៈវាវិធីនោះត្រូវយកទៅប្រើ។ នៅក្នុងវិស័យព័ត៌មានវិទ្យា

ភាពដែលអាចប្រែក្រឡាបាននេះហៅថា *លក្ខណៈប្រែរូប* (polymorphism) ដែលជាចំណុចដ៏សំខាន់មួយនៅក្នុងការសរសេរកម្មវិធីដោយបង្កើតថ្នាក់ផ្សេងៗ (OOP) ។

យើងបានដឹងរួចមកហើយថា នៅពេលដែលយើងយកវិធីមានឈ្មោះដូចគ្នាមកប្រើតាមរយៈសិស្សនៃថ្នាក់រង គឺជាវិធីបាំងគេនៅក្នុងថ្នាក់រងដែលត្រូវយកទៅប្រើ ព្រោះវាត្រូវបានរកឃើញមុនគេនៅទីនោះ។ ក៏ប៉ុន្តែ បើសិនជាយើងចង់ឲ្យវិធីត្រូវគេបាំងនៅក្នុងថ្នាក់មេត្រូវយកទៅប្រើដែរនៅពេលដែលវិធីបាំងគេនៅក្នុងថ្នាក់រងត្រូវយកទៅប្រើ យើងត្រូវសរសេរកម្មវិធីដូចខាងក្រោមនេះ៖

```
class ក្រឡាផ្ទៃ() :
    ពី = 3.14
    def __init__(សិស្ស, *វិមាត្រ) :
        សិស្ស.វិមាត្រ = វិមាត្រ

    def ផ្ទៃក្រឡា(សិស្ស) :
        return សិស្ស.វិមាត្រ

class ចតុកោណកែង(ក្រឡាផ្ទៃ) :
    def __init__(សិស្ស, បណ្តោយ, ទទឹង) :
        super().__init__(បណ្តោយ, ទទឹង)

    def ផ្ទៃក្រឡា(សិស្ស) :
        ចតុវិមាត្រ = super().ផ្ទៃក្រឡា()
        ផ្ទៃ = ចតុវិមាត្រ[0] * ចតុវិមាត្រ[1]
        return ផ្ទៃ

class ត្រីកោណ(ចតុកោណកែង) :
```

```
def __init__(សិស្ស, បាត, កំពស់) :
    super().__init__(បាត, កំពស់)

def ផ្ទៃក្រឡា(សិស្ស) :
    ផ្ទៃ = super().ផ្ទៃក្រឡា() / 2
    print("ក្រឡាផ្ទៃត្រីកោណគឺ៖", ផ្ទៃ)

សិស្សត្រីកោណ = ត្រីកោណ(25, 5)
សិស្សត្រីកោណ.ផ្ទៃក្រឡា()
```

super().__init__(បណ្តោយ, ទទឹង) គឺជាការយកស្ថាបនិកត្រូវគេបាំងនៅក្នុងថ្នាក់មេឈ្មោះ ក្រឡាផ្ទៃ មកប្រើនៅក្នុងស្ថាបនិកបាំងគេនៅក្នុងថ្នាក់រងឈ្មោះ ចតុកោណកែង ។ ដូចនេះនៅពេលដែលស្ថាបនិកបាំងគេនៅក្នុងថ្នាក់រងឈ្មោះ ចតុកោណកែង ត្រូវយកទៅប្រើ ស្ថាបនិកត្រូវគេបាំងនៅក្នុងថ្នាក់មេឈ្មោះ ក្រឡាផ្ទៃ ក៏ត្រូវយកទៅប្រើដែរ។

ចតុវិមាត្រ = super().ផ្ទៃក្រឡា() គឺជាបញ្ហាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកវិធីត្រូវគេបាំងឈ្មោះ ផ្ទៃក្រឡា នៅក្នុងថ្នាក់មេឈ្មោះ ក្រឡាផ្ទៃ មកប្រើនៅក្នុងវិធីបាំងគេឈ្មោះ ផ្ទៃក្រឡា នៅក្នុងថ្នាក់រងឈ្មោះ ចតុកោណកែង ។ ដូចនេះនៅពេលដែលវិធីបាំងគេឈ្មោះ ផ្ទៃក្រឡា នៅក្នុងថ្នាក់រងឈ្មោះ ចតុកោណកែង ត្រូវយកទៅប្រើ វិធីត្រូវគេបាំងឈ្មោះ ផ្ទៃក្រឡា នៅក្នុងថ្នាក់មេឈ្មោះ ក្រឡាផ្ទៃ ក៏ត្រូវយកទៅប្រើដែរ។

super().__init__(បាត, កំពស់) គឺជាការយកស្ថាបនិកត្រូវគេបាំងនៅក្នុងថ្នាក់មេឈ្មោះ ចតុកោណកែង មកប្រើនៅក្នុងស្ថាបនិកបាំងគេនៅក្នុងថ្នាក់រងឈ្មោះ ត្រីកោណ ។ ដូចនេះនៅពេលដែលស្ថាបនិកបាំងគេនៅក្នុងថ្នាក់រងឈ្មោះ ត្រីកោណ ត្រូវយកទៅប្រើ ស្ថាបនិកត្រូវគេបាំងនៅក្នុងថ្នាក់មេឈ្មោះ ចតុកោណកែង ក៏ត្រូវយកទៅប្រើដែរ។

ផ្ទៃ = super().*ផ្ទៃក្រឡា*() / 2 គឺជាបញ្ហាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកវិធីត្រូវគេបាំងឈ្មោះ ផ្ទៃក្រឡា នៅក្នុងថ្នាក់មេឈ្មោះ ចតុកោណកែង មកប្រើនៅក្នុងថ្នាក់រងឈ្មោះ ត្រីកោណ ។ ដូចនេះនៅពេលដែលវិធីបាំងគេឈ្មោះ ផ្ទៃក្រឡា នៅក្នុងថ្នាក់រងឈ្មោះ ត្រីកោណ ត្រូវយកទៅប្រើ វិធីត្រូវគេបាំងឈ្មោះ ផ្ទៃក្រឡា នៅក្នុងថ្នាក់មេឈ្មោះ ចតុកោណកែង ក៏ត្រូវយកទៅប្រើដែរ។

សិស្សត្រីកោណ = *ត្រីកោណ*(25, 5) គឺជាការយកថ្នាក់ឈ្មោះ ត្រីកោណ មកប្រើដើម្បីបង្កើតសិស្សឈ្មោះ សិស្សត្រីកោណ ម្នាក់។ ប្រការនេះធ្វើឲ្យស្ថាបនិកនៃថ្នាក់ឈ្មោះ ត្រីកោណ ត្រូវយកមកប្រើជាស្វ័យប្រវត្តិ ដែលជាហេតុបណ្តាលឲ្យស្ថាបនិកនៃថ្នាក់មេឈ្មោះ ចតុកោណកែង ក៏ត្រូវយកមកប្រើដែរ។ ហើយនៅពេលដែលស្ថាបនិកនៃថ្នាក់ឈ្មោះ ចតុកោណកែង ត្រូវយកមកប្រើ ស្ថាបនិកនៃថ្នាក់មេឈ្មោះ ក្រឡាផ្ទៃ ក៏ត្រូវយកមកប្រើដែរ។

និយាយរួម ការយកស្ថាបនិកនៃថ្នាក់រងឈ្មោះ ត្រីកោណ មកប្រើ ជាហេតុបណ្តាលឲ្យស្ថាបនិកនៃថ្នាក់មេឈ្មោះ ចតុកោណកែង និងថ្នាក់មេឈ្មោះ ក្រឡាផ្ទៃ ក៏ត្រូវយកទៅប្រើជាបន្តបន្ទាប់ជាប់គ្នារយោងដូចជាសម្លរក្រពិស។

សិស្សត្រីកោណ.ផ្ទៃក្រឡា() គឺជាការយកវិធីឈ្មោះ ផ្ទៃក្រឡា មកប្រើតាមរយៈសិស្សឈ្មោះ សិស្សត្រីកោណ ។ អាស្រ័យទៅតាមវិធាននៃការស្វែងរកសម្បត្តិ វិធីបាំងគេឈ្មោះ ផ្ទៃក្រឡា នៅក្នុងថ្នាក់របស់សិស្សឈ្មោះ សិស្សត្រីកោណ ត្រូវយកបានយកមកប្រើ។ ហើយនៅពេលដែលវិធីចុងក្រោយនេះត្រូវយកទៅប្រើ វិធីមានឈ្មោះ ផ្ទៃក្រឡា ដូចគ្នានៅក្នុងថ្នាក់មេឈ្មោះ ចតុកោណកែង និងនៅក្នុងថ្នាក់មេឈ្មោះ ក្រឡាផ្ទៃ ក៏ត្រូវយកទៅប្រើដែរ។

និយាយរួម ការយកវិធីឈ្មោះ ផ្ទៃក្រឡា នៅក្នុងថ្នាក់រងឈ្មោះ ត្រីកោណ មកប្រើ ជាហេតុបណ្តាលឲ្យវិធីមានឈ្មោះដូចគ្នានៅក្នុងថ្នាក់មេឈ្មោះ ចតុកោណកែង និងថ្នាក់មេឈ្មោះក្រឡាផ្ទៃ ក៏ត្រូវយកទៅប្រើជាបន្តបន្ទាប់ជាប់គ្នារយោងដូចសម្លូវក្រពិស។។

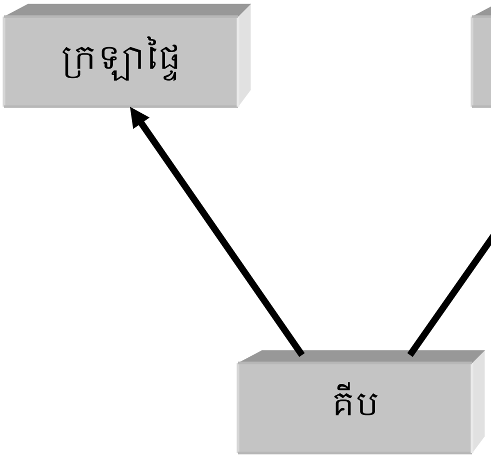
ពហុបន្តថ្នាក់

ពហុបន្តថ្នាក់ (multiple inheritance) គឺជាការបង្កើតថ្នាក់មួយបន្តភ្ជាប់ទៅនឹងថ្នាក់ជាច្រើនទៀត។ ដើម្បីបង្កើតថ្នាក់មួយបន្តភ្ជាប់ទៅនឹងថ្នាក់ជាច្រើនទៀត យើងត្រូវសរសេរកម្មវិធីដូចខាងក្រោមនេះ៖

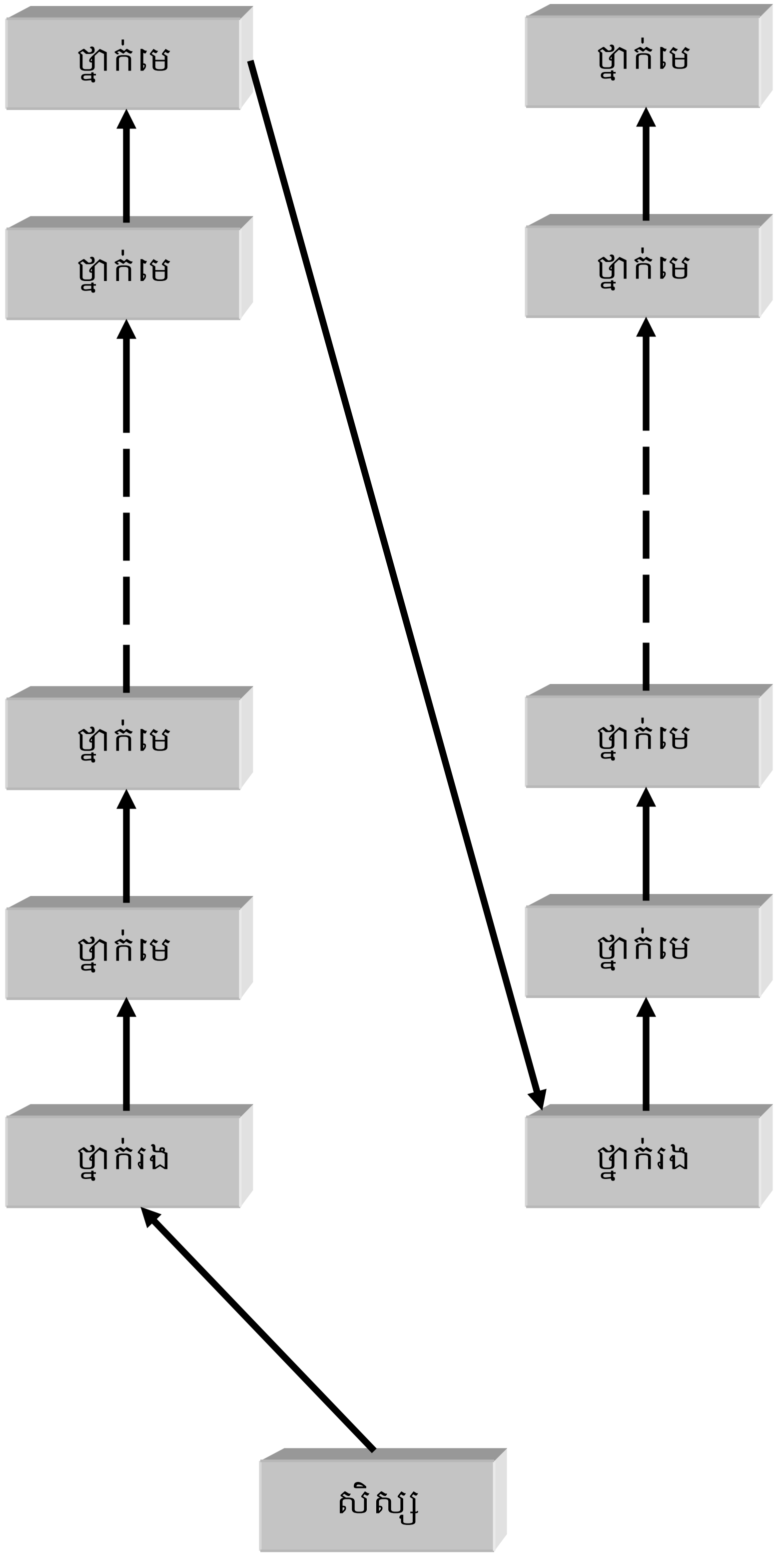
```
class ក្រឡាផ្ទៃ() :
    ពី = 3.14
    def __init__(សិស្ស, *វិមាត្រ) :
        សិស្ស.វិមាត្រ = វិមាត្រ

    def ផ្ទៃក្រឡា(សិស្ស) :
        return សិស្ស.វិមាត្រ

class មាឌ() :
    ពី = 3.1415
    def __init__(សិស្ស, *វិមាត្រ) :
        សិស្ស.វិមាត្រ = វិមាត្រ

    def មាឌ(សិស្ស) :
        return សិស្ស.វិមាត្រ
```

```
class គីប(ក្រឡាផ្ទៃ, មាឌ) :
    def __init__(សិស្ស, *វិមាត្រ) :
        សិស្ស.វិមាត្រ = វិមាត្រ

    def ផ្ទៃក្រឡា(សិស្ស) :
        ផ្ទៃ = សិស្ស.វិមាត្រ[0] * សិស្ស.វិមាត្រ[1] * 6
        print("ក្រឡាផ្ទៃគីបគឺ៖", ផ្ទៃ)

    def មាឌ(សិស្ស) :
        មាឌគីប = សិស្ស.វិមាត្រ[0] * សិស្ស.វិមាត្រ[1] * សិស្ស.វិមាត្រ[2]
        print("មាឌរបស់គីបគឺ៖", មាឌគីប)
```

class គីប(ក្រឡាផ្ទៃ, មាឌ) : គឺជាការបង្កើតថ្នាក់រងឈ្មោះ គីប មួយបន្តភ្ជាប់ទៅនឹងថ្នាក់មេឈ្មោះ ក្រឡាផ្ទៃ និងថ្នាក់មេឈ្មោះ មាឌ ។



ក្នុងករណីមានពហុបន្តថ្នាក់ នៅពេលដែលសម្បត្តិផ្សេងៗត្រូវយកទៅប្រើ ការស្វែងរកសម្បត្តិទាំងនោះត្រូវធ្វើឡើងដូចនៅក្នុងគំនូរបំព្រួញដូចខាងក្រោមនេះ៖


ថ្នាក់មេ
ថ្នាក់មេ
ថ្នាក់មេ
ថ្នាក់មេ
ថ្នាក់មេ
ថ្នាក់មេ
ថ្នាក់មេ
ថ្នាក់មេ
ថ្នាក់រង
ថ្នាក់រង
សិស្ស

គឺថាបើការយកសម្បត្តិទៅប្រើត្រូវធ្វើឡើងតាមរយៈសិស្សនៃថ្នាក់រង ការស្វែងរកសម្បត្តិនោះនឹងត្រូវធ្វើឡើងនៅក្នុងសិស្សនោះមុន រួចបានឡើងទៅថ្នាក់របស់សិស្សនោះ រួចបានឡើងទៅថ្នាក់មេនៃថ្នាក់របស់សិស្សនោះជាបន្តបន្ទាប់តាមសញ្ញាព្រួញរហូតដល់សម្បត្តិនោះត្រូវរកឃើញទើបឈប់។ ហើយក្នុងករណីដែលមានសម្បត្តិមានឈ្មោះដូចគ្នាជាច្រើននៅក្នុងថ្នាក់ផ្សេងៗ សម្បត្តិដែលត្រូវយកមកប្រើគឺជាសម្បត្តិដែលត្រូវរកឃើញមុនគេ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
class ក្រឡាផ្ទៃ() :
    ពី = 3.14
    def __init__(សិស្ស, *វិមាត្រ) :
        សិស្ស.វិមាត្រ = វិមាត្រ

    def ផ្ទៃក្រឡា(សិស្ស) :
        return សិស្ស.វិមាត្រ

class មាឌ() :
    ពី = 3.1415
    def __init__(សិស្ស, *វិមាត្រ) :
        សិស្ស.វិមាត្រ = វិមាត្រ

    def មាឌ(សិស្ស) :
        return សិស្ស.វិមាត្រ

class គីប(ក្រឡាផ្ទៃ, មាឌ) :
    def __init__(សិស្ស, *វិមាត្រ) :
        សិស្ស.វិមាត្រ = វិមាត្រ
```

```
    def ផ្ទៃក្រឡា(សិស្ស) :
        ផ្ទៃ = សិស្ស.វិមាត្រ[0] * សិស្ស.វិមាត្រ[1] * 6
        print("ក្រឡាផ្ទៃគីបគឺ៖", ផ្ទៃ)

    def មាឌ(សិស្ស) :
        មាឌគីប = សិស្ស.វិមាត្រ[0] * សិស្ស.វិមាត្រ[1] * សិស្ស.វិមាត្រ[2]
        print("មាឌរបស់គីបគឺ៖", មាឌគីប)

សិស្សគីប = គីប(25, 5, 10)
សិស្សគីប.ផ្ទៃក្រឡា()
សិស្សគីប.មាឌ()
print(សិស្សគីប.ពី)
```

សិស្សគីប = គីប(25, 5, 10) គឺជាការយកថ្នាក់ឈ្មោះ គីប មកប្រើដើម្បីបង្កើតសិស្សឈ្មោះ សិស្សគីប ម្នាក់។ ប្រការនេះបណ្តាលឲ្យស្ថាបនិកត្រូវយកមកប្រើជាស្វ័យប្រវត្តិ។ ការស្វែងរកស្ថាបនិកដែលជាវិធីឈ្មោះ __init__ ត្រូវធ្វើឡើងជាដំបូងនៅក្នុងសិស្សឈ្មោះ សិស្សគីប នោះមុន រួចបានឡើងទៅថ្នាក់របស់សិស្សនោះជាក្រោយ រួចបានឡើងទៅថ្នាក់មេនៃថ្នាក់របស់សិស្សនោះជាបន្តបន្ទាប់រហូតដល់ស្ថាបនិកត្រូវរកឃើញ។ ហើយគឺស្ថាបនិកនៅក្នុងថ្នាក់ឈ្មោះ គីប ដែលត្រូវបានយកមកប្រើ ព្រោះវាត្រូវបានរកឃើញមុនគេ។

សិស្សគីប.ផ្ទៃក្រឡា() គឺជាការយកវិធីឈ្មោះ ផ្ទៃក្រឡា មកប្រើតាមរយៈសិស្សឈ្មោះ សិស្សគីប ។ ការស្វែងរកវិធីឈ្មោះ ផ្ទៃក្រឡា ត្រូវធ្វើឡើងជាដំបូងនៅក្នុងសិស្សនោះមុន រួចបានឡើងទៅថ្នាក់របស់សិស្សនោះ រួចបានឡើងទៅថ្នាក់មេនៃថ្នាក់របស់សិស្សនោះជាបន្តបន្ទាប់រហូតដល់វិធីមានឈ្មោះដូចនោះត្រូវរកឃើញ។ ហើយគឺវិធីឈ្មោះ ផ្ទៃក្រឡា នៅក្នុងថ្នាក់ឈ្មោះ គីប ដែលត្រូវយកមកប្រើ ព្រោះវាត្រូវបានរកឃើញនៅទីនោះមុនគេ។

សិស្សគីប.មាឌ() គឺជាការយកវិធីឈ្មោះ មាឌ មកប្រើតាមរយៈសិស្សឈ្មោះ សិស្សគីប ។ ការស្វែងរកវិធីមានឈ្មោះដូចនេះត្រូវធ្វើឡើងជាដំបូងនៅក្នុងសិស្សនោះជាមុនសិន រួចបានឡើងទៅថ្នាក់របស់សិស្សនោះ រួចបានឡើងទៅថ្នាក់មេនៃថ្នាក់របស់សិស្សនោះជាបន្តបន្ទាប់ រហូតដល់វិធីនោះត្រូវរកឃើញ។ ហើយគឺវិធីឈ្មោះ មាឌ នៅក្នុងថ្នាក់ឈ្មោះ គីប ដែលត្រូវយកមកប្រើ ព្រោះវាត្រូវបានរកឃើញនៅទីនោះមុនគេ។

print(*សិស្សគីប.ពី*) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកសម្បត្តិឈ្មោះ ពី មកប្រើតាមរយៈសិស្សឈ្មោះ សិស្សគីប ។ ការស្វែងរកសម្បត្តិមានឈ្មោះដូចនោះត្រូវធ្វើឡើងនៅក្នុងសិស្សនោះជាមុនសិន រួចបានឡើងទៅថ្នាក់របស់នោះជាក្រោយ រួចបានឡើងទៅថ្នាក់មេនៃថ្នាក់របស់សិស្សនោះជាបន្តបន្ទាប់រហូតដល់សម្បត្តិមានឈ្មោះដូចនោះត្រូវរកឃើញ។ ហើយគឺជាទិន្នន័យគម្រូឈ្មោះ មាឌ នៅក្នុងថ្នាក់ឈ្មោះ ក្រឡាផ្ទៃ ដែលត្រូវបានយកទៅប្រើ ព្រោះវាត្រូវបានរកឃើញនៅក្នុងថ្នាក់នោះមុនគេ។

ចំពោះវិធីត្រូវគេបាំងនៅក្នុងថ្នាក់មេ បើសិនជាយើងចង់យកវិធីទាំងនោះទៅប្រើនៅក្នុងថ្នាក់រង យើងត្រូវសរសេរកម្មវិធីដូចខាងក្រោមនេះ៖

```
class ក្រឡាផ្ទៃ() :
    ពី = 3.14
    def __init__(សិស្ស, *វិមាត្រ) :
        សិស្ស.វិមាត្រផ្ទៃក្រឡា = វិមាត្រ

    def ផ្ទៃក្រឡា(សិស្ស) :
        return សិស្ស.វិមាត្រផ្ទៃក្រឡា

class មាឌ() :
```

```
    ពី = 3.1415
    def __init__(សិស្ស, *វិមាត្រ) :
        សិស្ស.វិមាត្រមាឌ = វិមាត្រ

    def មាឌ(សិស្ស) :
        return សិស្ស.វិមាត្រមាឌ

class គីប(ក្រឡាផ្ទៃ, មាឌ) :
    def __init__(សិស្ស, បណ្តោយ, ទទឹង, កំពស់) :
        super().__init__(បណ្តោយ, ទទឹង)
        មាឌ.__init__(សិស្ស, បណ្តោយ, ទទឹង, កំពស់)

    def ផ្ទៃក្រឡា(សិស្ស) :
        វិមាត្រផ្ទៃក្រឡា = super().ផ្ទៃក្រឡា()
        ផ្ទៃ = វិមាត្រផ្ទៃក្រឡា[0] * វិមាត្រផ្ទៃក្រឡា[1] * 6
        print("ក្រឡាផ្ទៃគីបគឺ៖", ផ្ទៃ)

    def មាឌ(សិស្ស) :
        វិមាត្រមាឌ = super().មាឌ()
        មាឌគីប = វិមាត្រមាឌ[0] * វិមាត្រមាឌ[1] * វិមាត្រមាឌ[2]
        print("មាឌរបស់គីបគឺ៖", មាឌគីប)

សិស្សគីប = គីប(25, 5, 10)
សិស្សគីប.ផ្ទៃក្រឡា()
សិស្សគីប.មាឌ()
```

super().__init__(*បណ្តោយ, ទទឹង*) គឺជាការយកស្ថាបនិកនៅក្នុងថ្នាក់មេឈ្មោះ ក្រឡាផ្ទៃ មកប្រើនៅក្នុងស្ថាបនិកនៅក្នុងថ្នាក់រងឈ្មោះ គីប ។ ដូចនេះនៅពេលដែលស្ថាបនិកនៅក្នុងថ្នាក់ឈ្មោះ គីប ត្រូវយកទៅប្រើ ស្ថាបនិកនៅក្នុងថ្នាក់ឈ្មោះ ក្រឡាផ្ទៃ ក៏ត្រូវយកទៅប្រើដែរ។

មាឌ.__init__(*សិស្ស, បណ្តោយ, ទទឹង, កំពស់*) គឺជាការយកស្ថាបនិកនៅក្នុងថ្នាក់មេឈ្មោះ មាឌ មកប្រើនៅក្នុងស្ថាបនិកនៅក្នុងថ្នាក់រងឈ្មោះ គីប ។ ដូចនេះនៅពេលដែលស្ថាបនិកនៅក្នុងថ្នាក់ឈ្មោះ គីប ត្រូវយកទៅប្រើ ស្ថាបនិកនៅក្នុងថ្នាក់ឈ្មោះ មាឌ ក៏ត្រូវយកទៅប្រើដែរ។

វិមាត្រផ្ទៃក្រឡា = super().*ផ្ទៃក្រឡា*() គឺជាបញ្ហាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកវិធីឈ្មោះ ផ្ទៃក្រឡា នៅក្នុងថ្នាក់មេឈ្មោះ ក្រឡាផ្ទៃ មកប្រើនៅក្នុងវិធីឈ្មោះ ផ្ទៃក្រឡា នៅក្នុងថ្នាក់រងឈ្មោះ គីប ។ ដូចនេះនៅពេលដែលវិធីឈ្មោះ ផ្ទៃក្រឡា នៅក្នុងថ្នាក់រងឈ្មោះ គីប ត្រូវយកទៅប្រើ វិធីមានឈ្មោះដូចគ្នានៅក្នុងថ្នាក់មេឈ្មោះ ក្រឡាផ្ទៃ ក៏ត្រូវយកទៅប្រើដែរ។

វិមាត្រមាឌ = super().*មាឌ*() គឺជាបញ្ហាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកវិធីឈ្មោះ មាឌ នៅក្នុងថ្នាក់មេឈ្មោះ មាឌ មកប្រើនៅក្នុងវិធីឈ្មោះ មាឌ នៅក្នុងថ្នាក់រងឈ្មោះ គីប ។ ដូចនេះនៅពេលដែលវិធីឈ្មោះ មាឌ នៅក្នុងថ្នាក់រងឈ្មោះ គីប ត្រូវយកទៅប្រើ វិធីមានឈ្មោះដូចគ្នានៅក្នុងថ្នាក់មេឈ្មោះ មាឌ ក៏ត្រូវយកទៅប្រើដែរ។

សិស្សគីប = *គីប*(25, 5, 10) គឺជាការយកថ្នាក់ឈ្មោះ គីប មកប្រើដើម្បីបង្កើតសិស្សឈ្មោះ សិស្សគីប ម្នាក់។ ប្រការនេះធ្វើឲ្យស្ថាបនិកនៅក្នុងថ្នាក់ឈ្មោះ គីប ត្រូវយកមកប្រើជាស្វ័យប្រវត្តិ។ ហើយនៅពេលដែលស្ថាបនិកនៃថ្នាក់ឈ្មោះ គីប ត្រូវយកមកប្រើ ស្ថាបនិកនៅក្នុងថ្នាក់មេឈ្មោះ ក្រឡាផ្ទៃ និងនៅក្នុងថ្នាក់មេឈ្មោះ មាឌ ក៏ត្រូវយកមកប្រើជាស្វ័យប្រវត្តិដែរ ។

សិស្សគីប.ផ្ទៃក្រឡា() គឺជាការយកវិធីឈ្មោះ ផ្ទៃក្រឡា មកប្រើតាមរយៈសិស្សឈ្មោះ សិស្សគីប ។ ប្រការនេះធ្វើឲ្យការស្វែងរកវិធីឈ្មោះ ផ្ទៃក្រឡា ត្រូវធ្វើឡើងជាដំបូងនៅក្នុងសិស្សនោះជាមុនសិន រួចបានឡើងទៅថ្នាក់របស់សិស្សនោះ រួចបានឡើងទៅថ្នាក់មេនៃថ្នាក់របស់សិស្សនោះជាបន្តបន្ទាប់។ វិធីឈ្មោះ ផ្ទៃក្រឡា នៅក្នុងថ្នាក់ឈ្មោះ គីប ត្រូវបានយកមកប្រើ ព្រោះវាត្រូវបានរកឃើញនៅក្នុងថ្នាក់នោះមុនគេបង្អស់។

នៅពេលដែលវិធីឈ្មោះ ផ្ទៃក្រឡា នៃថ្នាក់ឈ្មោះ គីប ត្រូវយកមកប្រើ វិធីមានឈ្មោះដូចគ្នានៅក្នុងថ្នាក់ឈ្មោះ ក្រឡាផ្ទៃ ក៏ត្រូវយកមកប្រើដែរ។

សិស្សគីប.មាឌ() គឺជាការយកវិធីឈ្មោះ មាឌ មកប្រើតាមរយៈសិស្សឈ្មោះ សិស្សគីប ។ ប្រការនេះធ្វើឲ្យការស្វែងរកវិធីឈ្មោះ មាឌ ត្រូវធ្វើឡើងជាដំបូងនៅក្នុងសិស្សនោះជាមុនសិន រួចបានឡើងទៅថ្នាក់របស់សិស្សនោះ រួចបានឡើងទៅថ្នាក់មេនៃថ្នាក់របស់សិស្សនោះជាបន្តបន្ទាប់។ វិធីឈ្មោះ មាឌ នៅក្នុងថ្នាក់ឈ្មោះ គីប ត្រូវបានយកមកប្រើ ព្រោះវាត្រូវបានរកឃើញនៅក្នុងថ្នាក់នោះមុនគេបង្អស់។

នៅពេលដែលវិធីឈ្មោះ មាឌ នៃថ្នាក់ឈ្មោះ គីប ត្រូវយកមកប្រើ វិធីមានឈ្មោះដូចគ្នានៅក្នុងថ្នាក់ឈ្មោះ មាឌ ក៏ត្រូវយកមកប្រើដែរ។

ការបន្តថ្នាក់ រាងចតុកោណស្មើ

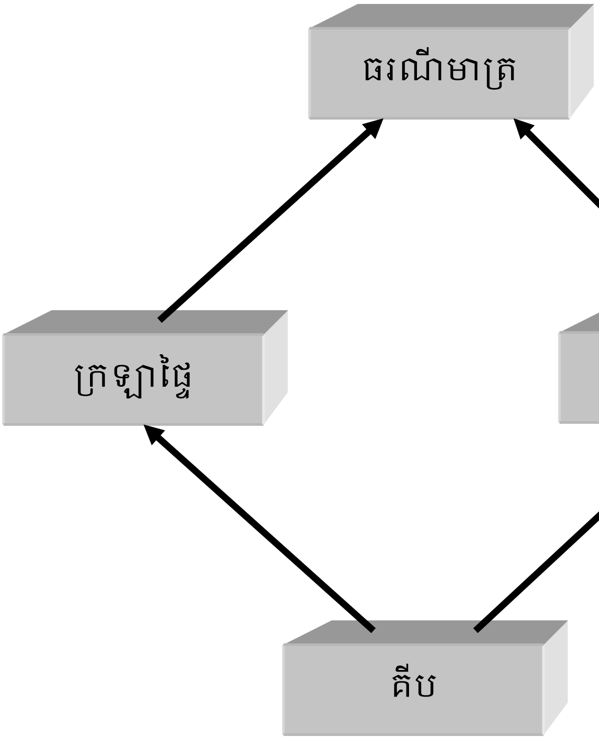
ការបន្តថ្នាក់រាងចតុកោណស្មើ (diamond shape) គឺជាការបង្កើតថ្នាក់មួយបន្តភ្ជាប់ទៅនឹងថ្នាក់ពីរទៀតដែលត្រូវបានបន្តភ្ជាប់ទៅនឹងថ្នាក់តែមួយដូចគ្នា។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
class ធរណីមាត្រ() :
    pi = 180
```

```
    def បង្ហាញព័ត៌មាន(សិស្ស, ព័ត៌មាន) :
        print(ព័ត៌មាន)

class ក្រឡាផ្ទៃ(ធរណីមាត្រ) :
    ពី = 3.14
    def __init__(សិស្ស, *វិមាត្រ) :
        សិស្ស.វិមាត្រផ្ទៃក្រឡា = វិមាត្រ

    def ផ្ទៃក្រឡា(សិស្ស) :
        return សិស្ស.វិមាត្រផ្ទៃក្រឡា

class មាឌ(ធរណីមាត្រ) :
    pi = 3.1415
    def __init__(សិស្ស, *វិមាត្រ) :
        សិស្ស.វិមាត្រមាឌ = វិមាត្រ

    def មាឌ(សិស្ស) :
        return សិស្ស.វិមាត្រមាឌ

class គីប(ក្រឡាផ្ទៃ, មាឌ) :
    def __init__(សិស្ស, បណ្តោយ, ទទឹង, កំពស់) :
        ក្រឡាផ្ទៃ.__init__(សិស្ស, បណ្តោយ, ទទឹង)
        មាឌ.__init__(សិស្ស, បណ្តោយ, ទទឹង, កំពស់)

    def ផ្ទៃក្រឡា(សិស្ស) :
        វិមាត្រផ្ទៃក្រឡា = ក្រឡាផ្ទៃ.ផ្ទៃក្រឡា(សិស្ស)
        ផ្ទៃ = វិមាត្រផ្ទៃក្រឡា[0] * វិមាត្រផ្ទៃក្រឡា[1] * 6
```

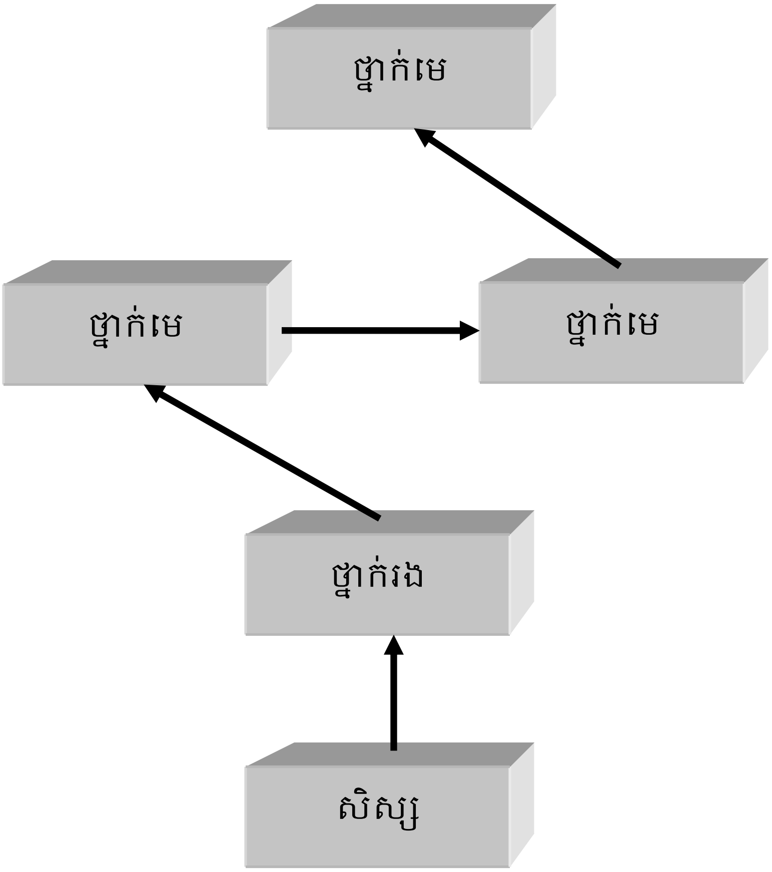
```
    print("ក្រឡាផ្ទៃគីបគឺ៖", ផ្ទៃ)

def មាឌ(សិស្ស) :
    វិមាត្រមាឌ = មាឌ.មាឌ(សិស្ស)
    មាឌគីប = វិមាត្រមាឌ[0] * វិមាត្រមាឌ[1] * វិមាត្រមាឌ[2]
    print("មាឌរបស់គីបគឺ៖", មាឌគីប)
```

នៅក្នុងកម្មវិធីខាងលើនេះ យើងឃើញថាថ្នាក់ឈ្មោះ គីប ត្រូវបានបង្កើតឡើងបន្តភ្ជាប់ទៅនឹងថ្នាក់ឈ្មោះ ក្រឡាផ្ទៃ និងថ្នាក់ឈ្មោះ មាឌ ។ ហើយថ្នាក់មេពីរចុងក្រោយនេះត្រូវបានបន្តភ្ជាប់ទៅនឹងថ្នាក់មេឈ្មោះ ធរណីមាត្រ តែមួយដូចគ្នា។ ដូចនេះ បើយើងគូសគំនូសបំព្រួញនៃការបន្តថ្នាក់ដូចនៅក្នុងកម្មវិធីខាងលើនេះ យើងនឹងបានរូបដូចខាងក្រោមនេះ៖



ការបន្តថ្នាក់ដូចនៅក្នុងរូបខាងលើនេះហៅថាការបន្តថ្នាក់រាងចតុកោណស្មើ ព្រោះទម្រង់របស់វាមានរាងជាចតុកោណស្មើ។

ក្នុងករណីមានការបន្តថ្នាក់មានរាងចតុកោណស្មើ នៅពេលដែលសម្បត្តិណាមួយត្រូវយកមកប្រើ ការស្វែងរកសម្បត្តិនោះត្រូវធ្វើឡើងទៅតាមគំនូសបំព្រួញដូចខាងក្រោមនេះ៖



គឺថានៅពេលដែលសម្បត្តិណាមួយត្រូវយកមកប្រើតាមរយៈសិស្សឬថ្នាក់ណាមួយ ការស្វែងរកសម្បត្តិនោះត្រូវធ្វើឡើងនៅក្នុងសិស្សឬថ្នាក់នោះជាមុនសិន រួចបានឡើងទៅថ្នាក់

ផ្សេងៗទៀតតាមសញ្ញាព្រួញរហូតដល់សម្បត្តិនោះត្រូវរកឃើញ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
class ធរណីមាត្រ() :
   pi = 180
   def បង្ហាញព័ត៌មាន(សិស្ស, ព័ត៌មាន) :
      print(ព័ត៌មាន)

class ក្រឡាផ្ទៃ(ធរណីមាត្រ) :
   ពី = 3.14
   def __init__(សិស្ស, *វិមាត្រ) :
      សិស្ស.វិមាត្រផ្ទៃក្រឡា = វិមាត្រ

   def ផ្ទៃក្រឡា(សិស្ស) :
      return សិស្ស.វិមាត្រផ្ទៃក្រឡា

class មាឌ(ធរណីមាត្រ) :
   pi = 3.1415
   def __init__(សិស្ស, *វិមាត្រ) :
      សិស្ស.វិមាត្រមាឌ = វិមាត្រ

   def មាឌ(សិស្ស) :
      return សិស្ស.វិមាត្រមាឌ

class គីប(ក្រឡាផ្ទៃ, មាឌ) :
   def __init__(សិស្ស, បណ្ដោយ, ទទឹង, កំពស់) :
      ក្រឡាផ្ទៃ.__init__(សិស្ស, បណ្ដោយ, ទទឹង)
```

```
        មាឌ.__init__(សិស្ស, បណ្ដោយ, ទទឹង, កំពស់)

    def ផ្ទៃក្រឡា(សិស្ស) :
        វិមាត្រផ្ទៃក្រឡា = ក្រឡាផ្ទៃ.ផ្ទៃក្រឡា(សិស្ស)
        ផ្ទៃ = វិមាត្រផ្ទៃក្រឡា[0] * វិមាត្រផ្ទៃក្រឡា[1] * 6
        print("ក្រឡាផ្ទៃគីបគឺ៖", ផ្ទៃ)

    def មាឌ(សិស្ស) :
        វិមាត្រមាឌ = មាឌ.មាឌ(សិស្ស)
        មាឌគីប = វិមាត្រមាឌ[0] * វិមាត្រមាឌ[1] * វិមាត្រមាឌ[2]
        print("មាឌរបស់គីបគឺ៖", មាឌគីប)

សិស្សគីប = គីប(25, 5, 10)
សិស្សគីប.ផ្ទៃក្រឡា()
សិស្សគីប.មាឌ()
print(សិស្សគីប.pi)
```

សិស្សគីប = គីប(25, 5, 10) គឺជាការយកថ្នាក់ឈ្មោះ គីប មកប្រើដើម្បីបង្កើតសិស្សឈ្មោះ សិស្សគីប ម្នាក់។ ប្រការនេះធ្វើឲ្យស្ថាបនិកត្រូវយកមកប្រើជាស្វ័យប្រវត្តិ។ ការស្វែងរកស្ថាបនិកដែលជាវិធីមានឈ្មោះថា __init__ ត្រូវធ្វើឡើងជាដំបូងនៅក្នុងសិស្សនោះជាមុនសិន រួចបានឡើងទៅថ្នាក់របស់សិស្សនោះ រួចបានឡើងទៅថ្នាក់មេនៃថ្នាក់របស់សិស្សនោះជាបន្តបន្ទាប់រហូតដល់ស្ថាបនិកនោះត្រូវរកឃើញ។ ហើយគឺស្ថាបនិកនៅក្នុងថ្នាក់ឈ្មោះ គីប ដែលត្រូវបានយកមកប្រើ ព្រោះវាត្រូវបានរកឃើញនៅក្នុងថ្នាក់នោះមុនគេ។

សិស្សគីប.ផ្ទៃក្រឡា() គឺជាការយកវិធីឈ្មោះ ផ្ទៃក្រឡា មកប្រើតាមរយៈសិស្សឈ្មោះ សិស្សគីប ។ ការស្វែងរកវិធីនេះត្រូវធ្វើឡើងនៅក្នុងសិស្សនោះមុន រួចបានឡើងទៅថ្នាក់របស់

សិស្សនោះ រួចបានឡើងទៅថ្នាក់មេនៃថ្នាក់របស់សិស្សនោះជាបន្តបន្ទាប់រហូតដល់វិធីឈ្មោះ ផ្ទៃក្រឡា នោះត្រូវរកឃើញ។ ហើយគឺជាវិធីឈ្មោះ ផ្ទៃក្រឡា នៅក្នុងថ្នាក់ឈ្មោះ គីប ដែលត្រូវបានយកមកប្រើព្រោះវាត្រូវបានរកឃើញនៅក្នុងថ្នាក់នោះមុនគេ។

សិស្សគីប.មាឌ() គឺជាការយកវិធីឈ្មោះ មាឌ មកប្រើតាមរយៈសិស្សឈ្មោះ សិស្សគីប ។ ការស្វែងរកវិធីនេះត្រូវធ្វើឡើងនៅក្នុងសិស្សនោះមុន រួចបានឡើងទៅថ្នាក់របស់សិស្សនោះ រួចបានឡើងទៅថ្នាក់មេនៃថ្នាក់របស់សិស្សនោះជាបន្តបន្ទាប់រហូតដល់វិធីឈ្មោះ មាឌ នោះត្រូវរកឃើញ។ ហើយគឺជាវិធីឈ្មោះ មាឌ នៅក្នុងថ្នាក់ឈ្មោះ គីប ដែលត្រូវបានយកមកប្រើព្រោះវាត្រូវបានរកឃើញនៅក្នុងថ្នាក់នោះមុនគេ។

print(*សិស្សគីប*.pi) គឺជាការយកសម្បត្តិឈ្មោះ pi មកប្រើតាមរយៈសិស្សឈ្មោះ សិស្សគីប ។ ការស្វែងរកវត្ថុនេះត្រូវធ្វើឡើងនៅក្នុងសិស្សនោះមុន រួចបានឡើងទៅថ្នាក់របស់សិស្សនោះ រួចបានឡើងទៅថ្នាក់មេនៃថ្នាក់របស់សិស្សនោះជាបន្តបន្ទាប់រហូតដល់សម្បត្តិឈ្មោះ pi នោះត្រូវរកឃើញ។ ហើយគឺជាទិន្នន័យគម្រូឈ្មោះ pi នៅក្នុងថ្នាក់ឈ្មោះ មាឌ ដែលត្រូវបានយកមកប្រើព្រោះវាត្រូវបានរកឃើញនៅក្នុងថ្នាក់នោះមុនគេ។

ថ្នាក់មានស្រាប់

ថ្នាក់មានស្រាប់ (built-in type) គឺជាថ្នាក់ទាំងឡាយណាដែលត្រូវបានបង្កើតឡើងរួចជាស្រេចទុកនៅក្នុងដែនកំណត់ទូទៅជាមួយនឹងក្បួនផ្សេងៗទៀត។

កន្លងមកយើងបានបង្កើតវត្ថុផ្សេងៗមានដូចជា ចំនួនគត់ ចំនួនទសភាគ និងសមាសវត្ថុមួយចំនួន។ វត្ថុទាំងអស់នោះគឺជាសិស្សនៃថ្នាក់មានស្រាប់ទាំងនោះ តែការបង្កើតសិស្សទាំងនោះមានលក្ខណៈខុសប្លែកពីការបង្កើតសិស្សនៃថ្នាក់ធម្មតា។

ដើម្បីឲ្យដឹងថាតើថ្នាក់របស់វត្ថុណាមួយជាអ្វីនោះ យើងត្រូវធ្វើដូចខាងក្រោមនេះ៖

```
ថ្លៃលក់ = 1000
ថ្លៃទិញ = 800.5
ពិត = True
ឃ្លា = "អ្នកមានរក្សាខ្សត់"
កម្រងថេរចម្រុះ = (100, 1.33, "នាម", True)
កម្រងអថេរចម្រុះ = [2.5, 300, "គោត្តនាម", False]
វចនានុក្រមចម្រុះ = {"ថ្លៃលក់":1000, "ថ្លៃទិញ":900}
សំណុំចម្រុះ = {3.5, True, "អ្នកប្រាជ្ញរក្សាខ្លៅ"}
print(type(ថ្លៃលក់))
print(type(ថ្លៃទិញ))
print(type(ពិត))
print(type(ឃ្លា))
print(type(កម្រងថេរចម្រុះ))
```

```
print(type(កម្រងអថេរចម្រុះ))
print(type(វចនានុក្រមចម្រុះ))
print(type(សំណុំចម្រុះ))
```

print(type(ថ្លៃលក់)) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកចំនួនគត់ឈ្មោះ ថ្លៃលក់ មកពិនិត្យមើលថាតើវាជាសិស្សនៃថ្នាក់ណាដែរ។ លទ្ធផលបញ្ជាក់ប្រាប់ថា ថ្នាក់របស់វត្ថុដែលជាចំនួនគត់គឺជាថ្នាក់មានស្រាប់ឈ្មោះ int ។

print(type(ថ្លៃទិញ)) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកចំនួនពិតឈ្មោះ ថ្លៃទិញ មកពិនិត្យមើលថាតើវាជាសិស្សនៃថ្នាក់ណាដែរ។ លទ្ធផលបញ្ជាក់ប្រាប់ថា ថ្នាក់របស់វត្ថុដែលជាចំនួនពិតគឺជាថ្នាក់មានស្រាប់ឈ្មោះ float ។

print(type(ពិត)) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកតក្កវត្ថុឈ្មោះ ពិត មកពិនិត្យមើលថាតើវាជាសិស្សនៃថ្នាក់ណាដែរ។ លទ្ធផលបញ្ជាក់ប្រាប់ថា ថ្នាក់របស់វត្ថុដែលជាតក្កវត្ថុគឺជាថ្នាក់មានស្រាប់ឈ្មោះ bool ។

print(type(ឃ្លា)) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកកម្រងអក្សរឈ្មោះ ឃ្លា មកពិនិត្យមើលថាតើវាជាសិស្សនៃថ្នាក់ណាដែរ។ លទ្ធផលបញ្ជាក់ប្រាប់ថា ថ្នាក់របស់វត្ថុដែលជាកម្រងអក្សរគឺជាថ្នាក់មានស្រាប់ឈ្មោះ str ។

print(type(កម្រងថេរចម្រុះ)) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកកម្រងថេរឈ្មោះ កម្រងថេរចម្រុះ មកពិនិត្យមើលថាតើវាជាសិស្សនៃថ្នាក់ណាដែរ។ លទ្ធផលបញ្ជាក់ប្រាប់ថា ថ្នាក់របស់វត្ថុដែលជាកម្រងថេរគឺជាថ្នាក់មានស្រាប់ឈ្មោះ tuple។

print(type(*កម្រងអថេរចម្រុះ*)) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកកម្រងអថេរឈ្មោះ កម្រងអថេរចម្រុះ មកពិនិត្យមើលថាតើវាជាសិស្សនៃថ្នាក់ណាដែរ។ លទ្ធផលបញ្ជាក់ប្រាប់ថា ថ្នាក់របស់វត្ថុដែលជាកម្រងអថេរគឺជាថ្នាក់មានស្រាប់ឈ្មោះ list ។

print(type(*កម្រងអថេរចម្រុះ*)) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកកម្រងអថេរឈ្មោះ កម្រងអថេរចម្រុះ មកពិនិត្យមើលថាតើវាជាសិស្សនៃថ្នាក់ណាដែរ។ លទ្ធផលបញ្ជាក់ប្រាប់ថា ថ្នាក់របស់វត្ថុដែលជាកម្រងអថេរគឺជាថ្នាក់មានស្រាប់ឈ្មោះ list ។

print(type(*វចនានុក្រមចម្រុះ*)) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកវចនានុក្រមឈ្មោះ វចនានុក្រមចម្រុះ មកពិនិត្យមើលថាតើវាជាសិស្សនៃថ្នាក់ណាដែរ។ លទ្ធផលបញ្ជាក់ប្រាប់ថា ថ្នាក់របស់វត្ថុដែលជាវចនានុក្រមគឺជាថ្នាក់មានស្រាប់ឈ្មោះ dict ។

print(type(*សំណុំចម្រុះ*)) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកសំណុំឈ្មោះ សំណុំចម្រុះ មកពិនិត្យមើលថាតើវាជាសិស្សនៃថ្នាក់ណាដែរ។ លទ្ធផលបញ្ជាក់ប្រាប់ថា ថ្នាក់របស់វត្ថុដែលជាសំណុំគឺជាថ្នាក់មានស្រាប់ឈ្មោះ set ។

ថ្នាក់ int

គ្រប់ចំនួនគត់ទាំងអស់គឺជាសិស្សនៃថ្នាក់មានស្រាប់ឈ្មោះ int ។ ដូចនេះក្រៅពីការបង្កើតចំនួនគត់ដូចដែលធ្លាប់បានធ្វើកន្លងមក យើងក៏អាចបង្កើតចំនួនគត់ដោយយកថ្នាក់មានស្រាប់ឈ្មោះ int មកប្រើបានដែរ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
ថ្លៃលក់ = int(1000)
ថ្លៃទិញ = int(800.5)
print(ថ្លៃលក់)
```

```
print(ថ្លៃទិញ)
```

ថ្លៃលក់ = int(1000) គឺជាការយកថ្នាក់មានស្រាប់ឈ្មោះ int មកប្រើដើម្បីបង្កើតចំនួនគត់ឈ្មោះ ថ្លៃលក់ មួយដែលជាលេខ 1000 ។

ថ្លៃទិញ = int(800.5) គឺជាការយកថ្នាក់មានស្រាប់ឈ្មោះ int មកប្រើដើម្បីបង្កើតចំនួនគត់ឈ្មោះ ថ្លៃលក់ មួយដែលជាលេខ 800 ។

ដូចនេះយើងឃើញថា នៅពេលដែលយើងយកថ្នាក់មានស្រាប់ឈ្មោះ int មកប្រើដើម្បីបង្កើតចំនួនគត់ យើងចាំបាច់ត្រូវផ្តល់ដំណឹងជាលេខណាមួយឲ្យទៅស្ថាបនិកនៃថ្នាក់ឈ្មោះ int នោះ ។ ហើយលេខទាំងនោះនឹងត្រូវ *កែ* (convert) ឲ្យទៅជាចំនួនគត់។ អាស្រ័យហេតុនេះ យើងអាចប្រើថ្នាក់ឈ្មោះ int នេះសម្រាប់កែវត្ថុផ្សេងៗឲ្យទៅជាចំនួនគត់បាន។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
ថ្លៃលក់ = int(1000.33)
ថ្លៃទិញ = int("800")
print(ថ្លៃលក់)
print(ថ្លៃទិញ)
```

ថ្លៃលក់ = int(1000.33) គឺជាការយកថ្នាក់មានស្រាប់ឈ្មោះ int មកប្រើដើម្បីកែលេខ 1000.33 ឲ្យទៅជាចំនួនគត់ 1000 ។

ថ្លៃទិញ = int("800") គឺជាការយកថ្នាក់មានស្រាប់ឈ្មោះ int មកប្រើដើម្បីកែកម្រងអក្សរ "800" ឲ្យទៅជាចំនួនគត់ 800 ។

ដោយថ្នាក់មានស្រាប់ឈ្មោះ int ក៏ជាថ្នាក់មួយដូចជាថ្នាក់ដទៃទៀតដែរ ដូចនេះនៅក្នុងថ្នាក់នេះត្រូវតែមានវិធីមួយចំនួនដែលអាចត្រូវយកទៅប្រើប្រាស់នៅក្នុងការធ្វើប្រមាណវិធីផ្សេងៗដែលទាក់ទងទៅនឹងចំនួនគត់។ ដើម្បីពិនិត្យមើលសម្បត្តិផ្សេងៗដែលមាននៅក្នុងថ្នាក់មានស្រាប់ឈ្មោះ int យើងត្រូវធ្វើដូចខាងក្រោមនេះ៖

```
help(int)
```

help(int) គឺជាបញ្ជាតម្រូវឲ្យយកថ្នាក់មានស្រាប់ឈ្មោះ int មកពិនិត្យមើលសម្បត្តិផ្សេងៗដែលមាននៅក្នុងនោះ។

យើងឃើញថា វិធីទាំងអស់នៅក្នុងថ្នាក់មានស្រាប់ឈ្មោះ int សុទ្ធតែមានសញ្ញា _ នេះពីរនៅពីមុខនឹងនៅពីក្រោយ។ វិធីទាំងនោះគឺជាវិធីពិសេសដែលត្រូវទៅប្រើជាស្វ័យប្រវត្តិនៅក្នុងកាលៈទេសៈណាមួយច្បាស់លាស់។ យើងនឹងធ្វើការសិក្សាពីវិធីទាំងយ៉ាងលម្អិតនៅពេលខាងមុខនេះ។

ថ្នាក់ *float*

ដូចគ្នានឹងចំនួនគត់ដែរ វត្ថុដែលជាចំនួនពិតទាំងអស់គឺជាសិស្សនៃថ្នាក់មានស្រាប់ឈ្មោះ float ។ ហើយដើម្បីបង្កើតចំនួនពិតដោយយកថ្នាក់មានស្រាប់ឈ្មោះ float មកប្រើ យើងត្រូវធ្វើដូចខាងក្រោមនេះ៖

```
ថ្លៃលក់ = float(1000.33)
ថ្លៃទិញ = float(800)
print(ថ្លៃលក់)
print(ថ្លៃទិញ)
```

ថ្លៃលក់ = float(1000.33) គឺជាការយកថ្នាក់មានស្រាប់ឈ្មោះ float មកប្រើដើម្បីបង្កើតចំនួនពិតមួយមានឈ្មោះថា ថ្លៃលក់ ដែលជាលេខ 1000.33 ។

ថ្លៃលក់ = float(800) គឺជាការយកថ្នាក់មានស្រាប់ឈ្មោះ float មកប្រើដើម្បីបង្កើតចំនួនពិតមួយមានឈ្មោះថា ថ្លៃទិញ ដែលជាលេខ 800.0 ។

ដូចនេះយើងឃើញថា ដើម្បីបង្កើតចំនួនពិតដោយប្រើថ្នាក់មានស្រាប់ឈ្មោះ float យើងត្រូវផ្តល់ដំណឹងជាលេខឲ្យទៅស្ថាបនិកនៃថ្នាក់នោះ។ អាស្រ័យហេតុនេះ យើងអាចប្រើប្រាស់ថ្នាក់ឈ្មោះ float នេះសម្រាប់កែវត្ថុមួយចំនួនឲ្យទៅជាចំនួនពិត។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ

```
ថ្លៃលក់ = float(1000)
ថ្លៃទិញ = float("800")
print(ថ្លៃលក់)
print(ថ្លៃទិញ)
```

ថ្លៃលក់ = float(1000) គឺជាការប្រើថ្នាក់មានស្រាប់ឈ្មោះ float ដើម្បីកែចំនួនគត់ដែលជាលេខ 1000 ឲ្យទៅជាចំនួនពិត 1000.0 ។

ថ្លៃទិញ = float("800") គឺជាការប្រើថ្នាក់មានស្រាប់ឈ្មោះ float ដើម្បីកែកម្រងអក្សរ "800" ឲ្យទៅជាចំនួនពិត 800.0 ។

យើងអាចយកថ្នាក់មានស្រាប់ឈ្មោះ float មកពិនិត្យមើលដោយធ្វើដូចខាងក្រោមនេះ៖

```
help(float)
```

help(float) គឺជាបញ្ជាតម្រូវឲ្យយកថ្នាក់មានស្រាប់ឈ្មោះ float មកពិនិត្យមើល។

ដូចគ្នានឹងថ្នាក់មានស្រាប់ឈ្មោះ int ដែរ វិធីផ្សេងៗនៅក្នុងថ្នាក់មានស្រាប់ឈ្មោះ float សុទ្ធតែមានសញ្ញា _ ពីរនៅពីមុខនិងនៅពីក្រោយ។ ដូចនេះវិធីទាំងនោះគឺជាវិធីពិសេសដែលនឹងត្រូវយកទៅប្រើជាស្វ័យប្រវត្តិនៅក្នុងកាលៈទេសៈដ៏ជាក់លាក់ណាមួយ។

ថ្នាក់ bool

គ្រប់តក្កវត្ថុទាំងអស់គឺជាសិស្សនៃថ្នាក់មានស្រាប់ឈ្មោះ bool ។ ដូចនេះយើងអាចបង្កើតតក្កវត្ថុឬកែវត្ថុផ្សេងៗឲ្យទៅជាតក្កវត្ថុដោយប្រើថ្នាក់មានស្រាប់ឈ្មោះ bool នេះ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
print(bool(0))
print(bool(1.5))
print(bool("កម្មវិធីជាភាសា Python"))
print(bool([ ]))
```

print(bool(0)) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកថ្នាក់មានស្រាប់ឈ្មោះ bool មកប្រើដើម្បីកែលេខ 0 ឲ្យទៅជាតក្កវត្ថុ។

print(bool(1.5)) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកថ្នាក់មានស្រាប់ឈ្មោះ bool មកប្រើដើម្បីកែលេខ 1.5 ឲ្យទៅជាតក្កវត្ថុ។

print(bool("កម្មវិធីជាភាសា Python")) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកថ្នាក់មានស្រាប់ឈ្មោះ bool មកប្រើដើម្បីកែកម្រងអក្សរ "*កម្មវិធីជាភាសា* Python" ឲ្យទៅជាតក្កវត្ថុ ។

print(bool([ ])) គឺជាបញ្ហាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកថ្នាក់មានស្រាប់ឈ្មោះ bool មកប្រើដើម្បីកែកម្រងអថេរទទេមួយឲ្យទៅជាតក្កវត្ថុ។

ដូចនេះយើងឃើញថា គ្រប់វត្ថុទាំងឡាយណាដែលជាលេខសូន្យនិងឬសមាសវត្ថុទទេ សមមូលនឹងតក្កវត្ថុ False ហើយគ្រប់វត្ថុទាំងឡាយណាដែលជាលេខខុសពីសូន្យនិងឬសមាសវត្ថុដែលមានធាតុនៅក្នុងនោះ សមមូលនឹងតក្កវត្ថុ True ។

ម៉្យាងទៀត បើយើងយកថ្នាក់មានស្រាប់ឈ្មោះ bool មកពិនិត្យមើល យើងនឹងឃើញថា វិធីទាំងអស់នៅក្នុងថ្នាក់នោះសុទ្ធតែមានសញ្ញា _ នេះពីរនៅពីមុខនឹងនៅពីក្រោយ។ ដូចនេះវិធីទាំងអស់នោះគឺជាវិធីពិសេសដែលនឹងត្រូវយកទៅប្រើជាស្វ័យប្រវត្តិនៅក្នុងកាលៈទេសៈណាមួយជាក់លាក់។

ថ្នាក់ str

str គឺជាថ្នាក់នៃគ្រប់វត្ថុទាំងឡាយណាដែលជាកម្រងអក្សរ។ យើងបានដឹងរួចមកហើយថា ដើម្បីបង្កើតកម្រងអក្សរ យើងត្រូវប្រើសញ្ញា " " ឬ ' ' នៅអមសងខាងឃ្លាណាមួយ។ ក៏ប៉ុន្តែលើសពីនេះទៀត យើងក៏អាចបង្កើតកម្រងអក្សរដោយយកថ្នាក់មានស្រាប់ឈ្មោះ str នេះមកប្រើបានដែរ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
ឃ្លា = str("តក់ៗពេញបំពង់")
print(ឃ្លា)
```

ឃ្លា = str("តក់ៗពេញបំពង់") គឺជាការបង្កើតកម្រងអក្សរឈ្មោះ ឃ្លា មួយដោយយកថ្នាក់មានស្រាប់ឈ្មោះ str មកប្រើ។

នៅក្នុងការបង្កើតកម្រងអក្សរដោយយកថ្នាក់មានស្រាប់ឈ្មោះ str មកប្រើ យើងចាំបាច់ត្រូវតែផ្តល់វត្ថុណាមួយជាដំណឹងសម្រាប់ស្ថាបនិកនៅក្នុងថ្នាក់នោះ។ ហើយលទ្ធផលបានមកពីការយកថ្នាក់ឈ្មោះ str មកប្រើគឺជាកម្រងអក្សរ។ អាស្រ័យហេតុនេះ យើងអាចប្រើប្រាស់ថ្នាក់មានស្រាប់ឈ្មោះ str នេះដើម្បីកែវត្ថុមួយចំនួនឲ្យទៅជាកម្រងអក្សរ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
ថ្លៃលក់ = str(1000)
ថ្លៃទិញ = str(800.33)
print(ថ្លៃលក់)
print(ថ្លៃទិញ)
```

ថ្លៃលក់ = str(1000) គឺជាការយកថ្នាក់មានស្រាប់ឈ្មោះ str មកប្រើដើម្បីកែចំនួនគត់ដែលជាលេខ 1000 ឲ្យទៅជាកម្រងអក្សរ "1000" ។

ថ្លៃទិញ = str(800.33) គឺជាការយកថ្នាក់មានស្រាប់ឈ្មោះ str មកប្រើដើម្បីកែចំនួនពិតដែលជាលេខ 800.33 ឲ្យទៅជាកម្រងអក្សរ "800.33" ។

បើយើងយកថ្នាក់មានស្រាប់ឈ្មោះ str មកពិនិត្យមើល យើងនឹងឃើញថាក្រៅពីវិធីមានសញ្ញា _ ពីរនៅពីមុខនិងនៅពីក្រោយ ដែលនឹងត្រូវយកទៅប្រើជាស្វ័យប្រវត្តិ នៅមានវិធីមួយចំនួនទៀតដែលគ្មានសញ្ញា _ នេះ។ យើងអាចយកវិធីដែលគ្មានសញ្ញា _ ទៅប្រើការមួយចំនួនដែលទាក់ទងនឹងកម្រងអក្សរ។ វិធីសំខាន់ៗទាំងនោះមាន៖

capitalize(…)

គឺជាវិធីប្រើសម្រាប់ចម្លងយកកម្រងអក្សរឡាតាំងណាមួយមកកែឲ្យទៅជាកម្រងអក្សរដែលមានអក្សរនៅខាងដើមគេជាអក្សរធំ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
ឃ្លាដើម = str("python")
ឃ្លាថ្មី = ឃ្លាដើម.capitalize()
print(ឃ្លាថ្មី)
```

ឃ្លាថ្មី = ឃ្លាដើម.capitalize() គឺជាការយកវិធីឈ្មោះ capitalize នៅក្នុងថ្នាក់មានស្រាប់ឈ្មោះ str មកប្រើដើម្បីចម្លងយកកម្រងអក្សរឈ្មោះ ឃ្លាដើម មកកែឲ្យទៅជាកម្រងអក្សរថ្មីដែលមានអក្សរនៅខាងដើមគេជាអក្សរធំ។

center(…)

គឺជាវិធីប្រើសម្រាប់ចម្លងយកកម្រងអក្សរណាមួយមកបង្កើតជាកម្រងអក្សរថ្មីដែលមានអក្សរផ្សេងទៀតនៅអមសងខាង។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
ឃ្លាដើម = str("python")
ឃ្លាថ្មី = ឃ្លាដើម.center(60, "~")
print(ឃ្លាថ្មី)
```

ឃ្លាថ្មី = ឃ្លាដើម.center(60, "~") គឺជាការយកវិធីឈ្មោះ center នៅក្នុងថ្នាក់មានស្រាប់ឈ្មោះ str មកប្រើដើម្បីចម្លងយកកម្រងអក្សរឈ្មោះ ឃ្លាដើម មកបង្កើតជាកម្រងអក្សរឈ្មោះ ឃ្លាថ្មីមួយថ្មីទៀតដែលមានអក្សរ "~" នៅអមសងខាង។

count(…)

គឺជាវិធីប្រើសម្រាប់រាប់ចំនួនពាក្យឬកម្រងអក្សរណាមួយដែលមាននៅក្នុងកម្រងណាមួយទៀត។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
ឃ្លាដើម = str("លិខិតចាត់តាំង")
ចំនួនអក្សរ = ឃ្លាដើម.count("ត")
```

```
print(ចំនួនអក្សរ)
```

ចំនួនអក្សរ = ឃ្លាដើម.count("ត") គឺជាការយកវិធីឈ្មោះ count នៅក្នុងថ្នាក់មានស្រាប់ឈ្មោះ str មកប្រើដើម្បីរាប់ចំនួនអក្សរ "ត" នៅក្នុងកម្រងអក្សរឈ្មោះ ឃ្លាដើម ។

format(…)

គឺជាវិធីប្រើសម្រាប់បញ្ចូលវត្ថុមួយចំនួនទៅក្នុងកម្រងអក្សរណាមួយដើម្បីបង្កើតកម្រងអក្សរមួយថ្មីទៀត។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
ឃ្លាដើម = str("ចំនួនប្រាក់{0}គឺ {ថ្លៃលក់}")
ឃ្លាថ្មី = ឃ្លាដើម.format("ថ្លៃលក់", ថ្លៃលក់=1000)
print(ឃ្លាថ្មី)
```

ឃ្លាថ្មី = ឃ្លាដើម.format("ថ្លៃលក់", ថ្លៃលក់=1000) គឺជាការយកវិធីឈ្មោះ format មកប្រើដើម្បីបញ្ចូលពាក្យថា "ថ្លៃលក់" និងវត្ថុឈ្មោះ ថ្លៃលក់ ចូលទៅក្នុងកម្រងអក្សរឈ្មោះ ឃ្លាដើម នៅត្រង់កន្លែងដែលមានសញ្ញា {0} និង {ថ្លៃលក់} រៀងគ្នា។

index(…)

គឺជាវិធីប្រើសម្រាប់រកលេខរៀងនៃអក្សរទីមួយនៅក្នុងពាក្យណាមួយនៅក្នុងកម្រងអក្សរណាមួយ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
ឃ្លាដើម = str("លិខិតចាត់តាំង")
លេខរៀង = ឃ្លាដើម.index("ចាត់តាំង")
print(លេខរៀង)
```

លេខរៀង = *ឃ្លាដើម*.index("*ចាត់តាំង*") គឺជាការយកវិធីឈ្មោះ index មកប្រើដើម្បីរកលេខរៀងនៃអក្សរទីមួយនៅក្នុងពាក្យ "ចាត់តាំង" ដែលជាពាក្យនៅក្នុងកម្រងអក្សរឈ្មោះ ឃ្លាដើម ។

join(…)

គឺជាវិធីប្រើសម្រាប់បញ្ចូលពាក្យណាមួយនៅចន្លោះកម្រងអក្សរពីរដើម្បីបង្កើតកម្រងអក្សរថ្មីមួយទៀត។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
ពាក្យ = str("ជា")
ឃ្លាថ្មី = ពាក្យ.join(["កម្មវិធី", "ភាសា Python"])
print(ឃ្លាថ្មី)
```

ឃ្លាថ្មី = *ពាក្យ*.join(["*កម្មវិធី*", "*ភាសា* Python"]) គឺជាការយកវិធីឈ្មោះ join មកប្រើដើម្បីបញ្ចូលពាក្យថា "ជា" នៅចន្លោះកណ្តាលកម្រងអក្សរ "កម្មវិធី" និង "ភាសា Python" ដើម្បីបង្កើតកម្រងអក្សរមួយថ្មីទៀត។ កម្រងអក្សរទាំងពីរនោះគឺជាធាតុនៅក្នុងកម្រងអថេរមួយ។

replace(…)

គឺជាវិធីប្រើសម្រាប់យកពាក្យណាមួយទៅជំនួសពាក្យនៅក្នុងកម្រងអក្សរណាមួយផ្សេងទៀត ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
ឃ្លា = str("តក់ៗពេញបំពង់")
ឃ្លាថ្មី = ឃ្លា.replace("បំពង់", "ពាង")
print(ឃ្លាថ្មី)
```

ឃ្លាថ្មី = *ឃ្លា*.replace("*បំពង់*", "*ពាង*") គឺជាការយកវិធីឈ្មោះ replace មកប្រើដើម្បីជំនួសពាក្យថា "បំពង់" នៅក្នុងកម្រងអក្សរឈ្មោះ ឃ្លា ដោយពាក្យថា "ពាង" វិញ។

upper(…)

គឺជាវិធីប្រើសម្រាប់ចម្លងយកកម្រងអក្សរឡាតាំងណាមួយមកកែឲ្យទៅកម្រងអក្សរថ្មីមួយទៀតដែលមានសុទ្ធតែអក្សរធំនៅក្នុងនោះ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
ឃ្លា =str("programming in python")
ឃ្លាថ្មី = ឃ្លា.upper()
print(ឃ្លាថ្មី)
```

ឃ្លាថ្មី = ឃ្លា.upper() គឺជាការយកវិធីឈ្មោះ upper មកប្រើដើម្បីចម្លងយកកម្រងអក្សរឈ្មោះ ឃ្លា មកកែទៅជាកម្រងអក្សរថ្មីមួយទៀតដែលនៅក្នុងនោះមានសុទ្ធតែអក្សរធំទាំងអស់។

ថ្នាក់ tuple

គ្រប់វត្ថុដែលជាកម្រងថេរទាំងអស់គឺជាសិស្សនៃថ្នាក់មានស្រាប់ឈ្មោះ tuple ។ ដូចនេះយើងអាចយកថ្នាក់ឈ្មោះ tuple នេះមកប្រើដើម្បីបង្កើតកម្រងថេរឬកែសមាសវត្ថុផ្សេងៗឲ្យទៅជាកម្រងថេរ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
កម្រងទី១ = tuple("Python")
កម្រងទី២ = tuple([100, 1.5, True])
កម្រងទី៣ = tuple({'A':100, 'B':1.5, 'C':True})
កម្រងទី៤ = tuple({100, False, 200})
print(កម្រងទី១)
print(កម្រងទី២)
print(កម្រងទី៣)
print(កម្រងទី៤)
```

កម្រងទី១ = tuple("Python") គឺជាការយកថ្នាក់មានស្រាប់ឈ្មោះ tuple មកប្រើដើម្បីកែកម្រងអក្សរ "Python" ឲ្យទៅជាកម្រងថេរឈ្មោះ កម្រងទី១ ។

កម្រងទី២ = tuple([100, 1.5, True]) គឺជាការយកថ្នាក់មានស្រាប់ឈ្មោះ tuple មកប្រើដើម្បីកែកម្រងអថេរ [100, 1.5, True] ឲ្យទៅជាកម្រងថេរឈ្មោះ កម្រងទី២ ។

កម្រងទី៣ = tuple({'A':100, 'B':1.5, 'C':True}) គឺជាការយកថ្នាក់មានស្រាប់ឈ្មោះ tuple មកប្រើដើម្បីកែវចនានុក្រម {'A':100, 'B':1.5, 'C':True} ឲ្យទៅជាកម្រងថេរឈ្មោះ កម្រងទី៣ ។

កម្រងទី៤ = tuple({100, False, 200}) គឺជាការយកថ្នាក់មានស្រាប់ឈ្មោះ tuple មកប្រើដើម្បីកែសំណុំ {100, False, 200} ឲ្យទៅជាកម្រងថេរឈ្មោះ កម្រងទី៤ ។

នៅក្នុងថ្នាក់មានស្រាប់ឈ្មោះ tuple មានវិធីមួយចំនួនដែលយើងអាចយកទៅប្រើជាមួយនឹងកម្រងថេរផ្សេងៗ។ វិធីទាំងនោះមានដូចតទៅនេះ៖

count(…)

គឺជាវិធីប្រើសម្រាប់រាប់ចំនួនធាតុដូចគ្នានៅក្នុងកម្រងថេណាមួយ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
កម្រងចម្រុះ = (100, 1.5, "ប្រាក់ចំណេញ", 100, 'ថ្ងៃលក់')
ចំនួនធាតុ = កម្រងចម្រុះ.count(100)
print(ចំនួនធាតុ)
```

ចំនួនធាតុ = *កម្រងចម្រុះ*.count(100) គឺជាការយកវិធីឈ្មោះ count មកប្រើដើម្បីរាប់ចំនួនធាតុដែលជាលេខ 100 នៅក្នុងកម្រងថេរឈ្មោះ កម្រងចម្រុះ ។

index(..)

គឺជាវិធីប្រើសម្រាប់រកលេខរៀងនៃធាតុណាមួយនៅក្នុងកម្រងថេរណាមួយ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
កម្រងចម្រុះ = (100, 1.5, "ប្រាក់ចំណេញ", 100, 'ថ្លៃលក់')
លេខរៀង = កម្រងចម្រុះ.index('ថ្លៃលក់')
print(លេខរៀង)
```

លេខរៀង = កម្រងចម្រុះ.index('ថ្លៃលក់') គឺជាការយកវិធីឈ្មោះ index មកប្រើដើម្បីពិនិត្យមើលលេខរៀងនៃធាតុដែលជាពាក្យ "ថ្លៃលក់" នៅក្នុងកម្រងថេរឈ្មោះ កម្រងចម្រុះ ។

ថ្នាក់ list

គ្រប់កម្រងអថេរទាំងអស់គឺជាសិស្សនៃថ្នាក់មានស្រាប់ឈ្មោះ list ។ ដូចនេះយើងអាចថ្នាក់ឈ្មោះ list នេះមកប្រើដើម្បីបង្កើតកម្រងអថេរឬកែសមាសវត្ថុមួយចំនួនឲ្យទៅជាកម្រងអថេរ ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
កម្រងទី១ = list("Python")
កម្រងទី២ = list((100, 1.5, True))
កម្រងទី៣ = list({'A':100, 'B':1.5, 'C':True})
កម្រងទី៤ = list({100, False, 200})
print(កម្រងទី១)
print(កម្រងទី២)
print(កម្រងទី៣)
print(កម្រងទី៤)
```

កម្រង៍ទី១ = list("Python") គឺជាការយកថ្នាក់មានស្រាប់ឈ្មោះ list មកប្រើដើម្បីកែកម្រងអក្សរ "Python" ឲ្យទៅជាកម្រងអថេរឈ្មោះ កម្រងទី១ ។

កម្រង៍ទី២ = list((100, 1.5, True)) គឺជាការយកថ្នាក់មានស្រាប់ឈ្មោះ list មកប្រើដើម្បីកែកម្រងថេរ (100, 1.5, True) ឲ្យទៅជាកម្រងអថេរឈ្មោះ កម្រងទី២ ។

កម្រង៍ទី៣ = list({'A':100, 'B':1.5, 'C':True}) គឺជាការយកថ្នាក់មានស្រាប់ឈ្មោះ list មកប្រើដើម្បីកែវចនានុក្រម {'A':100, 'B':1.5, 'C':True} ឲ្យទៅជាកម្រងអថេរឈ្មោះ កម្រងទី៣ ។

កម្រង៍ទី៤ = list({100, False, 200}) គឺជាការយកថ្នាក់មានស្រាប់ឈ្មោះ list មកប្រើដើម្បីកែសំណុំ {100, False, 200} ឲ្យទៅជាកម្រងអថេរឈ្មោះ កម្រងទី៤ ។

នៅក្នុងថ្នាក់មានស្រាប់ឈ្មោះ list មានវិធីមួយចំនួនដែលយើងអាចយកទៅប្រើជាមួយនឹងកម្រងអថេរផ្សេងៗ។ វិធីទាំងនោះមានដូចតទៅនេះ៖

append(…)

គឺជាវិធីប្រើសម្រាប់បន្ថែមធាតុណាមួយនៅខាងចុងកម្រងអថេរណាមួយ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
កម្រង៍ចម្រុះ = list([100, True, "ប្រាក់ចំណេញ", 3.5])
កម្រង៍ចម្រុះ.append("ថ្លៃលក់")
print(កម្រង៍ចម្រុះ)
```

កម្រង៍ចម្រុះ.append("*ថ្លៃលក់*") គឺជាការយកវិធីឈ្មោះ append មកប្រើដើម្បីបន្ថែមវត្ថុដែលជាកម្រងអក្សរ "ថ្លៃលក់" ធ្វើជាធាតុនៅខាងចុងកម្រងអថេរឈ្មោះ កម្រងចម្រុះ ។

count(…)

គឺជាវិធីប្រើសម្រាប់រាប់ចំនួនធាតុដូចគ្នានៅក្នុងកម្រងអថេរណាមួយ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
កម្រងចម្រុះ = list([100, True, "ប្រាក់ចំណេញ", 3.5, 100])
ចំនួនធាតុ = កម្រងចម្រុះ.count(100)
print(ចំនួនធាតុ)
```

ចំនួនធាតុ = កម្រងចម្រុះ.count(100) គឺជាការយកវិធីឈ្មោះ count មកប្រើដើម្បីរាប់ចំនួនធាតុដែលជាលេខ 100 នៅក្នុងកម្រងអថេរឈ្មោះ កម្រងចម្រុះ ។

extend(…)

គឺជាវិធីប្រើសម្រាប់បន្ថែមធាតុនៃសមាសវត្ថុណាមួយចូលទៅក្នុងកម្រងអថេរណាមួយនៅខាងចុង។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
កម្រងចម្រុះ = list([100, True, "ប្រាក់ចំណេញ", 3.5, 100])
កម្រងចម្រុះ.extend(["ថ្លៃលក់", "ថ្លៃទិញ", False])
print(កម្រងចម្រុះ)
```

កម្រងចម្រុះ.extend(["ថ្លៃលក់", "ថ្លៃទិញ", False]) គឺជាការយកវិធីឈ្មោះ extend មកប្រើដើម្បីបន្ថែមធាតុមួយចំនួននៅក្នុងកម្រងអថេរមួយចូលទៅក្នុងកម្រងអថេរឈ្មោះ កម្រងចម្រុះ នៅខាងចុង។

index(…)

គឺជាវិធីប្រើសម្រាប់រកលេខរៀងនៃធាតុណាមួយនៅក្នុងកម្រងអថេរណាមួយ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
កម្រងចម្រុះ = list([100, True, "ប្រាក់ចំណេញ", 3.5, 100])
```

```
លេខរៀង = កម្រងចម្រុះ.index("ប្រាក់ចំណេញ")
print(លេខរៀង)
```

លេខរៀង = កម្រងចម្រុះ.index("ប្រាក់ចំណេញ") គឺជាការយកវិធីឈ្មោះ index មកប្រើដើម្បីរកលេខរៀងនៃធាតុដែលជាកម្រងអក្សរ "ប្រាក់ចំណេញ" នៅក្នុងកម្រងអថេរឈ្មោះ កម្រងចម្រុះ ។

insert(…)

គឺជាវិធីប្រើសម្រាប់បញ្ចូលវត្ថុណាមួយចូលទៅក្នុងកម្រងអថេរណាមួយនៅខាងមុខធាតុមានលេខរៀងណាមួយ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
កម្រងចម្រុះ = list([100, True, "ប្រាក់ចំណេញ", 3.5, 100])
កម្រងចម្រុះ.insert(2, "ថ្ងៃលក់")
print(កម្រងចម្រុះ)
```

កម្រងចម្រុះ.insert(2, "ថ្ងៃលក់") គឺជាការយកវិធីឈ្មោះ insert មកប្រើដើម្បីបញ្ចូលកម្រងអក្សរ "ថ្លៃលក់" នៅខាងមុខធាតុមានលេខរៀង 2 នៅក្នុងកម្រងអថេរឈ្មោះ កម្រងចម្រុះ ។

pop(…)

គឺជាវិធីប្រើសម្រាប់កាត់យកធាតុមានលេខរៀងអ្វីមួយនៅក្នុងកម្រងអថេរណាមួយមកប្រើការ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
កម្រងចម្រុះ = list([100, True, "ប្រាក់ចំណេញ", 3.5, 100])
ធាតុ = កម្រងចម្រុះ.pop(2)
print(ធាតុ)
print(កម្រងចម្រុះ)
```

ធាតុ = កម្រងចម្រុះ.pop(2) គឺជាការយកវិធីឈ្មោះ pop មកប្រើដើម្បីកាត់យកធាតុមានលេខរៀង 2 នៅក្នុងកម្រងអថេរឈ្មោះ កម្រងចម្រុះ មកប្រើការ។

remove(…)

គឺជាវិធីប្រើសម្រាប់លុបធាតុណាមួយចេញពីកម្រងអថេរណាមួយ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
កម្រងចម្រុះ = list([100, True, "ប្រាក់ចំណេញ", 3.5, 100])
កម្រងចម្រុះ.remove("ប្រាក់ចំណេញ")
print(កម្រងចម្រុះ)
```

កម្រងចម្រុះ.remove("*ប្រាក់ចំណេញ*") គឺជាការយកវិធីឈ្មោះ remove មកប្រើដើម្បីលុបធាតុដែលជាកម្រងអក្សរ "ប្រាក់ចំណេញ" ចេញពីកម្រងអថេរឈ្មោះ កម្រងចម្រុះ ។

reverse(…)

គឺជាវិធីប្រើសម្រាប់តម្រៀបធាតុនៅក្នុងកម្រងអថេរណាមួយឲ្យមានលំដាប់ថ្នាក់បញ្ច្រាសមកវិញ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
កម្រងចម្រុះ = list([100, True, "ប្រាក់ចំណេញ", 3.5])
កម្រងចម្រុះ.reverse()
print(កម្រងចម្រុះ)
```

កម្រងចម្រុះ.reverse() គឺជាការយកវិធីឈ្មោះ reverse មកប្រើដើម្បីតម្រៀបធាតុនៅក្នុងកម្រងអថេរឈ្មោះ កម្រងចម្រុះ ឲ្យមានលំដាប់ថ្នាក់បញ្ច្រាសមកវិញ។

sort(…)

គឺជាវិធីប្រើសម្រាប់តម្រៀបធាតុនៅក្នុងកម្រងអថេរណាមួយឲ្យមានលំដាប់ថ្នាក់ពីតូចទៅធំ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
កម្រងចម្រុះ = list([100, 76, 23, 3.5])
កម្រងចម្រុះ.sort()
print(កម្រងចម្រុះ)
```

កម្រងចម្រុះ.sort() គឺជាការយកវិធីឈ្មោះ sort មកប្រើដើម្បីតម្រៀបធាតុនៅក្នុងកម្រងអថេរឈ្មោះ កម្រងចម្រុះ ឲ្យមានលំដាប់ថ្នាក់ពីតូចទៅធំ។

ថ្នាក់ dict

គ្រប់វត្ថុដែលជាវចនានុក្រមទាំងអស់គឺជាសិស្សនៃថ្នាក់មានស្រាប់ឈ្មោះ dict ។ ដូចនេះយើងអាចយកថ្នាក់ឈ្មោះ dict នេះមកប្រើដើម្បីបង្កើតវចនានុក្រមឬកែវត្ថុផ្សេងៗឲ្យទៅជាវចនានុក្រម។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
វចនានុក្រមទី១ = dict({"ថ្លៃលក់":1000, "ថ្លៃទិញ":900, "ប្រាក់ចំណេញ":100})
វចនានុក្រមទី២ = dict([("ថ្លៃលក់", 1000), ("ថ្លៃទិញ", 900), ("ប្រាក់ចំណេញ", 100)])
វចនានុក្រមទី៣ = dict(ថ្លៃលក់=1000, ថ្លៃទិញ=900, ប្រាក់ចំណេញ=100)
print(វចនានុក្រមទី១)
print(វចនានុក្រមទី២)
print(វចនានុក្រមទី៣)
```

វចនានុក្រមទី១ = dict({"ថ្លៃលក់":1000, "ថ្លៃទិញ":900, "ប្រាក់ចំណេញ":100}) គឺជាការយកថ្នាក់មានស្រាប់ឈ្មោះ dict មកប្រើដើម្បីបង្កើតវចនានុក្រម ឈ្មោះ វចនានុក្រមទី១ មួយ។

វចនានុក្រមទី២ = dict([("*ថ្ងៃលក់*", 1000), ("*ថ្ងៃទិញ*", 900), ("*ប្រាក់ចំណេញ*", 100)]) គឺជាការយកថ្នាក់មានស្រាប់ឈ្មោះ dict មកប្រើដើម្បីកែកម្រងអថេរមួយឲ្យទៅជាវចនានុក្រមឈ្មោះ វចនានុក្រមទី២ មួយទៀត។ ធាតុនៃកម្រងអថេរដែលត្រូវយកមកកែ គឺជាកម្រងថេរមានធាតុពីរ។

វចនានុក្រមទី៣ = dict(*ថ្ងៃលក់*=1000, *ថ្ងៃទិញ*=900, *ប្រាក់ចំណេញ*=100) គឺជាការយកថ្នាក់មានស្រាប់ឈ្មោះ dict មកប្រើដោយផ្តល់ដំណឹងតាមដំណាងមួយចំនួនឲ្យទៅស្ថាបនិកនៃថ្នាក់ឈ្មោះ dict នេះដើម្បីបង្កើតវចនានុក្រមមួយមានឈ្មោះថា វចនានុក្រមទី៣ មួយ។

នៅក្នុងថ្នាក់មានស្រាប់ឈ្មោះ dict មានវិធីមួយចំនួនដែលយើងអាចយកមកប្រើជាមួយនឹងវត្ថុដែលជាសិស្សនៃថ្នាក់ឈ្មោះ dict នេះ។ វិធីទាំងនោះមានដូចតទៅនេះ៖

clear(…)

គឺជាវិធីប្រើសម្រាប់លុបធាតុទាំងអស់ដែលមាននៅក្នុងវចនានុក្រមណាមួយ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
បង្ហិចប្រាក់ = dict({"ថ្ងៃលក់":1000, "ថ្ងៃទិញ":900, "ប្រាក់ចំណេញ":100})
បង្ហិចប្រាក់.clear()
print(បង្ហិចប្រាក់)
```

បង្ហិចប្រាក់.clear() គឺជាការយកវិធីឈ្មោះ clear មកប្រើដើម្បីលុបធាតុទាំងអស់ដែលមាននៅក្នុងវចនានុក្រមឈ្មោះ បង្ហិចប្រាក់ ។

copy(…)

គឺជាវិធីប្រើសម្រាប់ចម្លងយកវចនានុក្រមណាមួយមកបង្កើតជាវចនានុក្រមដូចគ្នាថ្មីមួយទៀត ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
បង្វិចប្រាក់ = dict({"ថ្លៃលក់":1000, "ថ្លៃទិញ":900, "ប្រាក់ចំណេញ":100})
បង្វិចថ្មី = បង្វិចប្រាក់.copy()
print(បង្វិចប្រាក់)
print(បង្វិចថ្មី)
```

បង្វិចថ្មី = បង្វិចប្រាក់.copy() គឺជាការយកវិធីឈ្មោះ copy មកប្រើដើម្បីចម្លងយកវចនានុក្រមឈ្មោះ បង្វិចប្រាក់ មកបង្កើតជាវចនានុក្រមដូចគ្នាថ្មីមួយទៀតមានឈ្មោះថា បង្វិចថ្មី ។

get(…)

គឺជាវិធីប្រើសម្រាប់ចម្លងយកតម្លៃជាប់នឹងកូនសោរណាមួយនៅក្នុងវចនានុក្រមណាមួយមកប្រើការ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
បង្វិចប្រាក់ = dict({"ថ្លៃលក់":1000, "ថ្លៃទិញ":900, "ប្រាក់ចំណេញ":100})
តម្លៃ = បង្វិចប្រាក់.get("ប្រាក់ចំណេញ")
print(តម្លៃ)
```

តម្លៃ = បង្វិចប្រាក់.get("ប្រាក់ចំណេញ") គឺជាការយកវិធីឈ្មោះ get មកប្រើដើម្បីចម្លងយកតម្លៃជាប់នឹងកូនសោរ "ប្រាក់ចំណេញ" នៅក្នុងវចនានុក្រមឈ្មោះ បង្វិចប្រាក់ មកប្រើការ។

pop(…)

គឺជាវិធីប្រើសម្រាប់កាត់យកតម្លៃជាប់នឹងកូនសោរណាមួយនៅក្នុងវចនានុក្រមណាមួយមកប្រើការ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
បង្វិចប្រាក់ = dict({"ថ្លៃលក់":1000, "ថ្លៃទិញ":900, "ប្រាក់ចំណេញ":100})
តម្លៃ = បង្វិចប្រាក់.pop("ប្រាក់ចំណេញ")
print(តម្លៃ)
```

តម្លៃ = បង្វិចប្រាក់.pop("*ប្រាក់ចំណេញ*") គឺជាការយកវិធីឈ្មោះ pop មកប្រើដើម្បីកាត់យកតម្លៃជាប់នឹងកូនសោរ "ប្រាក់ចំណេញ" នៅក្នុងវចនានុក្រមឈ្មោះ បង្វិចប្រាក់ មកប្រើការ។

popitem(…)

គឺជាវិធីប្រើសម្រាប់កាត់យកធាតុគូណាមួយនៅក្នុងវចនានុក្រមណាមួយមកបង្កើតជាកម្រងថេរមានធាតុពីរ គឺធាតុមួយជាកូនសោរនិងធាតុមួយទៀតជាតម្លៃ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
បង្វិចប្រាក់ = dict({"ថ្លៃលក់":1000, "ថ្លៃទិញ":900, "ប្រាក់ចំណេញ":100})
លទ្ធផល = បង្វិចប្រាក់.popitem()
print(លទ្ធផល)
```

លទ្ធផល = បង្វិចប្រាក់.popitem() គឺជាការយកវិធីឈ្មោះ popitem មកប្រើដើម្បីកាត់យកធាតុគូណាមួយមិនកំណត់នៅក្នុងវចនានុក្រមឈ្មោះ បង្វិចប្រាក់ មកប្រើដើម្បីបង្កើតជាកម្រងថេរមួយមានឈ្មោះថា លទ្ធផល ។

update(…)

គឺជាវិធីប្រើសម្រាប់ធ្វើឲ្យវចនានុក្រមមួយទាន់សម័យ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
បង្វិចប្រាក់ = dict({"ថ្លៃលក់":1000, "ថ្លៃទិញ":900, "ប្រាក់ចំណេញ":100})
បង្វិចប្រាក់.update({"ថ្លៃលក់":2000, "ថ្លៃទិញ":800, "ប្រាក់ចំណេញ":1200, "សោហ៊ុយ":50})
print(បង្វិចប្រាក់)
```

បង្វិចប្រាក់.update({"*ថ្លៃលក់*":2000, "*ថ្លៃទិញ*":800, "*ប្រាក់ចំណេញ*":1200, "*សោហ៊ុយ*":50}) គឺជាការយកវិធីឈ្មោះ update មកប្រើដើម្បីធ្វើឲ្យវចនានុក្រមឈ្មោះ បង្វិចប្រាក់ ទាន់សម័យ។

ថ្នាក់ set

ការបង្កើតវត្ថុដែលជាសំណុំកន្លងមក គឺជាការបង្កើតសិស្សនៃថ្នាក់មានស្រាប់ឈ្មោះ set ដោយមិនបាច់យកថ្នាក់ឈ្មោះ set នេះមកប្រើ។ ក៏ប៉ុន្តែយើងក៏អាចយកថ្នាក់ឈ្មោះ set នេះមកប្រើដោយផ្ទាល់ដើម្បីបង្កើតសំណុំឬកែវត្ថុផ្សេងៗទៀតឲ្យទៅជាសំណុំបានដែរ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
សំណុំទី១ = set({1000, 900, 100})
សំណុំទី២ = set({"ថ្លៃលក់":1000, "ថ្លៃទិញ":900, "ប្រាក់ចំណេញ":100})
សំណុំទី៣ = set([1000, 900, 100])
សំណុំទី៤ = set((1000, 900, 100))
print(សំណុំទី១)
print(សំណុំទី២)
print(សំណុំទី៣)
print(សំណុំទី៤)
```

សំណុំទី១ = set({1000, 900, 100}) គឺជាការបង្កើតសំណុំមួយមានឈ្មោះថា សំណុំទី១ ដោយយកថ្នាក់មានស្រាប់ឈ្មោះ set មកប្រើ។

សំណុំទី២ = set({"ថ្លៃលក់":1000, "ថ្លៃទិញ":900, "ប្រាក់ចំណេញ":100}) គឺជាការបង្កើតសំណុំមួយមានឈ្មោះថា សំណុំទី២ ដោយយកថ្នាក់មានស្រាប់ឈ្មោះ set មកប្រើនិងផ្តល់វចនានុក្រមមួយជាដំណឹងសម្រាប់ស្ថាបនិកនៅក្នុងថ្នាក់នោះ។

សំណុំទី៣ = set([1000, 900, 100]) គឺជាការបង្កើតសំណុំមួយមានឈ្មោះថា សំណុំទី៣ ដោយយកថ្នាក់មានស្រាប់ឈ្មោះ set មកប្រើនិងផ្តល់កម្រងអថេរមួយជាដំណឹងសម្រាប់ស្ថាបនិកនៅក្នុងថ្នាក់នោះ។

សំណុំទី៤ = set((1000, 900, 100)) គឺជាការបង្កើតសំណុំមួយមានឈ្មោះថា សំណុំទី៤ ដោយយកថ្នាក់មានស្រាប់ឈ្មោះ set មកប្រើនិងផ្តល់កម្រងថេរមួយជាដំណឹងសម្រាប់ស្ថាបនិកនៅក្នុងថ្នាក់នោះ។

នៅក្នុងថ្នាក់មានស្រាប់ឈ្មោះ set មានវិធីមួយចំនួនដែលយើងអាចយកមកប្រើជាមួយនឹងវត្ថុដែលជាសំណុំផ្សេងៗ។ វិធីទាំងនោះមានដូចតទៅនេះ៖

add(…)

គឺជាវិធីប្រើសម្រាប់បន្ថែមធាតុថ្មីមួយទៀតចូលទៅក្នុងសំណុំណាមួយ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
សំណុំដើម = set({1000, 900, 100})
សំណុំដើម.add("ប្រាក់ចំណេញ")
print(សំណុំដើម)
```

សំណុំដើម.add("*ប្រាក់ចំណេញ*") គឺជាការយកវិធីឈ្មោះ add មកប្រើដើម្បីថែមធាតុដែលជាពាក្យថា "ប្រាក់ចំណេញ" មួយទៀតចូលទៅក្នុងសំណុំឈ្មោះ សំណុំដើម ។

clear(…)

គឺជាវិធីប្រើសម្រាប់លុបធាតុទាំងអស់នៅក្នុងសំណុំណាមួយ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
សំណុំដើម = set({1000, 900, 100})
```

```
សំណុំដើម.clear()
print(សំណុំដើម)
```

សំណុំដើម.clear() គឺជាការយកវិធីឈ្មោះ clear មកប្រើដើម្បីលុបធាតុទាំងអស់ដែលមាននៅក្នុងសំណុំឈ្មោះ សំណុំដើម ។

copy(…)

គឺជាវិធីប្រើសម្រាប់ចម្លងយកសំណុំណាមួយមកបង្កើតជាសំណុំដូចគ្នាមួយថ្មីទៀត។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
សំណុំដើម = set({1000, 900, 100})
សំណុំថ្មី = សំណុំដើម.copy()
print(សំណុំដើម)
print(សំណុំថ្មី)
```

សំណុំថ្មី = សំណុំដើម.copy() គឺជាការយកវិធីឈ្មោះ copy មកប្រើដើម្បីចម្លងយកសំណុំឈ្មោះ សំណុំដើម មកបង្កើតជាសំណុំមួយថ្មីមានឈ្មោះថា សំណុំថ្មី ។

discard(…)

គឺជាវិធីប្រើសម្រាប់លុបធាតុណាមួយចេញពីសំណុំណាមួយ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
សំណុំដើម = set({1000, 900, 100})
សំណុំដើម.discard(100)
print(សំណុំដើម)
```

សំណុំដើម.discard(100) គឺជាការយកវិធីឈ្មោះ discard មកប្រើដើម្បីលុបធាតុដែលជាលេខ 100 ចេញពីក្នុងសំណុំឈ្មោះ សំណុំដើម ។

pop(…)

គឺជាវិធីប្រើសម្រាប់កាត់យកធាតុណាមួយពីក្នុងសំណុំណាមួយមកប្រើការ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
សំណុំដើម = set({1000, 900, 100})
ធាតុ = សំណុំដើម.pop()
print(ធាតុ)
```

ធាតុ = សំណុំដើម.pop() គឺជាការយកវិធីឈ្មោះ pop មកប្រើដើម្បីកាត់យកធាតុណាមួយនៅក្នុងសំណុំឈ្មោះ សំណុំដើម មកប្រើការ។

remove(…)

គឺជាវិធីប្រើសម្រាប់លុបធាតុណាមួយចេញពីសំណុំណាមួយ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
សំណុំដើម = set({1000, 900, 100})
សំណុំដើម.remove(900)
print(សំណុំដើម)
```

សំណុំដើម.remove(900) គឺជាការយកវិធីឈ្មោះ remove មកប្រើដើម្បីលុបធាតុដែលជាលេខ 900 ចេញពីសំណុំឈ្មោះ សំណុំដើម ។

សាស្ត្រា

នៅក្នុងភាសាខ្មែរយើង ពាក្យថាសាស្ត្រា គឺសំដៅទៅលើឯកសារផ្សេងៗដែលនៅក្នុងនោះមានការកត់ត្រាទុកនូវក្បួនខ្នាតមួយចំនួន មានដូចជាសាស្ត្រាស្លឹករឹតជាដើម។ នៅក្នុងភាសា Python ពាក្យថា module គឺសំដៅទៅលើឯកសារផ្សេងៗដែលនៅក្នុងនោះមានការកត់ត្រាទុកនូវ ទិន្នន័យ ក្បួន និងថ្នាក់មួយចំនួនដែលទាក់ទងគ្នាក្នុងការដោះស្រាយបញ្ហាដោយឡែកណាមួយ។ ដូចនេះយើងគួរតែប្រើពាក្យថា *សាស្ត្រា* នេះដើម្បីបកប្រែពាក្យថា module នៅក្នុងភាសា Python ។

ការបង្កើត សាស្ត្រា

នៅក្នុងភាសា Python បើយើងប្រមូលទិន្នន័យ ក្បួន និងថ្នាក់ផ្សេងៗមកកត់ត្រាទុកជាឯកសារមួយ ដោយដាក់ឈ្មោះថាអ្វីមួយដែលមានអក្សរ .py នៅខាងចុង យើងនឹងបាន module មួយដែលយើងអាចបកប្រែជាភាសាខ្មែរថាសាស្ត្រា។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
pi = 3.14
def ផ្ទៃរង្វង់(កាំ) :
   S = កាំ * កាំ * pi
   return S

class ធរណីមាត្រ() :
   def __init__(សិស្ស) :
      print("ស្ថាបនិកនៃថ្នាក់ ធរណីមាត្រ។")

   def ក្រឡាផ្ទៃ() :
```

```
    print("ក្រឡាផ្ទៃ")

def មាឌ() :
    print("មាឌ")
```

បើយើងរក្សាកម្មវិធីខាងលើនេះទុកជាឯកសារមួយដែលមានឈ្មោះជាឧទាហរណ៍ថា khmer.py នៅក្នុងថាសរឹង តាមភាសា Python ឯកសារនេះគឺជាសាស្ត្រាមួយមានឈ្មោះថា khmer ។

ឈ្មោះរបស់សាស្ត្រាក៏ជាឈ្មោះមួយដូចជាឈ្មោះរបស់វត្ថុដទៃទៀតនៅក្នុងភាសា Python ។ ដូចនេះយើងត្រូវគោរពទៅតាមវិធាននៃការបង្កើតឈ្មោះនៅក្នុងភាសា Python ដើម្បីបង្កើតឈ្មោះរបស់សាស្ត្រា។

គោលគំនិតសំខាន់នៃការបង្កើតសាស្ត្រា គឺដើម្បីរក្សាទុកនូវក្បួនខ្នាតមួយចំនួនដើម្បីអាចទៅប្រើនៅក្នុងកម្មវិធីផ្សេងៗទៀតបាន។ ដូចនេះក្បួនខ្នាតដែលត្រូវរក្សាទុកនៅក្នុងសាស្ត្រាត្រូវតែមានលក្ខណៈទូទៅ និងដែលអាចមានប្រយោជន៍ក្នុងការដោះស្រាយបញ្ហាបានច្រើនប្រភេទខុសៗគ្នា។ ដោយឡែក ការបង្កើតសាស្ត្រាឈ្មោះ khmer ខាងលើនេះ គឺក្នុងគោលបំណងរក្សាទុកនូវទិន្នន័យ ក្បួន និង ថ្នាក់មួយចំនួនដែលអាចត្រូវយកទៅប្រើនៅក្នុងកម្មវិធីផ្សេងៗទៀតដែលទាក់ទងទៅនឹងការដោះស្រាយបញ្ហាធរណីមាត្រ។

ការយកសាស្ត្រាមកប្រើ

ដើម្បីអាចយកសាស្ត្រាណាមួយមកប្រើនៅក្នុងកម្មវិធីណាមួយបាន យើងត្រូវប្រើបញ្ជា import ដោយធ្វើដូចខាងក្រោមនេះៈ

```
import khmer
```

import khmer គឺជាការយកបញ្ជា import មកប្រើដើម្បីយកសាស្ត្រាឈ្មោះ khmer មកប្រើនៅក្នុងកម្មវិធីខាងលើ។ បន្ទាប់ពីនេះមក យើងនឹងអាចយកអ្វីៗទាំងអស់ដែលមាននៅក្នុងសាស្ត្រាឈ្មោះ khmer នេះមកប្រើបានតាមចិត្ត។

នៅពេលដែលយើងយកសាស្ត្រាឈ្មោះ khmer មកប្រើនៅក្នុងកម្មវិធីខាងលើ ការស្វែងរកឯកសារនេះត្រូវធ្វើឡើងនៅក្នុងថតដែលមានឯកសារដែលជាកម្មវិធីខាងលើនេះមុនគេ។ ហើយក្រោយពីសាស្ត្រានេះត្រូវបានរកឃើញ ផ្នែកទន់បកប្រែនឹងបកប្រែសាស្ត្រានេះឲ្យទៅជាឯកសារមួយទៀតដែលមានឈ្មោះជា .pyc នៅខាងចុង។ បន្ទាប់មកទៀត វត្ថុមួយដែលជាសាស្ត្រាឈ្មោះ khmer ត្រូវបានបង្កើតឡើងនៅក្នុងសតិរបស់កំព្យូទ័រ ហើយបញ្ជាទាំងឡាយនៅក្នុងសាស្ត្រានោះ ក៏ត្រូវបានយកទៅអនុវត្តដែរ ដែលជាប្រការធ្វើឲ្យវត្ថុមួយចំនួនទៀតត្រូវបានបង្កើតឡើងនិងទុកនៅក្នុងដែនកំណត់ដែលជាសាស្ត្រានោះ។ ឈ្មោះ khmer ត្រូវបានបង្កើតឡើងនិងភ្ជាប់ទៅនឹងវត្ថុដែលជាសាស្ត្រានោះ។ ដូចនេះ សាស្ត្រាខាងលើមានឈ្មោះបីគឺឈ្មោះ khmer នៅក្នុងសតិរបស់កំព្យូទ័រ និងឈ្មោះ khmer.pyc និង khmer.py នៅក្នុងថាសរឹង។

គ្រប់វត្ថុទាំងឡាយណាដែលត្រូវបានបង្កើតឡើងនិងទុកនៅក្នុងសាស្ត្រាត្រូវហៅថា *សម្បត្តិសាស្ត្រា* (module attribute) ។ វត្ថុទាំងនោះអាចជា ទិន្នន័យ ក្បួន និង ថ្នាក់មួយចំនួន។

ក្រោយពីវត្ថុដែលជាសាស្ត្រាត្រូវបានបង្កើតឡើងរួចហើយ យើងអាចពិនិត្យមើលសម្បត្តិទាំងឡាយនៅក្នុងនោះដោយធ្វើដូចខាងក្រោមនេះ៖

```
import khmer
help(khmer)
```

help(khmer) គឺជាបញ្ជាតម្រូវឲ្យពិនិត្យមើលសម្បត្តិទាំងឡាយដែលមាននៅក្នុងសាស្ត្រាឈ្មោះ khmer ។

ការយកសម្បត្តិសាស្ត្រាមកប្រើ

ដើម្បីយកសម្បត្តិនៅក្នុងសាស្ត្រាណាមួយមកប្រើ យើងត្រូវធ្វើដូចខាងក្រោមនេះ៖

```
import khmer

print(khmer.pi)
print(khmer.ផ្ទៃរង្វង់(15))
ត្រីកោណ = khmer.ធរណីមាត្រ()
print(ត្រីកោណ)
```

print(khmer.pi) គឺជាបញ្ជាដែលនៅក្នុងមានការតម្រូវឲ្យយកវត្ថុឈ្មោះ pi ដែលជាទិន្នន័យនៅក្នុងសាស្ត្រាឈ្មោះ khmer មកប្រើ។

print(khmer.ផ្ទៃរង្វង់(15)) គឺជាបញ្ជាដែលនៅក្នុងមានការតម្រូវឲ្យយកវត្ថុឈ្មោះ ផ្ទៃរង្វង់ ដែលជាក្បួននៅក្នុងសាស្ត្រាឈ្មោះ khmer មកប្រើ។

ត្រីកោណ = khmer.ធរណីមាត្រ() គឺជាបញ្ជាដែលនៅក្នុងមានការតម្រូវឲ្យយកវត្ថុឈ្មោះ ធរណីមាត្រ ដែលជាថ្នាក់នៅក្នុងសាស្ត្រាឈ្មោះ khmer មកប្រើ។

ដូចនេះយើងឃើញថា ការយកសម្បត្តិសាស្ត្រាមកប្រើ ចាំបាច់ត្រូវតែត្រូវធ្វើឡើងតាមរយៈសាស្ត្រាផ្ទាល់តែម្តង ពោលគឺយើងមិនអាចយកសម្បត្តិទាំងនោះមកប្រើដោយផ្ទាល់បានទេ។ ក៏ប៉ុន្តែមានវិធីម៉្យាងដែលអនុញ្ញាតឲ្យយើងអាចយកសម្បត្តិសាស្ត្រាមកប្រើដោយផ្ទាល់បាន។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
from khmer import pi
from khmer import ផ្ទៃរង្វង់
from khmer import ធរណីមាត្រ
```

```
print(pi)
print(ផ្ទៃរង្វង់(15))
ត្រីកោណ = ធរណីមាត្រ()
print(ត្រីកោណ)
```

from khmer import pi គឺជាការប្រើបញ្ជា from/import ដើម្បីចម្លងយកទិន្នន័យឈ្មោះ pi នៅក្នុងសាស្ត្រាឈ្មោះ khmer មកទុកប្រើ។

from khmer import *ផ្ទៃរង្វង់* គឺជាការប្រើបញ្ជា from/import ដើម្បីចម្លងយកក្បួនឈ្មោះ ផ្ទៃរង្វង់ នៅក្នុងសាស្ត្រាឈ្មោះ khmer មកទុកប្រើ។

from khmer import *ធរណីមាត្រ* គឺជាការប្រើបញ្ជា from/import ដើម្បីចម្លងយកថ្នាក់ឈ្មោះ ធរណីមាត្រ នៅក្នុងសាស្ត្រាឈ្មោះ khmer មកទុកប្រើ។

យើងសង្កេតឃើញថា បន្ទាប់ពីការប្រើបញ្ជា from/import ដើម្បីចម្លងយកវត្ថុផ្សេងៗនៅក្នុងសាស្ត្រាមកទុកប្រើ យើងអាចយកវត្ថុទាំងនោះមកប្រើដោយផ្ទាល់បានដោយមិនចាំបាច់ប្រើឈ្មោះរបស់សាស្ត្រា។

ក្រៅពីការប្រើបញ្ជា from/import ដើម្បីចម្លងយកសម្បត្តិមួយចំនួននៅក្នុងសាស្ត្រាមកប្រើ យើងក៏អាចប្រើបញ្ជា from/import* ដើម្បីចម្លងយកសម្បត្តិទាំងអស់នៅក្នុងសាស្ត្រាមកទុកប្រើ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
from khmer import*

print(pi)
print(ផ្ទៃរង្វង់(15))
ត្រីកោណ = ធរណីមាត្រ()
print(ត្រីកោណ)
```

from khmer import* គឺជាការប្រើបញ្ជា from/import* ដើម្បីចម្លងយកសម្បត្តិទាំងអស់ដែលមាននៅក្នុងសាស្ត្រាឈ្មោះ khmer មកទុកប្រើនៅពេលក្រោយទៀត។

ដោយការប្រើបញ្ជា from/import ឬ from/import* គឺជាការចម្លងឬការបង្កើតឈ្មោះថ្មីភ្ជាប់ទៅនឹងសម្បត្តិដែលមានឈ្មោះដូចគ្នានៅក្នុងសាស្ត្រាផ្សេងៗ ដូចនេះការយកឈ្មោះថ្មីទាំងនោះទៅភ្ជាប់នឹងវត្ថុថ្មីនឹងមិនធ្វើឲ្យមានផលប៉ះពាល់ដល់សម្បត្តិនៅក្នុងសាស្ត្រាដើមឡើយ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
from khmer import*
import khmer

pi = "តក់ៗពេញបំពង់"
print(khmer.pi)
```

pi = "តក់ៗពេញបំពង់" គឺជាការយកឈ្មោះ pi ចម្លងមកពីសាស្ត្រាឈ្មោះ khmer ទៅភ្ជាប់នឹងកម្រងអក្សរមួយ។

print(khmer.pi) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការយកទិន្នន័យឈ្មោះ pi ដែលជាសម្បត្តិនៅក្នុងសាស្ត្រាឈ្មោះ khmer មកពិនិត្យមើល។

យើងឃើញថា បន្ទាប់ពីឈ្មោះ pi នៅក្នុងកម្មវិធីខាងលើត្រូវបានយកទៅភ្ជាប់ទៅនឹងវត្ថុថ្មីរួចមក វត្ថុឈ្មោះ pi ដែលជាសម្បត្តិនៅក្នុងសាស្ត្រាឈ្មោះ khmer គ្មានបានទទួលរងនូវការប្រែប្រួលអ្វីឡើយ។ បានន័យថា ការយកឈ្មោះ pi ដែលត្រូវបានចម្លងមកពីក្នុងសាស្ត្រាឈ្មោះ khmer ទៅភ្ជាប់នឹងវត្ថុថ្មី មិនបណ្តាលឲ្យមានផលប៉ះពាល់អ្វីដល់វត្ថុឈ្មោះ pi ដូចគ្នានៅក្នុងសាស្ត្រានោះឡើយ។

មួយវិញទៀត ក្រៅពីការយកសាស្ត្រាមកប្រើតាមរយៈឈ្មោះដើមរបស់ដើមរបស់វា យើងក៍អាចយកសាស្ត្រាមកប្រើតាមរយៈឈ្មោះថ្មីមួយផ្សេងទៀតដែរ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
import khmer as ខ្មែរ

ត្រីកោណ = ខ្មែរ.ធរណីមាត្រ()
print(ត្រីកោណ)
```

import khmer as *ខ្មែរ* គឺជាការយកសាស្ត្រាឈ្មោះ khmer មកប្រើដោយដាក់ឈ្មោះថ្មីឲ្យវាមួយទៀតថា ខ្មែរ ។

ត្រីកោណ = ខ្មែរ.ធរណីមាត្រ() គឺជាការយកសម្បត្តិឈ្មោះ ធរណីមាត្រ នៅក្នុងសាស្ត្រាឈ្មោះ khmer ឬ ខ្មែរ មកប្រើ។

ដូចគ្នាដែរ យើងអាចយកសម្បត្តិនៅក្នុងសាស្ត្រាណាមួយមកទុកប្រើដោយដាក់ឈ្មោះថ្មីឲ្យវា ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
from khmer import pi as ពី

print(ពី)
```

from khmer import pi as *ពី* គឺជាការប្រើបញ្ជា from/import/as ដើម្បីយកទិន្នន័យឈ្មោះ pi មកទុកប្រើដោយដាក់ឈ្មោះថ្មីមួយទៀតឲ្យវាគឺ ពី ។

កញ្ចប់

ការបង្កើត កញ្ចប់

កញ្ចប់ (package) គឺជាថតមួយដែលមានសាស្ត្រាមួយចំនួននៅក្នុងនោះ។ ហើយនៅក្នុងចំណោមសាស្ត្រាទាំងនោះ មានសាស្ត្រាមួយមានឈ្មោះថា __init__.py ។ ដូចនេះដើម្បីបង្កើតកញ្ចប់ យើងគ្រាន់តែប្រមូលសាស្ត្រាមួយចំនួនរួមជាមួយសាស្ត្រាមួយមានឈ្មោះថា __init__.py មកដាក់នៅក្នុងថតជាមួយគ្នា គឺជាការស្រេច។ ឈ្មោះរបស់ថតនោះនឹងក្លាយទៅជាឈ្មោះរបស់កញ្ចប់។

ឈ្មោះរបស់កញ្ចប់ក៏ជាឈ្មោះមួយដូចជាឈ្មោះរបស់ដទៃទៀតនៅក្នុងភាសា Python ដែរ ពោលគឺយើងត្រូវតែគោរពទៅតាមវិធាននៃការបង្កើតឈ្មោះនៅក្នុងភាសា Python ដើម្បីបង្កើតឈ្មោះរបស់កញ្ចប់។

ជាកិច្ចចាប់ផ្តើម ចូរយើងបង្កើតកញ្ចប់មួយមានឈ្មោះថា KhmerPython ដោយធ្វើដូចខាងក្រោមនេះ៖

```
#ការបង្កើតសាស្ត្រាឈ្មោះ geometry
ពី = 3.14
def អង្កត់ផ្ចិត(កាំ) :
    return កាំ * 2

class មាឌ() :
    def ក្រឡាផ្ទៃ(សិស្ស) :
```

```
        print("ក្រឡាផ្ទៃ")

    def មាឌ(សិស្ស) :
        print("មាឌ")
```

បើយើងរក្សាកម្មវិធីខាងលើនេះទុកដោយដាក់ឈ្មោះថា geometry.py យើងនឹងបានសាស្ត្រាមួយមានឈ្មោះថា geometry ។

```
#ការបង្កើតសាស្ត្រាឈ្មោះ equation
def សមីការបន្ទាត់() :
    print("សមីការបន្ទាត់")

class សមីការ() :
    def ដឺក្រេទីពីរ(សិស្ស) :
        print("សមីការដឺក្រេទីពីរ")

    def ដឺក្រេទីបី(សិស្ស) :
        print("សមីការដឺក្រេទីបី")
```

បើយើងរក្សាកម្មវិធីខាងលើនេះទុកដោយដាក់ឈ្មោះថា equation.py យើងនឹងបានសាស្ត្រាមួយមានឈ្មោះថា equation ។

```
#ការបង្កើតសាស្ត្រាឈ្មោះ __init__
#សាស្ត្រា __init__
```

បើយើងរក្សាកម្មវិធីខាងលើនេះទុកដោយដាក់ឈ្មោះថា __init__.py យើងនឹងបានសាស្ត្រាមួយមានឈ្មោះថា __init__ ។

បើយើងប្រមូលសាស្ត្រាឈ្មោះ geometry, equqtion, __init__ ខាងលើមកដាក់នៅក្នុងថត មួយជាមួយគ្នា យើងនឹងបានកញ្ចប់មួយ ហើយឈ្មោះរបស់ថតនោះនឹងក្លាយទៅជាឈ្មោះរបស់របស់កញ្ចប់នោះ។ ជាក់ស្តែងបើយើងប្រមូលយកសាស្ត្រា geometry, equation, __init__ មកដាក់នៅក្នុងថតមួយដែលមានឈ្មោះថា KhmerPython ជាមួយគ្នា យើងនឹងបានកញ្ចប់មួយមានឈ្មោះថា KhmerPython ។

ម៉្យាងទៀត យើងត្រូវធ្វើការកត់សំគាល់ថា ដើម្បីបង្កើតកញ្ចប់ណាមួយ ទាមទារចាំបាច់ឲ្យមានសាស្ត្រាមានឈ្មោះថា __init__ មួយ បើពុំនោះសោតទេកំហុសនឹងកើតមានឡើង។ ហើយសាស្ត្រាឈ្មោះ __init__ អាចជាសាស្ត្រាទទេឬអាចជាសាស្ត្រាដែលមានសម្បត្តិផ្សេងៗនៅក្នុងនោះ។

ការយកសាស្ត្រាក្នុងកញ្ចប់មកប្រើ

ដើម្បីយកសាស្ត្រាក្នុងកញ្ចប់ណាមួយមកប្រើ យើងត្រូវធ្វើដូចខាងក្រោមនេះ៖

```
import KhmerPython.equation

KhmerPython.equation.សមីការបន្ទាត់()
```

import KhmerPython.equation គឺជាការយកសាស្ត្រាឈ្មោះ equation នៅក្នុងកញ្ចប់ឈ្មោះ KhmerPython មកប្រើ។

យើងត្រូវធ្វើការកត់សំគាល់ថា ដើម្បីអាចយកសាស្ត្រានៅក្នុងកញ្ចប់ណាមួយមកប្រើបានតាមរបៀបដូចខាងលើនេះ កញ្ចប់នោះត្រូវតែស្ថិតនៅក្នុងថតជាមួយនឹងឯកសារដែលជាកម្មវិធី ឬស្ថិតនៅក្នុងកន្លែងណាមួយដែលផ្នែកទន់បកប្រែអាចរកឃើញ។

KhmerPython.equation.*សមីការបន្ទាត់*() គឺជាការយកក្បួនឈ្មោះ សមីការបន្ទាត់ នៅក្នុងសាស្ត្រាឈ្មោះ equation នៃកញ្ចប់ឈ្មោះ KhmerPython មកប្រើការ។

ការយកសាស្ត្រាផ្សេងៗនៅក្នុងកញ្ចប់ណាមួយមកប្រើ អាចត្រូវធ្វើតាមរបៀបម៉្យាងទៀតដូចខាងក្រោមនេះ៖

```
from KhmerPython import equation
equation.សមីការបន្ទាត់()
```

from KhmerPython import equation គឺជាការចម្លងយកសាស្ត្រាឈ្មោះ equation នៅក្នុងកញ្ចប់ឈ្មោះ KhmerPython មកប្រើ។

equation.*សមីការបន្ទាត់*() គឺជាការយកក្បួនឈ្មោះ សមីការបន្ទាត់ នៅក្នុងសាស្ត្រាឈ្មោះ equation មកប្រើ។

យើងអាចចម្លងយកសម្បត្តិទាំងអស់នៅក្នុងសាស្ត្រាណាមួយដែលស្ថិតនៅក្នុងកញ្ចប់ណាមួយមកទុកប្រើ ដោយធ្វើដូចខាងក្រោមនេះ៖

```
from KhmerPython.equation import*
សមីការបន្ទាត់()
សមីការ១ = សមីការ()
សមីការ១.ដឺក្រេទីពីរ()
សមីការ១.ដឺក្រេទីបី()
```

from KhmerPython.equation import* គឺជាការចម្លងយកសម្បត្តិទាំងអស់នៅក្នុងសាស្ត្រាឈ្មោះ equation ដែលស្ថិតនៅក្នុងកញ្ចប់ឈ្មោះ KhmerPython មកទុកប្រើ។

មួយវិញទៀត យើងក៏អាចចម្លងយកសាស្ត្រាណាមួយនៅក្នុងកញ្ចប់ណាមួយមកដាក់ឈ្មោះថ្មីមួយទៀតបានដែរ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
import KhmerPython.equation as គណិតវិទ្យា

គណិតវិទ្យា.សមីការបន្ទាត់()
សមីការ១ = គណិតវិទ្យា.សមីការ()
សមីការ១.ដឺក្រេទីពីរ()
សមីការ១.ដឺក្រេទីបី()
```

import KhmerPython.equation as *គណិតវិទ្យា* គឺជាការចម្លងយកសាស្ត្រាឈ្មោះ equation នៅក្នុងកញ្ចប់ឈ្មោះ KhmerPython មកដាក់ឈ្មោះថ្មីថា គណិតវិទ្យា ។

គណិតវិទ្យា.សមីការបន្ទាត់() គឺជាការយកក្បួនឈ្មោះ សមីការបន្ទាត់ ស្ថិតនៅក្នុងសាស្ត្រាឈ្មោះ equation ដែលមានឈ្មោះថ្មីថា គណិតវិទ្យា មកប្រើ។

ដូចគ្នាដែរ យើងក៏អាចចម្លងយកសម្បត្តិនៅក្នុងសាស្ត្រាណាមួយដែលស្ថិតនៅក្នុងកញ្ចប់ណាមួយមកដាក់ឈ្មោះថ្មី ដោយធ្វើដូចខាងក្រោមនេះ៖

```
from KhmerPython.equation import សមីការបន្ទាត់ as line

line()
```

from KhmerPython.equation import *សមីការបន្ទាត់* as line គឺជាការចម្លងយកក្បួនឈ្មោះ សមីការបន្ទាត់ នៅក្នុងសាស្ត្រាឈ្មោះ equation ដែលស្ថិតនៅក្នុងកញ្ចប់ឈ្មោះ KhmerPython មកដាក់ឈ្មោះថ្មីថា line ។

line() គឺជាការយកក្បួនឈ្មោះ សមីការបន្ទាត់ ដែលមានឈ្មោះថ្មីថា line មកប្រើ។

បណ្ណាល័យ មជ្ឈឹម

បណ្ណាល័យមជ្ឈឹម (standard library) គឺជាថតមួយដែលនៅក្នុងនោះមានសាស្ត្រាមួយចំនួនដែលត្រូវបានបង្កើតឡើងរួចជាស្រេច សម្រាប់ឲ្យយើងយកទៅប្រើការក្នុងការដោះស្រាយបញ្ហាទូទៅជាច្រើន។ នៅក្នុងបណ្ណាល័យមជ្ឈឹមនោះ មានសាស្ត្រាមានស្រាប់លើសពី 200 ដែលមិនជាប់ទាក់ទងឬអាស្រ័យទៅនឹងប្រព័ន្ធប្រតិបត្តិការណាមួយឡើយ។ បានន័យថាយើងអាចយកសាស្ត្រាទាំងនោះទៅប្រើការសម្រាប់សរសេរកម្មវិធីជាភាសា Python និងតម្រូវឲ្យដំណើរការនៅក្នុងប្រព័ន្ធប្រតិបត្តិការណាមួយក៏បានដែរ។ លើសពីនេះទៀត យើងអាចបង្កើតសាស្ត្រាផ្សេងៗទៀតទុកនៅក្នុងបណ្ណាល័យមជ្ឈឹមនោះ បន្ថែមពីលើសាស្ត្រាមានស្រាប់ទាំងនោះទៀតបានផង។ ដើម្បីឲ្យដឹងថាតើនៅក្នុងបណ្ណាល័យកណ្តាលនោះមានសាស្ត្រាអ្វីខ្លះនោះ យើងអាចអានព័ត៌មានបន្ថែមនៅលើគេហទំព័រ www.python.org ។

ការស្វែង រកសាស្ត្រា

នៅក្នុងភាសា Python នៅពេលដែលយើងយកសាស្ត្រាណាមួយមកប្រើ ការស្វែងរកសាស្ត្រានោះត្រូវធ្វើឡើងនៅក្នុងថតមួយចំនួនដែលត្រូវបានកំណត់ទុកជាមុនរួចជាស្រេច។ ហើយដើម្បីឲ្យដឹងថាតើថតដែលត្រូវបានកំណត់ទុកមុនរួចជាស្រេចនោះជាថតអ្វីខ្លះនោះ យើងត្រូវធ្វើដូចខាងក្រោមនេះ៖

```
import KhmerPython.equation
import sys

print(sys.path)
```

print(sys.path) គឺជាការយកសម្បត្តិឈ្មោះ path នៅក្នុងសាស្ត្រាឈ្មោះ sys មកពិនិត្យមើល។

យើងឃើញថា សម្បត្តិឈ្មោះ path នេះគឺជាកម្រងអថេរមួយដែលមានធាតុទាំងឡាយជាឈ្មោះនៃថតមួយចំនួនដែលត្រូវបានកំណត់ទុកមុនរួចជាស្រេចសម្រាប់ការស្វែងរកសាស្ត្រាផ្សេងៗដែលយើងយកមកប្រើ។ បានន័យថា ការស្វែងរករកសាស្ត្រាផ្សេងៗគឺត្រូវធ្វើឡើងនៅក្នុងថតទាំងឡាយដែលជាធាតុនៃកម្រងអថេរឈ្មោះ path នោះដោយចាប់ផ្តើមពីឆ្វេងទៅស្តាំ។

យ៉ាងណាមិញ ដោយសម្បត្តិឈ្មោះ path ជាកម្រងអថេរ ដូចនេះយើងអាចធ្វើការកែប្រែការស្វែងរកសាស្ត្រាដែលត្រូវយកមកប្រើ ដោយធ្វើការកែប្រែធាតុនៃកម្រងអថេរឈ្មោះ path នេះ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
import KhmerPython.equation
import sys

print(sys.path)
sys.path.append("c:\\")
print(sys.path)
```

sys.path.append("c:\\") គឺជាការបន្ថែមថតឈ្មោះ c:\ មួយទៀតចូលទៅក្នុងកម្រងអថេរឈ្មោះ path ក្នុងគោលបំណងធ្វើឲ្យការស្វែងរកសាស្ត្រាផ្សេងៗត្រូវធ្វើឡើងនៅក្នុងថតនោះមួយទៀត។

ភាពមិនប្រក្រតី

ភាពមិនប្រក្រតី (exception) គឺជារឿងហេតុមិនធម្មតាទាំងឡាយណាដែលកើតមានឡើងនៅពេលដែលកម្មវិធីកំពុងតែដំណើរការ។ កន្លងមក យើងបានជួបប្រទះនឹងភាពមិនប្រក្រតីមួយចំនួនរួចទៅហើយ មានដូចជាកំហុសផ្សេងៗដែលបណ្តាលមកពីការសរសេរបញ្ជាមិនត្រឹមត្រូវជាដើម។ល។

ប្រភេទនៃភាពមិនប្រក្រតី

ភាពមិនប្រក្រតីមានច្រើនប្រភេទណាស់ ដូចជានៅពេលដែលយើងសរសេរបញ្ជាមិនត្រឹមត្រូវទៅតាមក្បួនវេយ្យាករណ៍នៅក្នុងភាសា Python ភាពមិនប្រក្រតីប្រភេទ SyntaxError នឹងកើតមានឡើង។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
print("អ្នកមានវក្យាខ្សត់"
```

prin("អ្នកមានវក្យាខ្សត់" គឺជាការយកក្បួនមានស្រាប់ឈ្មោះ print មកប្រើដោយភ្លេចសរសេរសញ្ញាវង្វង់ក្រចកនៅខាងចុងគេ។ ប្រការនេះបានបណ្តាលឲ្យកំហុសប្រភេទ SyntaxError បានកើតមានឡើង។

តាមពិតភាពមិនប្រក្រតីគឺជាសិស្សនៃថ្នាក់ដែលជាប្រភេទនៃភាពមិនប្រក្រតីនោះ។ ជាក់ស្តែងដូចជានៅពេលដែលយើងយកក្បួនមានស្រាប់ឈ្មោះ print មកប្រើមិនត្រឹមត្រូវតាមក្បួនខ្នាតសិស្សនៃថ្នាក់ឈ្មោះ SyntaxError ម្នាក់ត្រូវបានបង្កើតឡើង។

ក្រៅពីថ្នាក់ឈ្មោះ SyntaxError នេះ នៅមានថ្នាក់នៃភាពមិនប្រក្រតីជាច្រើនទៀតដូចខាងក្រោមនេះ៖

```
BaseException
 +-- SystemExit
 +-- KeyboardInterrupt
 +-- GeneratorExit
 +-- Exception
      +-- StopIteration
      +-- ArithmeticError
      |    +-- FloatingPointError
      |    +-- OverflowError
      |    +-- ZeroDivisionError
      +-- AssertionError
      +-- AttributeError
      +-- BufferError
      +-- EnvironmentError
      |    +-- IOError
      |    +-- OSError
      |         +-- WindowsError (Windows)
      |         +-- VMSError (VMS)
      +-- EOFError
      +-- ImportError
      +-- LookupError
      |    +-- IndexError
      |    +-- KeyError
      +-- MemoryError
      +-- NameError
      |    +-- UnboundLocalError
      +-- ReferenceError
      +-- RuntimeError
      |    +-- NotImplementedError
      +-- SyntaxError
      |    +-- IndentationError
      |         +-- TabError
      +-- SystemError
      +-- TypeError
      +-- ValueError
      |    +-- UnicodeError
      |         +-- UnicodeDecodeError
      |         +-- UnicodeEncodeError
      |         +-- UnicodeTranslateError
      +-- Warning
```

```
            +-- DeprecationWarning
            +-- PendingDeprecationWarning
            +-- RuntimeWarning
            +-- SyntaxWarning
            +-- UserWarning
            +-- FutureWarning
            +-- ImportWarning
            +-- UnicodeWarning
            +-- BytesWarning
```

បញ្ជា try/except

try/except គឺជាបញ្ជាតម្រូវឲ្យសាកល្បងអនុវត្តក្រុមបញ្ជានៅក្នុងបញ្ជា try និងតម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងបញ្ជា except ក្នុងករណីការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងបញ្ជា try បង្កឲ្យមានភាពមិនប្រក្រតីកើតឡើង។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
try :
    5/0
except :
    print("ភាពមិនប្រក្រតីបានកើតមានឡើង។")
```

try : គឺជាបញ្ជាតម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងនោះ។

except : គឺជាបញ្ជាតម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងនោះ ក្នុងករណីការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងបញ្ជា try បណ្តាលឲ្យមានភាពមិនប្រក្រតីកើតមានឡើង។

ដោយក្រុមបញ្ជានៅក្នុងបញ្ជា try ជាកន្សោមប្រមាណវិធី 5/0 ដូចនេះការអនុវត្តន៍បញ្ជានេះបណ្តាលឲ្យភាពមិនប្រក្រតីបានកើតមានឡើង ព្រោះ 5 មិនអាចចែកនឹង 0 បានទេ។ ជាផលវិបាក ក្រុមបញ្ជានៅក្នុងបញ្ជា except ត្រូវបានយកទៅអនុវត្ត។

ក្នុងករណីមានភាពមិនប្រក្រតីកើតមានឡើង បើយើងមិនធ្វើអ្វីសោះនោះទេ កម្មវិធីនឹងឈប់លែងដំណើរការ។ តែបើយើងប្រើបញ្ជា try/except នេះ នៅពេលមានភាពមិនប្រក្រតីកើតមានឡើង ក្រុមបញ្ជានៅក្នុងបញ្ជា except ត្រូវយកទៅអនុវត្ត ហើយបញ្ជាបន្ទាប់ពីក្រុមបញ្ជានៅក្នុងបញ្ជា except នឹងត្រូវយកទៅអនុវត្តជាបន្តទៅទៀត។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
try :
    5/0
    print("ការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងបញ្ជា try ត្រូវចប់ត្រឹមនេះ។")
except :
    print("ភាពមិនប្រក្រតីបានកើតមានឡើង។")
    print("ការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងបញ្ជា except ត្រូវចប់ត្រឹមនេះ។")

print("កម្មវិធីត្រូវចប់ត្រឹមនេះ។")
```

print("*កម្មវិធីត្រូវចប់ត្រឹមនេះ។*") គឺជាបញ្ជាដែលត្រូវយកទៅអនុវត្តជាបន្តទៅទៀត បន្ទាប់ពីក្រុមបញ្ជានៅក្នុងបញ្ជា except ត្រូវបានអនុវត្តរួចហើយ។

យើងឃើញថា នៅពេលមានភាពមិនប្រក្រតីកើតមានឡើង បណ្តាលមកពីការអនុវត្តន៍បញ្ជាដែលជាកន្សោមប្រមាណវិធី 5/0 បញ្ជាបន្ទាប់ពីនោះដែលស្ថិតនៅក្នុងក្រុមជាមួយគ្នាត្រូវទុកចោល។

ការប្រើបញ្ជា try/except នៅក្នុងកម្មវិធីហៅថា *ការទទួលយកភាពមិនប្រក្រតី* (catching exception) ។ ពីព្រោះបើភាពមិនប្រក្រតីកើតមានឡើង ក្រុមបញ្ជានៅក្នុងបញ្ជា except នឹងត្រូវយកទៅអនុវត្ត កម្មវិធីនឹងបន្តដំណើរការជាបន្តទៅទៀត។ បានន័យថា ភាពមិនប្រក្រតីត្រូវបានទទួលយក។ ផ្ទុយទៅវិញ បើយើងមិនប្រើបញ្ជា try/except នេះដើម្បីទទួលយក

ភាពមិនប្រក្រតីនោះទេ កម្មវិធីនឹងឈប់លែងដំណើរការនៅត្រង់កន្លែងដែលមានភាពមិនប្រក្រតីនោះ។

ការទទួលយកភាពមិនប្រក្រតីអាចត្រូវធ្វើឡើងតាមរបៀបម៉្យាងទៀតដូចខាងក្រោមនេះ៖

```
try :
    5/0
    print("ការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងបញ្ជា try ត្រូវចប់ត្រឹមនេះ។")
except ArithmeticError :
    print("ភាពមិនប្រក្រតីបានកើតមានឡើង។")
    print("ការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងបញ្ជា except ត្រូវចប់ត្រឹមនេះ។")
```

except ArithmeticError : គឺជាបញ្ជាតម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងនោះ ក្នុងករណីការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងបញ្ជា try បង្ករឲ្យមានភាពមិនប្រក្រតីប្រភេទ ArithmeticError កើតមានឡើង។ បានន័យថា បើភាពមិនប្រក្រតីគ្មានប្រភេទជា ArithmeticError នោះទេ ភាពមិនប្រក្រតីនឹងមិនត្រូវបានទទួលយកឡើយ ហើយកម្មវិធីនឹងឈប់លែងដំណើរការ។

ដោយភាពមិនប្រក្រតីជាវត្ថុដែលជាសិស្សនៃថ្នាក់ណាមួយ ដូចនេះយើងអាចទទួលយកភាពមិនប្រក្រតីដោយដាក់ឈ្មោះឲ្យវាដូចខាងក្រោមនេះ៖

```
try :
    5/0
    print("ការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងបញ្ជា try ត្រូវចប់ត្រឹមនេះ។")
except ArithmeticError as កំហុស :
    print("ភាពមិនប្រក្រតីបានកើតមានឡើង។")
    print("ការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងបញ្ជា except ត្រូវចប់ត្រឹមនេះ។")
```

except ArithmeticError as *កំហុស* : គឺជាបញ្ជាតម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងនោះ និងភ្ជាប់ឈ្មោះ កំហុស ទៅនឹងវត្ថុដែលជាភាពមិនប្រក្រតី នៅពេលដែលមានភាពមិនប្រក្រតីប្រភេទ ArithmeticError កើតមានឡើង។

ក្រៅពីការទទួលយកភាពមិនប្រក្រតីតែមួយប្រភេទ យើងអាចទទួលយកភាពមិនប្រក្រតីជាច្រើនប្រភេទផ្លាស់គ្នា ដោយធ្វើដូចខាងក្រោមនេះ៖

```
try :
   5/0
   print("ការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងបញ្ជា try ត្រូវចប់ត្រឹមនេះ។")
except (ArithmeticError, NameError) :
   print("ភាពមិនប្រក្រតីបានកើតមានឡើង។")
   print("ការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងបញ្ជា except ត្រូវចប់ត្រឹមនេះ។")
```

except (ArithmeticError, NameError) : គឺជាបញ្ជាតម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងនោះ នៅពេលដែលមានភាពមិនប្រក្រតីប្រភេទ ArithmeticError ឬ NameError កើតមានឡើង។

យើងអាចទទួលយកភាពមិនប្រក្រតីបានជាច្រើនប្រភេទផ្លាស់គ្នា និងភ្ជាប់ឈ្មោះណាមួយទៅនឹងវត្ថុនៃភាពមិនប្រក្រតីនោះ ដោយសរសេរកម្មវិធីដូចខាងក្រោមនេះ៖

```
try :
   5/0
   print("ការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងបញ្ជា try ត្រូវចប់ត្រឹមនេះ។")
except (ArithmeticError, NameError) as កំហុស :
   print("ភាពមិនប្រក្រតីបានកើតមានឡើង។")
   print("ការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងបញ្ជា except ត្រូវចប់ត្រឹមនេះ។")
```

except (ArithmeticError, NameError) as *កំហុស* : គឺជាបញ្ជាតម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងនោះ និងភ្ជាប់ឈ្មោះ កំហុស ទៅនឹងវត្ថុនៃភាពមិនប្រក្រតី ក្នុងករណីភាពមិនប្រក្រតីមានប្រភេទជា ArithmeticError ឬ NameError ។

ក្រៅពីការប្រើប្រាស់បញ្ជា except តែមួយ យើងអាចប្រើបញ្ជា except ជាច្រើនដើម្បីតម្រូវឲ្យអនុវត្តក្រុមបញ្ជាផ្សេងៗគ្នា អាស្រ័យទៅតាមប្រភេទនៃភាពមិនប្រក្រតី។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
try :
    5/0
    print("ការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងបញ្ជា try ត្រូវចប់ត្រឹមនេះ។")
except (ArithmeticError, NameError) as កំហុស :
    print("ភាពមិនប្រក្រតីបានកើតមានឡើង។")
    print("ការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងបញ្ជា except ត្រូវចប់ត្រឹមនេះ។")
except MemoryError as កំហុស :
    print("កំហុសមានប្រភេទជា MemoryError បានកើតមានឡើង។")
```

except (ArithmeticError, NameError) as *កំហុស* : គឺជាបញ្ជាតម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងនោះ និងភ្ជាប់ឈ្មោះ កំហុស ទៅនឹងវត្ថុនៃភាពមិនប្រក្រតី ក្នុងករណីភាពមិនប្រក្រតីមានប្រភេទជា ArithmeticError ឬ NameError ។

except MemoryError as *កំហុស* : គឺជាបញ្ជាតម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងនោះ និងភ្ជាប់ឈ្មោះ កំហុស ទៅនឹងវត្ថុនៃភាពមិនប្រក្រតី ក្នុងករណីភាពមិនប្រក្រតីមានប្រភេទជា MemoryError ។

បញ្ជា try/except/else

try/except/else គឺជាបញ្ជាតម្រូវឲ្យសាកល្បងអនុវត្តក្រុមបញ្ជានៅក្នុងបញ្ជា try និងតម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងបញ្ជា except ក្នុងករណីមានភាពមិនប្រក្រតីកើតឡើង។ តែបើការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងបញ្ជា try គ្មានរឿងអ្វីកើតឡើងទេ ក្រុមបញ្ជានៅក្នុងបញ្ជា except នឹងត្រូវរំលងចោល ហើយក្រុមបញ្ជានៅក្នុងបញ្ជា else នឹងត្រូវយកទៅអនុវត្ត។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
try :
    5/2
    print("ការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងបញ្ជា try ត្រូវចប់ត្រឹមនេះ។")
except :
    print("ភាពមិនប្រក្រតីបានកើតមានឡើង។")
    print("ការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងបញ្ជា except ត្រូវចប់ត្រឹមនេះ។")
else :
    print("គ្មានភាពមិនប្រក្រតីកើតឡើងទេ។")
```

try : គឺជាបញ្ជាតម្រូវឲ្យសាកល្បងអនុវត្តក្រុមបញ្ជានៅក្នុងនោះ។

except : គឺជាបញ្ជាតម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងនោះ ក្នុងករណីការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងបញ្ជា try បង្ករឲមានភាពមិនប្រក្រតីកើតមានឡើង។

else : គឺជាបញ្ជាតម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងនោះ ក្នុងករណីការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងបញ្ជា try មិនបានបង្ករឲ្យមានភាពមិនប្រក្រតីកើតមានឡើងទេ។

ដោយកន្សោមប្រមាណវិធី 5/2 ផ្តល់លទ្ធផល 2.5 ដូចនេះការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងបញ្ជា try ត្រូវបានបញ្ចប់ទៅតាមសម្រួលដោយមិនបង្ករឲ្យមានភាពមិនប្រក្រតីកើតមានឡើងឡើយ ។ ជាផលវិបាក ក្រុមបញ្ជានៅក្នុង else ត្រូវបានយកទៅអនុវត្ត។

បញ្ជា try/except/finally

try/except/finally គឺជាបញ្ជាតម្រូវឲ្យសាកល្បងអនុវត្តក្រុមបញ្ជានៅក្នុងបញ្ជា try និងតម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងបញ្ជា except ក្នុងករណីមានភាពមិនប្រក្រតីកើតឡើង និងតម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងបញ្ជា finally ទោះបីជាមានឬគ្មានភាពមិនប្រក្រតីកើតឡើងក៏ដោយ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
try :
    5/2
    print("ការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងបញ្ជា try ត្រូវចប់ត្រឹមនេះ។")
except :
    print("ភាពមិនប្រក្រតីបានកើតមានឡើង។")
    print("ការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងបញ្ជា except ត្រូវចប់ត្រឹមនេះ។")
finally :
    print("ត្រូវយកទៅអនុវត្តទោះបីជាមានឬគ្មានភាពមិនប្រក្រតីកើតឡើងក៏ដោយ។")
```

finally : គឺជាបញ្ជាតម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងនោះ ទោះជាមានឬគ្មានភាពមិនប្រក្រតីកើងឡើងក៏ដោយ។

បញ្ជា try/except/else/finally

try/except/else/finally គឺជាបញ្ជាតម្រូវឲ្យសាកល្បងអនុវត្តក្រុមបញ្ជានៅក្នុងបញ្ជា try និងតម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងបញ្ជា except ក្នុងករណីមានភាពមិនប្រក្រតីកើតឡើង និងតម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងបញ្ជា else ក្នុងករណីគ្មានភាពមិនប្រក្រតីកើតឡើង និងតម្រូវឲ្យអនុវត្តក្រុមនៅក្នុងបញ្ជា finally ទោះបីជាមានឬគ្មានភាពមិនប្រក្រតីកើតឡើងក៏ដោយ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
try :
    5/2
    print("ការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងបញ្ជា try ត្រូវចប់ត្រឹមនេះ។")
except :
    print("ភាពមិនប្រក្រតីបានកើតមានឡើង។")
    print("ការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងបញ្ជា except ត្រូវចប់ត្រឹមនេះ។")
else :
    print("គ្មានភាពមិនប្រក្រតីបានកើតឡើងទេ។")
finally :
    print("ត្រូវយកទៅអនុវត្តទោះបីជាមានឬគ្មានភាពមិនប្រក្រតីកើតឡើងក៏ដោយ។")
```

ដោយកន្សោមប្រមាណវិធី 5/2 ផ្តល់លទ្ធផលជាលេខ 2.5 ដូចនេះការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងបញ្ជា try មិនបណ្តាលឲ្យភាពមិនប្រក្រតីកើតឡើងឡើយ។ មូលហេតុនេះធ្វើឲ្យក្រុមបញ្ជានៅក្នុងបញ្ជា else ត្រូវបានយកទៅអនុវត្ត។ ចំណែកឯក្រុមបញ្ជានៅក្នុងបញ្ជា finally វិញត្រូវបានយកទៅអនុវត្តជានិច្ចទោះជាមានឬគ្មានភាពមិនប្រក្រតីកើតឡើងក៏ដោយ។

បញ្ជា raise

raise គឺជាបញ្ជាតម្រូវឲ្យបង្កើតភាពមិនប្រក្រតី ពីព្រោះនៅពេលខ្លះ យើងត្រូវបង្កើតភាពមិនប្រក្រតីដោយខ្លួនយើងផ្ទាល់។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
try :
    raise SyntaxError("សរសេរបញ្ជាខុស។")
except SyntaxError as កំហុស :
    print(កំហុស)
```

raise SyntaxError("សរសេរបញ្ជាខុស។") គឺជាការប្រើបញ្ជា raise ដើម្បីបង្កើតភាពមិនប្រក្រតីប្រភេទ SyntaxError ។

ក្រៅពីការបង្កើតភាពមិនប្រក្រតីមានប្រភេទជា SyntaxError យើងអាចបង្កើតភាពមិនប្រក្រតីប្រភេទណាក៏បានដែរដោយប្រើប្រាស់បញ្ជា raise នេះ។

បញ្ជា assert

assert គឺជាបញ្ជាតម្រូវឲ្យបង្កើតភាពមិនប្រក្រតីប្រភេទ AssertionError ក្នុងករណីកន្សោមប្រមាណវិធីមួយផ្តល់លទ្ធផលជាតក្កវត្ត False ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
ថ្លៃលក់ = 800
ថ្លៃទិញ = 900
try :
   assert ថ្លៃលក់ > ថ្លៃទិញ
except AssertionError :
   print("ភាពមិនប្រក្រតីប្រភេទ AssertionError បានកើតមានឡើង។")
```

assert ថ្លៃលក់ > ថ្លៃទិញ គឺជាបញ្ជាតម្រូវឲ្យបង្កើតភាពមិនប្រក្រតីប្រភេទ AssertionError ក្នុងករណីកន្សោមប្រមាណវិធី ថ្លៃ > ថ្លៃទិញ ផ្តល់លទ្ធផលជាតក្កវត្ត False ។

ដោយវត្ថុឈ្មោះ ថ្លៃលក់ ជាលេខ 800 និងវត្ថុឈ្មោះ ថ្លៃទិញ ជាលេខ 900 ដូចនេះកន្សោមប្រមាណវិធី ថ្លៃលក់ > ថ្លៃទិញ ផ្តល់លទ្ធផលជាតក្កវត្ត False ដែលធ្វើឲ្យភាពមិនប្រក្រតីប្រភេទ AssertionError ត្រូវកើតមានឡើង។

ការបង្កើតថ្នាក់នៃភាពមិនប្រក្រតី

ក្រៅពីភាពមិនប្រក្រតីដែលជាសិស្សនៃថ្នាក់មានស្រាប់មួយចំនួនដូចខាងលើនេះ យើងអាចបង្កើតថ្នាក់នៃភាពមិនប្រក្រតីមួយចំនួនទៀត ដើម្បីបង្កើតភាពមិនប្រក្រតីប្រភេទផ្សេងៗទៀត

។ យ៉ាងណាមិញ ថ្នាក់នៃភាពមិនប្រក្រតី ក៏ដូចជាថ្នាក់ធម្មតាឯទៀតៗដែរ ដូចនេះ ការបង្កើតថ្នាក់នៃភាពមិនប្រក្រតី គ្មានការខុសប្លែកអ្វីពីការបង្កើតថ្នាក់ធម្មតាដទៃទៀតឡើយ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
class កំហុសថ្មី(Exception) :
   def __str__(សិស្ស) :
      return "ភាពមិនប្រក្រតីប្រភេទ កំហុសថ្មី។"

try :
   raise កំហុសថ្មី()
except កំហុសថ្មី as កំហុស :
   print(កំហុស)
```

class កំហុសថ្មី(Exception) : គឺជាការបង្កើតថ្នាក់ឈ្មោះ កំហុសថ្មី បន្តភ្ជាប់ទៅនឹងថ្នាក់មានស្រាប់ឈ្មោះ Exception ។

ជារួម គេនិយមបង្កើតថ្នាក់នៃភាពមិនប្រក្រតីថ្មី ដោយបន្តភ្ជាប់ទៅនឹងថ្នាក់មានស្រាប់ឈ្មោះ Exception ។

ក្នុងករណីដែលថ្នាក់នៃភាពមិនប្រក្រតីមួយត្រូវបានបង្កើតឡើងដោយបន្តភ្ជាប់ទៅនឹងថ្នាក់មេផ្សេងៗទៀត តាមរយៈថ្នាក់មេ យើងអាចទទួលយកភាពមិនប្រក្រតីដែលជាសិស្សនៃថ្នាក់រងបាន។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
class កំហុសថ្មី(Exception) :
   def __str__(សិស្ស) :
      return "ភាពមិនប្រក្រតីប្រភេទ កំហុសថ្មី។"

try :
```

```
        raise កំហុសថ្មី()
    except Exception as កំហុស :
        print(កំហុស)
```

except Exception as *កំហុស* : គឺជាបញ្ជាតម្រូវឲ្យទទួលយកភាពមិនប្រក្រតីមានប្រភេទជា Exception ឬថ្នាក់រងនៃថ្នាក់ឈ្មោះ Exception នេះ។

ដោយភាពមិនប្រក្រតីដែលត្រូវបានបង្កើតឡើងជាសិស្សនៃថ្នាក់ឈ្មោះ កំហុសថ្មី ដែលជាថ្នាក់រងនៃថ្នាក់ឈ្មោះ Exception ដូចនេះភាពមិនប្រក្រតីនេះត្រូវបានទទួលយកដូចជាសិស្សនៃថ្នាក់មេឈ្មោះ Exception នោះដែរ។

បច្ចេកទេសជាន់ខ្ពស់

ក្រៅពីបញ្ជានិងក្បួនខ្នាតមួយចំនួនដែលយើងបានប្រើប្រាស់កន្លងមកសម្រាប់ដោះស្រាយបញ្ហាផ្សេងៗ នៅមាន *បច្ចេកទេសជាន់ខ្ពស់* (advanced feature) មួយចំនួនទៀតដែលយើងអាចយកមកប្រើប្រាស់បន្ថែមទៅលើក្បួនខ្នាតទាំងអស់នោះ។ បច្ចេកទេសជាន់ខ្ពស់ទាំងនោះមានដូចតទៅនេះ៖

ក្បួនអនាមិក

ក្បួនអនាមិក (lambda) គឺជាក្បួនគ្មានឈ្មោះ មានតែដំណាងផ្សេង និងកន្សោមប្រមាណវិធីមួយតែប៉ុណ្ណោះ។ ក្បួនអនាមិកបញ្ជូនវត្ថុដែលជាលទ្ធផលបានមកពីកន្សោមប្រមាណវិធីនៅក្នុងនោះទៅកាន់កន្លែងណាដែលក្បួនអនាមិកត្រូវបានយកទៅប្រើ។ ដើម្បីបង្កើតក្បួនអនាមិកយើងត្រូវធ្វើដូចខាងក្រោមនេះ៖

```
ប្រាក់ចំណេញ = lambda ថ្លៃលក់, ថ្លៃទិញ : ថ្លៃលក់ - ថ្លៃទិញ
print(ប្រាក់ចំណេញ(1000, 900))
```

ប្រាក់ចំណេញ = lambda ថ្លៃលក់, ថ្លៃទិញ : ថ្លៃលក់ - ថ្លៃទិញ គឺជាការប្រើបញ្ជា lambda ដើម្បីបង្កើតក្បួនអនាមិកមួយដែលមាន ថ្លៃលក់ និង ថ្លៃទិញ ជាដំណាង និង ថ្លៃលក់ - ថ្លៃទិញ ជាកន្សោមប្រមាណវិធី។

នៅក្នុងកម្មវិធីខាងលើនេះ នៅពេលដែលបញ្ជា lambda ត្រូវបានយកទៅអនុវត្ត វត្ថុដែលជាក្បួនអត់ឈ្មោះមួយត្រូវបានបង្កើតឡើង។ ក្បួននោះមានឈ្មោះ ថ្លៃលក់ និង ថ្លៃទិញ ជាដំណាង និងកន្សោមប្រមាណវិធី ថ្លៃលក់ - ថ្លៃទិញ ជាតួក្បួន។ ក្បួននេះត្រូវហៅថាក្បួន

អនាមិកព្រោះវាជាក្បួនគ្មានឈ្មោះ។ ថ្វីត្បិតតែនៅក្នុងក្បួនអនាមិកគ្មានបញ្ជា return ក៏ពិតមែន តែក្បួនអនាមិកនឹងបញ្ជូនវត្ថុដែលជាលទ្ធផលបានមកពីកន្សោមប្រមាណវិធីដែលជាតួក្បួន ទៅកាន់កន្លែងណាដែលក្បួនអនាមិកត្រូវបានយកទៅប្រើ។ ម៉្យាងទៀត ដោយក្បួនអនាមិកជាក្បួនអត់ឈ្មោះ ដូចនេះយើងត្រូវតែយកឈ្មោះ ណាមួយទៅភ្ជាប់នឹងក្បួននោះ ដើម្បីអាចយកវាមកប្រើការបាន បើពុំនោះសោតទេ វានឹងត្រូវលុបចោលទៅវិញ ដោយយន្តការបោសសំអាត ។

ការបង្កើតក្បួនអនាមិកដូចខាងលើនេះ សមមូលនឹងការបង្កើតក្បួមធម្មតាដូចខាងក្រោមនេះ៖

```
def ប្រាក់ចំណេញ(ថ្លៃលក់, ថ្លៃទិញ) :
    return ថ្លៃលក់ - ថ្លៃទិញ

print(ប្រាក់ចំណេញ(1000, 900))
```

ផលប្រយោជន៍នៃការបង្កើតក្បួនអនាមិកគឺថា ក្បួនអនាមិកមានលក្ខណៈជាកន្សោមប្រមាណវិធី ដូចនេះនៅកន្លែងណាដែលកន្សោមប្រមាណវិធីអាចត្រូវបង្កើត ក្បួនអនាមិកក៏អាចត្រូវបង្កើតនៅទីនោះបានដែរ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
កម្រងថេរ = (100, True, lambda ថ្លៃលក់, ថ្លៃទិញ : ថ្លៃលក់ - ថ្លៃទិញ)
print(កម្រងថេរ)
```

កម្រងថេរ = (100, True, lambda ថ្លៃលក់, ថ្លៃទិញ : ថ្លៃលក់ - ថ្លៃទិញ) គឺជាការបង្កើតកម្រងថេរមានឈ្មោះថា កម្រងថេរ មួយដែលនៅក្នុងនោះមានក្បួនអនាមិកមួយជាធាតុមានលេខរៀង 2 ។

ថង្កៀនការ

វត្ថុនាំមុខ (iterator) គឺជាវត្ថុដែលជាលទ្ធផលបានមកពីការយកក្បួនមានស្រាប់ឈ្មោះ iter មកប្រើ ដោយផ្តល់ដំណឹងជាសមាសវត្ថុណាមួយឲ្យវា។ ដើម្បីបង្កើតវត្ថុនាំមុខ យើងត្រូវធ្វើដូចខាងក្រោមនេះ៖

```
កម្រងដើម = (100, "ប្រាក់ចំណេញ", True)
វត្ថុនាំមុខកម្រងដើម = iter(កម្រងដើម)
print(វត្ថុនាំមុខកម្រងដើម)
```

វត្ថុនាំមុខកម្រងដើម = iter(កម្រងដើម) គឺជាការយកក្បួនមានស្រាប់ឈ្មោះ iter មកប្រើដើម្បីបង្កើតវត្ថុនាំមុខឈ្មោះ វត្ថុនាំមុខកម្រងដើម មួយ។ ដំណឹងសម្រាប់ក្បួនឈ្មោះ iter គឺជាកម្រងថេរមានឈ្មោះថា កម្រងដើម ។

បើសិនជាយើងយកក្បួនមានស្រាប់ឈ្មោះ next មកប្រើដោយផ្តល់ដំណឹងជាវត្ថុនាំមុខណាមួយឲ្យទៅក្បួននោះ យើងនឹងបានលទ្ធផលដូចខាងក្រោមនេះ៖

```
កម្រងដើម = (100, "ប្រាក់ចំណេញ", True)
វត្ថុនាំមុខកម្រងដើម = iter(កម្រងដើម)
print(next(វត្ថុនាំមុខកម្រងដើម))
print(next(វត្ថុនាំមុខកម្រងដើម))
print(next(វត្ថុនាំមុខកម្រងដើម))
print(next(វត្ថុនាំមុខកម្រងដើម))
```

print(next(វត្ថុនាំមុខកម្រងដើម)) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកក្បួនមានស្រាប់ឈ្មោះ next មកប្រើជាលើកទីមួយ ដោយផ្តល់វត្ថុនាំមុខឈ្មោះ វត្ថុនាំមុខកម្រងដើម ជាដំណឹងឲ្យទៅក្បួននោះ។ ជាលទ្ធផល ធាតុមានលេខរៀង 0 នៅក្នុងកម្រងថេរឈ្មោះ កម្រងដើម ត្រូវបានយកមកពិនិត្យមើល។

print(next(*វត្ថុនៃកម្រងដើម*)) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកក្បួនមានស្រាប់ឈ្មោះ next មកប្រើជាលើកទីពីរ ដោយផ្តល់វត្ថុនៃករឈ្មោះ វត្ថុនៃកម្រងដើម ជាដំណឹងឲ្យទៅក្បួននោះ។ ជាលទ្ធផល ធាតុមានលេខរៀង 1 នៅក្នុងកម្រងថេរឈ្មោះ កម្រងដើម ត្រូវបានយកមកពិនិត្យមើល។

print(next(*វត្ថុនៃកម្រងដើម*)) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកក្បួនមានស្រាប់ឈ្មោះ next មកប្រើជាលើកទីបី ដោយផ្តល់វត្ថុនៃករឈ្មោះ វត្ថុនៃកម្រងដើម ជាដំណឹងឲ្យទៅក្បួននោះ។ ជាលទ្ធផល ធាតុមានលេខរៀង 2 នៅក្នុងកម្រងថេរឈ្មោះ កម្រងដើម ត្រូវបានយកមកពិនិត្យមើល។

print(next(*វត្ថុនៃកម្រងដើម*)) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកក្បួនមានស្រាប់ឈ្មោះ next មកប្រើជាលើកទីបួន ដោយផ្តល់វត្ថុនៃករឈ្មោះ វត្ថុនៃកម្រងដើម ជាដំណឹងឲ្យទៅក្បួននោះ។ ជាលទ្ធផល ភាពមិនប្រក្រតីប្រភេទ StopIteration ត្រូវបានបង្កើតឡើង។

ដូចនេះយើងឃើញថា គ្រប់ការយកក្បួនមានស្រាប់ឈ្មោះ next មកប្រើដោយផ្តល់ដំណឹងជាវត្ថុនៃករឈ្មោះ វត្ថុនៃកម្រងដើម ឲ្យទៅក្បួននោះ ធាតុណាមួយនៅក្នុងកម្រងថេរឈ្មោះ កម្រងដើម ត្រូវយកមកពិនិត្យ។ ហើយកម្រងថេនោះគឺវត្ថុដែលត្រូវបានផ្តល់ជាដំណឹងសម្រាប់ក្បួនមានស្រាប់ឈ្មោះ iter នៅពេលបង្កើតវត្ថុនៃករឈ្មោះ វត្ថុនៃកម្រងដើម នោះ។ ហើយការយកក្បួនមានស្រាប់ឈ្មោះ next មកប្រើលើកក្រោយៗទៀត ធ្វើឲ្យធាតុជាបន្តបន្ទាប់នៅក្នុងកម្រងថេរនោះ ត្រូវយកមកពិនិត្យជាហូរហែរ។ នៅពេលដែលធាតុនៅក្នុងកម្រងថេរនោះត្រូវបានពិនិត្យអស់ហើយ ការយកក្បួនមានស្រាប់ឈ្មោះ next មកប្រើជាបន្តទៀត ធ្វើឲ្យភាពមិនប្រក្រតីប្រភេទ StopIteration កើតមានឡើង។

ដូចនេះវឌ្ឍនករគឺជាវត្ថុម៉្យាងដែលតាមរយៈវា យើងអាចយកធាតុនៅក្នុងសមាសវត្ថុណាមួយមកពិនិត្យមើលមួយម្តងៗជាហូរហែររហូតដល់អស់ធាតុ។ បានន័យថា វឌ្ឍនករគឺជាវត្ថុម៉្យាងដែលមាននាទីជាស្ពានសម្រាប់ចម្លងពីធាតុមួយទៅធាតុមួយទៀតនៅក្នុងសមាសវត្ថុណាមួយ។

ការស្វែងយល់ដឹងពីវឌ្ឍនករមានសារៈសំខាន់ណាស់ក្នុងការយល់ដឹងពីយន្តការរបស់បញ្ជា for ។ ពីព្រោះនៅពេលដែលបញ្ជា for ត្រូវយកមកប្រើជាមួយនឹងសមាសវត្ថុណាមួយ ក្បួនមានស្រាប់ឈ្មោះ iter ត្រូវយកមកប្រើជាស្វ័យប្រវត្តិ ហើយសមាសវត្ថុនោះនឹងត្រូវផ្តល់ជាដំណឹងឲ្យទៅក្បួនមានស្រាប់ឈ្មោះ iter នោះ។ ប្រការនេះធ្វើឲ្យវឌ្ឍនករមួយត្រូវបង្កើតឡើង។ បន្ទាប់មកទៀតក្បួនមានស្រាប់ឈ្មោះ next ក៏ត្រូវយកមកប្រើជាស្វ័យប្រវត្តិដែរ ហើយវឌ្ឍនករដែលទើបត្រូវបានបង្កើតនោះ ក៏ត្រូវផ្តល់ជាដំណឹងឲ្យទៅក្បួនមានស្រាប់ឈ្មោះ next នោះដែរ។ ហើយការយកក្បួនមានស្រាប់ឈ្មោះ next មកប្រើជាស្វ័យប្រវត្តិ ត្រូវប្រព្រឹត្តទៅជាច្រើនលើកច្រើនសាររហូតដល់ធាតុនៅក្នុងសមាសវត្ថុដែលត្រូវបានផ្តល់ជាដំណឹងសម្រាប់ក្បួនមានស្រាប់ឈ្មោះ iter ត្រូវបានពិនិត្យអស់។ នៅពេលដែលធាតុនៅក្នុងសមាសវត្ថុត្រូវបានពិនិត្យអស់ ភាពមិនប្រក្រតីប្រភេទ StopIteration ត្រូវកើតមានឡើង។

សរុបមក ការយកបញ្ជា for មកប្រើ បណ្តាលឲ្យក្បួនមានស្រាប់ឈ្មោះ iter និង next ត្រូវយកមកប្រើជាស្វ័យប្រវត្តិជាបន្តបន្ទាប់ដើម្បីបង្កើតវឌ្ឍនករមួយនិងដើម្បីពិនិត្យមើលធាតុនៅក្នុងសមាសវត្ថុដែលត្រូវបានផ្តល់ជាដំណឹងសម្រាប់ក្បួនមានស្រាប់ឈ្មោះ iter ។ ហើយនៅពេលដែលការពិនិត្យមើលធាតុនៅក្នុងសមាសវត្ថុនោះត្រូវបានចប់សព្វគ្រប់ហើយ ភាពមិនប្រក្រតីប្រភេទ StopIteration ត្រូវកើតមានឡើង។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
កម្រងដើម = (100, "ប្រាក់ចំណេញ", True)
for វត្ថុ in កម្រងដើម :
```

```
print(វត្ថុ)
```

for វត្ថុ in កម្រងដើម : គឺជាការប្រើបញ្ជា for ដើម្បីពិនិត្យមើលគ្រប់ធាតុទាំងអស់នៅក្នុងកម្រងថេរឈ្មោះ កម្រងដើម ។ ប្រការនេះធ្វើឲ្យក្បួនមានស្រាប់ឈ្មោះ iter ត្រូវបានយកមកប្រើជាស្វ័យប្រវត្តិដោយទទួលបានកម្រងថេរឈ្មោះ កម្រងដើម ជាដំណឹងសម្រាប់បង្កើតវឌ្ឍនករមួយ ។ បន្ទាប់មកទៀត ក្បួនមានស្រាប់ឈ្មោះ next ក៏ត្រូវបានយកមកប្រើជាស្វ័យប្រវត្តិដែរ ហើយវឌ្ឍនករដែលទើបត្រូវបានបង្កើតនោះ ត្រូវបានផ្តល់ជាដំណឹងសម្រាប់ក្បួននោះ។ ការយកក្បួនមានស្រាប់ឈ្មោះ next មកប្រើត្រូវប្រព្រឹត្តទៅជាច្រើនលើកច្រើនសាររហូតដល់ធាតុនៅក្នុងកម្រងថេរឈ្មោះ កម្រងដើម ត្រូវបានពិនិត្យអស់។

ក្បួនផលិតករ និងផលិតករ

ក្បួនផលិតករ (generator function) គឺជាក្បួនទាំងឡាយណាដែលមានបញ្ជា yield នៅក្នុងនោះ។ ដើម្បីបង្កើតក្បួនផលិតករ យើងត្រូវធ្វើដូចខាងក្រោមនេះ៖

```
def ផលិតកម្ម(កម្រង) :
    print("ដំណាក់ការទីមួយ")
    yield កម្រង[0] + 10
    print("ដំណាក់ការទីពីរ")
    yield កម្រង[1] + 10
    print("ដំណាក់ការទីបី")
    yield កម្រង[2] + 10

ពុម្ព = ផលិតកម្ម([1, 2, 3])
print(ពុម្ព)
```

def *ផលិតកម្ម(កម្រង)* : គឺជាបញ្ជាតម្រូវឲ្យបង្កើតក្បួនផលិតករឈ្មោះ ផលិតកម្ម មួយដែលមានបញ្ជា yield ជាច្រើននៅក្នុងនោះ។

ពុម្ព = *ផលិតកម្ម*([1, 2, 3]) គឺជាការយកក្បួនផលិតករឈ្មោះ ផលិតកម្ម មកប្រើដែលនាំឲ្យវត្ថុឈ្មោះ ពុម្ព មួយត្រូវបានបង្កើតឡើង។

ដូចនេះយើងឃើញថា ការយកក្បួនផលិតករមកប្រើ មិនបណ្តាលឲ្យបញ្ជានៅក្នុងក្បួននោះត្រូវយកទៅអនុវត្តទេ ផ្ទុយទៅវិញ វត្ថុមួយហៅថា *ផលិតករ* (generator) ត្រូវបានបង្កើតឡើង។

បើយើងយកក្បួនមានស្រាប់ឈ្មោះ next មកប្រើដោយផ្តល់ដំណឹងជាផលិតករណាមួយឲ្យវាយើងនឹងបានលទ្ធផលដូចខាងក្រោមនេះ៖

```
def ផលិតកម្ម(កម្រង) :
    print("ដំណាក់ការទីមួយ")
    yield កម្រង[0] + 10
    print("ដំណាក់ការទីពីរ")
    yield កម្រង[1] + 10
    print("ដំណាក់ការទីបី")
    yield កម្រង[2] + 10

ពុម្ព = ផលិតកម្ម([1, 2, 3])
print(next(ពុម្ព))
print(next(ពុម្ព))
print(next(ពុម្ព))
print(next(ពុម្ព))
```

print(next(ពុម្ព)) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកក្បួនមានស្រាប់ឈ្មោះ next មកប្រើជាលើកទីមួយដោយផ្តល់ផលិតករឈ្មោះ ពុម្ព ជាដំណឹងឲ្យទៅក្បួននោះ។ ប្រការនេះបណ្តាលឲ្យក្រុមបញ្ជានៅក្នុងក្បួនផលិតករឈ្មោះ ផលិតកម្ម ត្រូវយកមកអនុវត្តជាបន្តបន្ទាប់ រហូតដល់បញ្ជា yield ដំបូងគេទើបផ្អាកមួយរយៈសិន។

print(next(ពុម្ព)) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកក្បួនមានស្រាប់ឈ្មោះ next មកប្រើជាលើកទីពីរដោយផ្តល់ផលិតករឈ្មោះ ពុម្ព ជាដំណឹងឲ្យទៅក្បួននោះ។ ប្រការនេះបណ្តាលឲ្យក្រុមបញ្ជានៅក្នុងក្បួនផលិតករឈ្មោះ ផលិតកម្ម ត្រូវយកមកអនុវត្តជាបន្តបន្ទាប់ រហូតដល់បញ្ជា yield ទីពីរទើបផ្អាកមួយរយៈទៀត។

print(next(ពុម្ព)) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកក្បួនមានស្រាប់ឈ្មោះ next មកប្រើជាលើកទីបីដោយផ្តល់ផលិតករឈ្មោះ ពុម្ព ជាដំណឹងឲ្យទៅក្បួននោះ។ ប្រការនេះបណ្តាលឲ្យក្រុមបញ្ជានៅក្នុងក្បួនផលិតករឈ្មោះ ផលិតកម្ម ត្រូវយកមកអនុវត្តជាបន្តបន្ទាប់ រហូតដល់បញ្ជា yield ទីបីទើបផ្អាកមួយរយៈទៀត។

print(next(ពុម្ព)) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកក្បួនមានស្រាប់ឈ្មោះ next មកប្រើជាលើកទីបួនដោយផ្តល់ផលិតករឈ្មោះ ពុម្ព ជាដំណឹងឲ្យទៅក្បួននោះ។ ប្រការនេះធ្វើឲ្យភាពមិនប្រក្រតីប្រភេទ StopIteration ត្រូវកើតឡើង ព្រោះក្រុមបញ្ជានៅក្នុងក្បួនផលិតករឈ្មោះ ផលិតកម្ម ត្រូវបានយកមកអនុវត្តបានចប់សព្វគ្រប់អស់ហើយ។

សរុបមក ក្បួនផលិតករគឺជាក្បួនដែលមានបញ្ជា yield នៅក្នុងនោះ ហើយការយកក្បួនផលិតករមកប្រើ មិនបណ្តាលឲ្យក្រុមបញ្ជានៅក្នុងនោះត្រូវយកទៅអនុវត្តទេ តែផ្ទុយទៅវិញ វត្ថុដែលជាផលិតករមួយត្រូវបង្កើតឡើង។ ដើម្បីឲ្យក្រុមបញ្ជានៅក្នុងក្បួនផលិតករត្រូវយកទៅអនុវត្ត យើងត្រូវយកក្បួនមានស្រាប់ឈ្មោះ next មកប្រើដោយផ្តល់ផលិតករជាដំណឹង

សម្រាប់ក្បួននេះ។ ក៏ប៉ុន្តែ ការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងក្បួនផលិតករត្រូវផ្អាកនៅកន្លែងដែលមានបញ្ជា yield ដំបូងគេ ហើយបើយើងចង់ឲ្យការអនុវត្តន៍ក្រុមបញ្ជានោះមានដំណើរការជាបន្តទៅទៀតរហូតដល់បញ្ជា yield បន្ទាប់មកទៀត យើងត្រូវយកក្បួនមានស្រាប់ឈ្មោះ next មកប្រើតាមរបៀបដដែលនេះជាថ្មីម្តងទៀត។ លុះដល់ការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងក្បួនផលិតករបានចប់សព្វគ្រប់ហើយ ការសាកល្បងយកក្បួនមានស្រាប់ឈ្មោះ next មកប្រើតាមរបៀបដដែលនេះទៀត នឹងបង្ករឲ្យមានភាពមិនប្រក្រតីប្រភេទ StopIteration កើតមានឡើង។

បញ្ជា yield មានមុខងារស្រដៀងនឹងបញ្ជា return ដែរ ពោលគឺវាជាបញ្ជាដែលតម្រូវឲ្យបញ្ជូនវត្ថុណាមួយទៅកាន់កន្លែងណាដែលផលិតករកើតចេញពីក្បួននេះត្រូវយកទៅប្រើជាដំណឹងសម្រាប់ក្បួនមានស្រាប់ឈ្មោះ next ។ ក៏ប៉ុន្តែ បញ្ជា yield ខុសពីបញ្ជា return នៅត្រង់ថា បញ្ជា yield មិនតម្រូវឲ្យបញ្ចប់ការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងក្បួនផលិតករទេ គឺបញ្ជា yield គ្រាន់តែតម្រូវឲ្យផ្អាកការអនុវត្តន៍ក្រុមបញ្ជានោះមួយរយៈតែប៉ុណ្ណោះ ហើយការអនុវត្តន៍ក្រុមបញ្ជានោះ នឹងត្រូវធ្វើឡើងជាបន្តទៅទៀតនៅពេលណាដែលផលិតករត្រូវយកទៅប្រើជាដំណឹងសម្រាប់ក្បួនមានស្រាប់ឈ្មោះ next ជាថ្មីម្តងទៀត។

ដូចនេះយើងឃើញថា ផលិតករគឺជាវត្ថុមួយដែលតាមរយៈវា បញ្ជានៅក្នុងក្បួនផលិតករត្រូវយកទៅអនុវត្ត។

បើសិនណាជាយើងយកបញ្ជា for មកប្រើជាមួយនឹងផលិតករណាមួយ ក្បួនមានស្រាប់ឈ្មោះ next នឹងត្រូវយកមកប្រើជាស្វ័យប្រវត្តិជាច្រើនលើកច្រើនសាររហូតដល់ក្រុមបញ្ជានៅក្នុងក្បួនផលិតករដែលបង្កើតផលិតករនោះ ត្រូវយកទៅអនុវត្តបានចប់សព្វគ្រប់ ដែលជាប្រការធ្វើឲ្យភាពមិនប្រក្រតីប្រភេទ StopIteration ត្រូវកើតមានឡើង។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
def ផលិតកម្ម(កម្រង៍) :
```

```
    print("ដំណាក់ការទីមួយ")
    yield កម្រង៍[0] + 10
    print("ដំណាក់ការទីពីរ")
    yield កម្រង៍[1] + 10
    print("ដំណាក់ការទីបី")
    yield កម្រង៍[2] + 10

for វត្ថុ in ផលិតកម្ម([1, 2, 3]) :
    print(វត្ថុ)
```

for វត្ថុ in ផលិតកម្ម([1, 2, 3]) : គឺជាការយកបញ្ជា for មកប្រើជាមួយនឹងផលិតករដែលជាលទ្ធផលបានមកពីការយកក្បួនផលិតករឈ្មោះ ផលិតកម្ម មកប្រើ។ ប្រការនេះ បណ្តាលឲ្យក្បួនមានស្រាប់ឈ្មោះ next ត្រូវយកមកប្រើជាស្វ័យប្រវត្តិ ហើយផលិតករដែលទើបត្រូវបានបង្កើតឡើង ត្រូវបានផ្តល់ជាដំណឹងសម្រាប់ក្បួនមានស្រាប់ឈ្មោះ next នោះ។ ការយកក្បួនចុងក្រោយនេះមកប្រើ ត្រូវធ្វើឡើងជាច្រើនលើកច្រើនសារហូតដល់ការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងក្បួនផលិតករឈ្មោះ ផលិតកម្ម ត្រូវចប់សព្វគ្រប់ ដែលជាប្រការធ្វើឲ្យភាពមិនប្រក្រតីប្រភេទ StopIteration ត្រូវបានបង្កើតឡើងនិងទទួលយក។

ដោយហេតុថា ក្បួនផលិតករមានសមត្ថភាពអាចបញ្ជូនវត្ថុជាច្រើនប្រភេទខុសៗគ្នាតាមរយៈបញ្ជា yield ដូចនេះ យើងអាចប្រើក្បួនផលិតករជាពុម្ពសម្រាប់បង្កើតវត្ថុជាច្រើនខុសៗគ្នាអាស្រ័យទៅតាមរូបមន្តដែលជាក្រុមបញ្ជានៅក្នុងក្បួនផលិតករនោះ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
def ផលិតកម្ម() :
    yield 1000
```

```
        yield "ថ្ងៃលក់"
        yield 900
        yield "ប្រាក់ចំណេញ"
        yield True

    for វត្ថុ in ផលិតកម្ម() :
        print(វត្ថុ)
```

for វត្ថុ in ផលិតកម្ម() : គឺជាការយកបញ្ជា for មកប្រើជាមួយនឹងផលិតករដែលជាលទ្ធផលបានមកពីការយកក្បួនផលិករឈ្មោះ ផលិតកម្ម មកប្រើ។ ប្រការនេះធ្វើឲ្យវត្ថុមួយចំនួនខុសៗគ្នាត្រូវបានបង្កើតឡើង អាស្រ័យទៅតាមរូបមន្តដែលជាក្រុមបញ្ជានៅក្នុងក្បួនផលិតករឈ្មោះ ផលិតកម្ម នោះ។

យើងឃើញថា ការយកផលិតករមកប្រើជាមួយនឹងបញ្ជា for បណ្តាលឲ្យវត្ថុជាច្រើនត្រូវបង្កើតឡើង ដូចនេះក្បួនផលិតករក៏មានលក្ខណៈដូចជាសមាសវត្ថុដទៃទៀតដែរ។ ពោលគឺវត្ថុដែលត្រូវបង្កើតឡើងទាំងប៉ុន្មាន ហាក់ដូចជាធាតុនៅក្នុងក្បួនផលិតករនោះដែរ។ ក៏ប៉ុន្តែ ក្បួនផលិតករខុសពីសមាសវត្ថុដទៃទៀតនៅត្រង់ថា ធាតុរបស់ក្បួនផលិតករ គឺជាវត្ថុដែលមិនទាន់ត្រូវបានបង្កើតនៅឡើយ ធាតុទាំងនោះត្រូវបង្កើតឡើងតែនៅពេលដែលផលិតករត្រូវយកទៅប្រើជាមួយនឹងបញ្ជា for ឬក្បួនមានស្រាប់ឈ្មោះ next តែប៉ុណ្ណោះ។ ដូចនេះក្បួនផលិតករមានលក្ខណៈជា *សមាសវត្ថុអរូបី* (iterable object) ដែលមានធាតុជាវត្ថុមិនទាន់ត្រូវបានបង្កើតនៅឡើយ ហើយធាតុទាំងនោះលេចចេញមានរូបរាងឡើងតែនៅពេលណាផលិតកររបស់វាត្រូវបានយកទៅប្រើជាមួយនឹងបញ្ជា for ឬក្បួនមានស្រាប់ឈ្មោះ next តែប៉ុណ្ណោះ។ ដូចនេះ ការបង្កើតក្បួនផលិតករ ត្រូវការសតិតិចជាងការបង្កើតសមាសវត្ថុធម្មតាផ្សេងៗទៀត ដែលអាចមានធាតុមានចំនួនស្មើគ្នា។ ពីព្រោះក្បួនផលិតករគឺជាសមាសវត្ថុអរូបីដែលគ្មានធាតុនៅ

ឡើយ។ ក៏ប៉ុន្តែផ្ទុយទៅវិញ ការយកក្បួនផលិតករមកប្រើជំនួសឲ្យសមាសវត្ថុផ្សេងៗទៀត ធ្វើឲ្យល្បឿនចុះយឺត ព្រោះក្បួនផលិតករត្រូវការរយៈដ៏ចាំបាច់ណាមួយដើម្បីបង្កើតកធាតុរបស់វា ។

បើសិនជាយើងយកថ្នាក់មានស្រាប់ឈ្មោះ dict ដែលជាថ្នាក់នៃគ្រប់វត្ថុទាំងឡាយណាដែលជាវចនានុក្រម មកពិនិត្យមើលម្តងទៀត យើងនឹងឃើញថា មានវិធីមួយចំនួននៅក្នុងថ្នាក់នោះផ្តល់ទូផលជាសមាសវត្ថុអរូបីម្យ៉ាងដែលមានលក្ខណៈដូចជាក្បួនផលិតករដែរ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
បង្ខិចប្រាក់ = {"ថ្លៃលក់":1000, "ថ្លៃទិញ":900, "សោហ៊ុយ":25, "ប្រាក់ចំណេញ":100}
ជាតុគូអរូបី = បង្ខិចប្រាក់.items()
កូនសោរអរូបី = បង្ខិចប្រាក់.keys()
តម្លៃអរូបី = បង្ខិចប្រាក់.values()
for វត្ថុ in ជាតុគូអរូបី :
   print(វត្ថុ)
for វត្ថុ in កូនសោរអរូបី :
   print(វត្ថុ)
for វត្ថុ in តម្លៃអរូបី :
   print(វត្ថុ)
```

ជាតុគូអរូបី = បង្ខិចប្រាក់.items() គឺជាការយកវិធីឈ្មោះ items នៅក្នុងថ្នាក់មានស្រាប់ឈ្មោះ dict មកប្រើដើម្បីបង្កើតសមាសវត្ថុអរូបីមួយមានឈ្មោះថា ធាតុគូអរូបី ដែលមានធាតុជាធាតុគូនៅក្នុងវចនានុក្រមឈ្មោះ បង្ខិចប្រាក់ ។

កូនសោរអរូបី = បង្វិចប្រាក់.keys() គឺជាការយកវិធីឈ្មោះ keys នៅក្នុងថ្នាក់មានស្រាប់ឈ្មោះ dict មកប្រើដើម្បីបង្កើតសមាសវត្ថុអរូបីមួយមានឈ្មោះថា កូនសោរអរូបី ដែលមានធាតុជាកូនសោរនៅក្នុងវចនានុក្រមឈ្មោះ បង្វិចប្រាក់ ។

តម្លៃអរូបី = បង្វិចប្រាក់.values() គឺជាការយកវិធីឈ្មោះ values នៅក្នុងថ្នាក់មានស្រាប់ឈ្មោះ dict មកប្រើដើម្បីបង្កើតសមាសវត្ថុអរូបីមួយមានឈ្មោះថា តម្លៃអរូបី ដែលមានធាតុជាតម្លៃនៅក្នុងវចនានុក្រមឈ្មោះ បង្វិចប្រាក់ ។

កម្រងអថេររូបមន្ត

កម្រងអថេររូបមន្ត (list comprehension) គឺជាកម្រងអថេរដែលមានទម្រង់ជារូបមន្តសម្រាប់បង្កើតកម្រងអថេរថ្មីមួយទៀត។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
កម្រងថ្មី = [វត្ថុ + 10 for វត្ថុ in (1, 2, 3)]
print(កម្រងថ្មី)
```

កម្រងថ្មី = [*វត្ថុ* + 10 for *វត្ថុ* in (1, 2, 3)] គឺជាបញ្ជាតម្រូវឲ្យបង្កើតកម្រងអថេរឈ្មោះ កម្រងថ្មីមួយដោយប្រើកម្រងអថេររូបមន្ត [វត្ថុ + 10 for វត្ថុ in (1, 2, 3)] ។

នៅក្នុងកម្រងអថេររូបមន្តខាងលើនេះ *វត្ថុ* + 10 for *វត្ថុ* in (1, 2, 3) គឺជារូបមន្តសម្រាប់បង្កើតកម្រងអថេរថ្មីផ្សេងៗទៀត។ កន្សោមប្រមាណវិធី *វត្ថុ* + 10 គឺជាកន្សោមប្រមាណវិធីដែលផ្តល់លទ្ធផលជាធាតុមួយសម្រាប់កម្រងអថេរថ្មី។ ហើយបញ្ជា for នៅក្នុងនោះ គឺជាបញ្ជាប្រើសម្រាប់ពិនិត្យមើលធាតុនៃកម្រងថេរ (1, 2, 3) ។ ហើយគ្រប់ការពិនិត្យមើលធាតុណាមួយនៅ

ក្នុងកម្រងថេរ (1, 2, 3) នាំឲ្យបានធាតុមួយសម្រាប់កម្រងអថេរថ្មី។ ធាតុនោះគឺជាលទ្ធផលបានមកពីកន្សោមប្រមាណវិធី *វត្ថុ* + 10 ។

កម្មវិធីខាងលើនេះអាចត្រូវសរសេរតាមរបៀបម៉្យាងទៀតដូចខាងក្រោមនេះ៖

```
កម្រង់ថ្មី = [ ]
for វត្ថុ in (1, 2, 3) :
    កម្រង់ថ្មី.append(វត្ថុ + 10)
print(កម្រង់ថ្មី)
```

នៅក្នុងកម្រងអថេររូបមន្ត ក្រៅពីកន្សោមប្រមាណវិធី យើងក៏អាចប្រើក្បួនផ្សេងៗទៀតបានដែរសម្រាប់បង្កើតកម្រងអថេរថ្មី។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
def ផលបូក(វត្ថុ) :
    return វត្ថុ + 10

កម្រង់ថ្មី = [ផលបូក(វត្ថុ) for វត្ថុ in (1, 2, 3)]
print(កម្រង់ថ្មី)
```

កម្រង់ថ្មី = [*ផលបូក*(*វត្ថុ*) for *វត្ថុ* in (1, 2, 3)] គឺជាការប្រើប្រាស់កម្រងអថេររូបមន្តមួយដើម្បីបង្កើតកម្រងអថេរឈ្មោះ កម្រងថ្មី មួយ។ នៅក្នុងកម្រងអថេររូបមន្តនោះ មានការយកក្បួនឈ្មោះ ផលបូក មកប្រើដើម្បីបង្កើតធាតុសម្រាប់កម្រងអថេរថ្មី។

លើសពីនេះទៀត យើងអាចប្រើបញ្ជា if មួយទៀតនៅក្នុងកម្រងអថេររូបមន្តដើម្បីធ្វើការជ្រើសរើសផ្សេងៗនៅក្នុងការបង្កើតធាតុសម្រាប់កម្រងអថេរថ្មី។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
def ផលបូក(វត្ថុ) :
    return វត្ថុ + 10

កម្រង៍ថ្មី = [ផលបូក(វត្ថុ) for វត្ថុ in (1, 2, 3) if វត្ថុ != 2]
print(កម្រង៍ថ្មី)
```

កម្រង៍ថ្មី = [ផលបូក(វត្ថុ) for វត្ថុ in (1, 2, 3) if វត្ថុ != 2] គឺជាបញ្ជាតម្រូវឲ្យបង្កើតកម្រងអថេឈ្មោះ កម្រងថ្មី មួយដោយប្រើកម្រងអថេររូបមន្តមួយដែលមានបញ្ជា if នៅក្នុងនោះសម្រាប់ធ្វើការជ្រើសរើសយកធាតុនៅក្នុងកម្រងថេរ (1, 2, 3) ។

មួយវិញទៀត សមាសវត្ថុដែលយើងយកមកប្រើជាមួយនឹងបញ្ជា for នៅក្នុងកម្រងអថេររូបមន្ត អាចជាសមាសវត្ថុប្រភេទណាក៏បានដែរ វាអាចជាសមាសវត្ថុអរូបីក៏បានដែរ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
def ផលបូក(វត្ថុ) :
    return វត្ថុ + 10

def ផលិតកម្ម() :
    for វត្ថុ in (1, 2, 3) :
        yield វត្ថុ

កម្រង៍ថ្មី = [ផលបូក(វត្ថុ) for វត្ថុ in ផលិតកម្ម() if វត្ថុ != 2]
print(កម្រង៍ថ្មី)
```

កម្រង៍ថ្មី = [ផលបូក(វត្ថុ) for វត្ថុ in ផលិតកម្ម() if វត្ថុ != 2] គឺជាបញ្ជាតម្រូវឲ្យបង្កើតកម្រងអថេរឈ្មោះ កម្រងថ្មី មួយដោយប្រើកម្រងអថេររូបមន្តមួយដែលនៅក្នុងនោះមានការ

ប្រើប្រាស់សមាសវត្ថុអរូបី ដែលលទ្ធផលបានមកពីការយកក្បួនផលិតករឈ្មោះ ផលិតកម្ម មកប្រើ។

កន្សោមផលិតករ

កន្សោមផលិតករ (generator expression) គឺជាកន្សោមមួយដែលផ្តល់លទ្ធផលជាវត្ថុ ដែលមានប្រភេទជាផលិតករ។ បញ្ជាទាំងឡាយនៅក្នុងកន្សោមផលិតករ គឺជារូបមន្តសម្រាប់ បង្កើតវត្ថុមួយចំនួន។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
ពុម្ព = (វត្ថុ + 10 for វត្ថុ in [1, 2, 3])
print(ពុម្ព)
```

ពុម្ព = (វត្ថុ + 10 for វត្ថុ in [1, 2, 3]) គឺជាបញ្ជាតម្រូវឲ្យបង្កើតវត្ថុមួយដែលជាផលិតករមាន ឈ្មោះថា ពុម្ព ដោយយកកន្សោមផលិតករ (វត្ថុ + 10 for វត្ថុ in [1, 2, 3]) មកប្រើ។

បើសិនណាជាយើងយកផលិតករដែលត្រូវបានបង្កើតឡើងដោយកន្សោមផលិតករ មកប្រើ ជាមួយនឹងបញ្ជា for យើងនឹងបានលទ្ធផលដូចខាងក្រោមនេះ៖

```
ពុម្ព = (វត្ថុ + 10 for វត្ថុ in [1, 2, 3])
for វត្ថុ in ពុម្ព :
    print(វត្ថុ)
```

for វត្ថុ in ពុម្ព : គឺជាការយកផលិតករឈ្មោះ ពុម្ព មកប្រើជាមួយនឹងបញ្ជា for ។ ប្រការនេះ បណ្តាលឲ្យវត្ថុមួយចំនួនដែលត្រូវចាត់ទុកថាជាធាតុរបស់កន្សោមផលិតករ ត្រូវបានបង្កើត ឡើង។

យើងឃើញថា ធាតុទាំងអស់ដែលត្រូវបានបង្កើតឡើងដោយការយកផលិតករមកប្រើជាមួយនឹងបញ្ជា for គឺត្រូវបានបង្កើតឡើងទៅតាមរូបមន្តដែលមាននៅក្នុងកន្សោមផលិតករដែលបានបង្កើតផលិតករនោះ។

សរុបមក យើងសង្កេតឃើញថា កន្សោមផលិតករនិងកម្រងអថេររូបមន្តមានទម្រង់ដូចគ្នាបេះបិទ។ ពោលគឺនៅក្នុងវត្ថុទាំងពីរនេះ សុទ្ធតែមានកន្សោមប្រមាណវិធីមួយនិងបញ្ជា for មួយដូចគ្នា។ ក៏ប៉ុន្តែ ភាពខុសគ្នារវាងកន្សោមផលិតករនិងកម្រងអថេររូបមន្តគឺស្ថិតនៅត្រង់ថា កម្រងអថេររូបមន្តផ្តល់លទ្ធផលជាកម្រងអថេរថ្មីមួយ ចំណែកឯកន្សោមផលិតករវិញ ផ្តល់លទ្ធផលជាផលិតករមួយដែលអាចត្រូវយកទៅប្រើជាមួយនឹងបញ្ជា for ដើម្បីបង្កើតវត្ថុផ្សេងៗទៀត។

ដូចនេះ បើសិនណាជាយើងចង់បង្កើតកម្រងអថេរថ្មីមួយ យើងត្រូវប្រើប្រាស់កម្រងអថេររូបមន្ត តែបើយើងចង់បង្កើតផលិតករណាមួយវិញ យើងត្រូវប្រើប្រាស់កន្សោមផលិតករ។

វិធីពិសេស

កន្លងមកយើងបានយកថ្នាក់មានស្រាប់មួយចំនួនមកពិនិត្យមើល ហើយយើងបានឃើញថានៅក្នុងថ្នាក់ទាំងនោះ មានវិធីមួយចំនួនដែលជាឈ្មោះមានសញ្ញា _ នេះពីរនៅអមសងខាង។ វិធីទាំងនោះគឺជា *វិធីពិសេស* (special method) ដែលត្រូវយកទៅប្រើជាស្វ័យប្រវត្តិនៅក្នុងកាលៈទេសៈដ៏ជាក់លាក់ណាមួយ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
ថ្លៃលក់ = 1000
ថ្លៃទិញ = 900
ប្រាក់ចំណេញ = ថ្លៃលក់ - ថ្លៃទិញ
print(ប្រាក់ចំណេញ)
```

ថ្លៃលក់ = 1000 គឺការបង្កើតចំនួនគត់លេខ 1000 មួយមានឈ្មោះថា ថ្លៃលក់ ដែលជាសិស្សនៃថ្នាក់មានស្រាប់ឈ្មោះ int ។ ប្រការនេះបណ្តាលឲ្យស្ថាបនិកដែលជាវិធីពិសេសឈ្មោះ __init__ នៅក្នុងថ្នាក់នោះត្រូវយកមកប្រើជាស្វ័យប្រវត្តិ។

ថ្លៃទិញ = 900 គឺការបង្កើតចំនួនគត់លេខ 900 មួយមានឈ្មោះថា ថ្លៃទិញ ដែលជាសិស្សនៃថ្នាក់មានស្រាប់ឈ្មោះ int ។ ប្រការនេះបណ្តាលឲ្យស្ថាបនិកដែលជាវិធីពិសេសឈ្មោះ __init__ នៅក្នុងថ្នាក់នោះត្រូវយកមកប្រើជាស្វ័យប្រវត្តិ។

ប្រាក់ចំណេញ = *ថ្លៃលក់* - *ថ្លៃទិញ* គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យធ្វើប្រមាណវិធីដករវាងវត្ថុឈ្មោះ ថ្លៃលក់ និងវត្ថុឈ្មោះ ថ្លៃទិញ ។ ប្រការនេះបណ្តាលឲ្យវិធីពិសេសឈ្មោះ __sub__ នៅក្នុងថ្នាក់មានស្រាប់ឈ្មោះ int ត្រូវយកមកប្រើជាស្វ័យប្រវត្តិ ហើយវត្ថុឈ្មោះ ថ្លៃលក់ និងវត្ថុឈ្មោះ ថ្លៃទិញ ត្រូវបានផ្តល់ជាដំណឹងរៀងគ្នាឲ្យទៅវិធីពិសេសនោះ។ លទ្ធផលបានមកពីការយកវិធីពិសេសនោះមកប្រើ គឺជាវត្ថុឈ្មោះ ប្រាក់ចំណេញ ដែលជាផលដករវាងវត្ថុឈ្មោះ ថ្លៃលក់ និងវត្ថុឈ្មោះ ថ្លៃទិញ ។

print(*ប្រាក់ចំណេញ*) គឺជាការយកក្បួនមានស្រាប់ឈ្មោះ print មកប្រើដើម្បីសរសេរវត្ថុឈ្មោះ ប្រាក់ចំណេញ នៅលើបង្អួចបឋម។ ប្រការនេះ បណ្តាលឲ្យវិធីពិសេសឈ្មោះ __str__ នៅក្នុងថ្នាក់មានស្រាប់ឈ្មោះ int ត្រូវយកមកប្រើជាស្វ័យប្រវត្តិ ហើយវត្ថុឈ្មោះ ប្រាក់ចំណេញ ត្រូវផ្តល់ជាដំណឹងសម្រាប់វិធីពិសេសនោះ។ ជាលទ្ធផល វត្ថុឈ្មោះ ប្រាក់ចំណេញ ត្រូវបានកែឲ្យទៅជាកម្រងអក្សរដែលត្រូវយកទៅសរសេរនៅលើបង្អួចបឋម។

យើងឃើញថា វិធីពិសេសទាំងប៉ុន្មានខាងលើនេះត្រូវយកទៅប្រើជាស្វ័យប្រវត្តិនៅក្នុងកាលៈទេសៈណាមួយដ៏ជាក់លាក់។ ក៏ប៉ុន្តែ លើសពីនេះទៀត យើងក៏អាចយកវិធីពិសេសទាំងនោះមកប្រើដោយផ្ទាល់បានដែរ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
ថ្លៃលក់ = 1000
ថ្លៃទិញ = 900
ប្រាក់ចំណេញ = int.__sub__(ថ្លៃលក់, ថ្លៃទិញ)
លទ្ធផល = ប្រាក់ចំណេញ.__str__()
print(លទ្ធផល)
```

ប្រាក់ចំណេញ = int.__sub__(ថ្លៃលក់, ថ្លៃទិញ) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកវិធីពិសេសឈ្មោះ __sub__ នៅក្នុងថ្នាក់មានស្រាប់ឈ្មោះ int មកប្រើដើម្បីធ្វើប្រមាណវិធីដករវាងចំនួនគត់ឈ្មោះ ថ្លៃលក់ និង ថ្លៃទិញ ។

លទ្ធផល = ប្រាក់ចំណេញ.__str__() គឺជាការយកវិធីពិសេសឈ្មោះ __str__ នៅក្នុងថ្នាក់មានស្រាប់ឈ្មោះ int មកប្រើដើម្បីកែចំនួនគត់ឈ្មោះ ប្រាក់ចំណេញ ឲ្យទៅជាកម្រងអក្សរ។

ដូចគ្នាដែរ បើសិនជាយើងបង្កើតថ្នាក់មួយដែលនៅក្នុងនោះមានវិធីពិសេសដូចខាងលើនេះ នៅពេលដែលយើងយកសិស្សនៃថ្នាក់ទាំងនោះទៅប្រើនៅក្នុងកាលៈទេសៈដូចខាងលើនេះ វិធីពិសេសទាំងនោះនឹងត្រូវយកទៅប្រើជាស្វ័យប្រវត្តិតាមរបៀបដូចខាងលើនេះដែរ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
class ទឹកប្រាក់() :
    def __init__(សិស្ស, ប្រាក់=0) :
        សិស្ស.ប្រាក់ = ប្រាក់
```

```
    def __sub__(សិស្សឆ្វេង, សិស្សស្តាំ) :
        លទ្ធផល = ទឹកប្រាក់()
        លទ្ធផល.ប្រាក់ = សិស្សឆ្វេង.ប្រាក់ - សិស្សស្តាំ.ប្រាក់
        return លទ្ធផល

    def __str__(សិស្ស) :
        return str(សិស្ស.ប្រាក់)

ថ្ងៃលក់ = ទឹកប្រាក់(1000)
ថ្ងៃទិញ = ទឹកប្រាក់(900)
ប្រាក់ចំណេញ = ថ្ងៃលក់ - ថ្ងៃទិញ
print(ប្រាក់ចំណេញ)
```

def __init__(សិស្ស, ប្រាក់=0) : គឺជាការបង្កើតស្ថាបនិកដែលជាវិធីពិសេសមានឈ្មោះថា __init__ ក្នុងគោលបំណងបង្កើតសម្បត្តិឈ្មោះ ប្រាក់ ទុកនៅក្នុងសិស្សដែលនឹងត្រូវបង្កើតឡើងនៅពេលដែលថ្នាក់ឈ្មោះ ទឹកប្រាក់ ត្រូវយកទៅប្រើ។

def __sub__(សិស្សឆ្វេង, សិស្សស្តាំ) : គឺជាការបង្កើតវិធីពិសេសឈ្មោះ __sub__ ដែលនឹងត្រូវយកទៅប្រើជាស្វ័យប្រវត្តិនៅពេលណាដែលមានការធ្វើប្រមាណវិធីដករវាងសិស្សនៃថ្នាក់ឈ្មោះ ទឹកប្រាក់ ។

def __str__(សិស្ស) : គឺជាការបង្កើតវិធីពិសេសឈ្មោះ __str__ ដែលនឹងត្រូវយកទៅប្រើជាស្វ័យប្រវត្តិនៅពេលដែលសិស្សនៃថ្នាក់ឈ្មោះ ទឹកប្រាក់ ត្រូវយកទៅប្រើជាដំណឹងសម្រាប់ក្បួនមានស្រាប់ឈ្មោះ print ។

ថ្លៃលក់ = ទឹកប្រាក់(1000) គឺជាការយកថ្នាក់ឈ្មោះ ទឹកប្រាក់ មកប្រើដើម្បីបង្កើតសិស្ស ឈ្មោះ ថ្លៃលក់ ម្នាក់។ ប្រការនេះ បណ្តាលឲ្យស្ថាបនិកនៃថ្នាក់ឈ្មោះ ទឹកប្រាក់ ត្រូវបានយក មកប្រើជាស្វ័យប្រវត្តិ ដែលធ្វើឲ្យសម្បត្តិឈ្មោះ ប្រាក់ ដែលជាលេខ 1000 មួយត្រូវបានបង្កើត ឡើងនិងទុកនៅក្នុងសិស្សឈ្មោះ ថ្លៃលក់ នោះ។

ថ្លៃទិញ = ទឹកប្រាក់(900) គឺជាការយកថ្នាក់ឈ្មោះ ទឹកប្រាក់ មកប្រើដើម្បីបង្កើតសិស្សឈ្មោះ ថ្លៃទិញ ម្នាក់ទៀត។ ប្រការនេះ បណ្តាលឲ្យស្ថាបនិកនៃថ្នាក់ឈ្មោះ ទឹកប្រាក់ ត្រូវបានយកមក ប្រើជាស្វ័យប្រវត្តិ ដែលធ្វើឲ្យសម្បត្តិឈ្មោះ ប្រាក់ ដែលជាលេខ 900 មួយត្រូវបានបង្កើតឡើង និងទុកនៅក្នុងសិស្សឈ្មោះ ថ្លៃទិញ នោះ។

ប្រាក់ចំណេញ = ថ្លៃលក់ - ថ្លៃទិញ គឺជាបញ្ហាដែលនៅក្នុងនោះមានការតម្រូវឲ្យធ្វើប្រមាណ វិធីដករវាងវត្ថុឈ្មោះ ថ្លៃលក់ និងវត្ថុឈ្មោះ ថ្លៃទិញ ដែលជាសិស្សនៃថ្នាក់ឈ្មោះ ទឹកប្រាក់ ។ ប្រការនេះបណ្តាលឲ្យវិធីឈ្មោះពិសេសឈ្មោះ __sub__ ត្រូវយកមកប្រើជាស្វ័យប្រវត្តិ ហើយ វត្ថុទាំងពីរនោះត្រូវបានផ្តល់ជាដំណឹងសម្រាប់វិធីពិសេសនោះរៀងគ្នា។

print(*ប្រាក់ចំណេញ*) គឺជាការយកក្បួនមានស្រាប់ឈ្មោះ print មកប្រើដោយផ្តល់វត្ថុឈ្មោះ ប្រាក់ចំណេញ ជាដំណឹងសម្រាប់ក្បួននោះ។ ប្រការនេះបណ្តាលឲ្យវិធីពិសេសឈ្មោះ __str__ នៃថ្នាក់ឈ្មោះ ទឹកប្រាក់ ត្រូវយកមកប្រើជាស្វ័យប្រវត្តិ ហើយកម្រងអក្សរដែលជាលទ្ធផល បានមកពីវិធីនេះ ត្រូវយកទៅសរសេរនៅលើបង្អួចបឋម។

ក្រៅពីវិធីពិសេសទាំងប៉ុន្មានខាងលើនេះ នៅមានវិធីពិសេសជាច្រើនទៀត ដែលយើងអាច បង្កើតឡើងសម្រាប់ប្រើនៅក្នុងកាលៈទេសៈផ្សេងៗទៀត។ វិធីពិសេសទាំងនោះមានដូចតទៅ នេះ៖

វិធីពិសេសសម្រាប់ប្រមាណវិធីនព្វន្ត

វិធីពិសេសសំខាន់ប្រើសម្រាប់ធ្វើប្រមាណវិធីនព្វន្តមាន៖

```
__add__(សិស្សឆ្វេង, សិស្សស្តាំ)
__sub__(សិស្សឆ្វេង, សិស្សស្តាំ)
__mul__(សិស្សឆ្វេង, សិស្សស្តាំ)
__truediv__(សិស្សឆ្វេង, សិស្សស្តាំ)
__floordiv__(សិស្សឆ្វេង, សិស្សស្តាំ)
__mod__(សិស្សឆ្វេង, សិស្សស្តាំ)
__pow__(សិស្សឆ្វេង, សិស្សស្តាំ)
```

ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
class ទឹកប្រាក់() :
  def __init__(សិស្ស, ប្រាក់=0) :
    សិស្ស.ប្រាក់ = ប្រាក់

  def __add__(សិស្សឆ្វេង, សិស្សស្តាំ) :
    លទ្ធផល = ទឹកប្រាក់()
    លទ្ធផល.ប្រាក់ = សិស្សឆ្វេង.ប្រាក់ + សិស្សស្តាំ.ប្រាក់
    return លទ្ធផល

  def __sub__(សិស្សឆ្វេង, សិស្សស្តាំ) :
    លទ្ធផល = ទឹកប្រាក់()
    លទ្ធផល.ប្រាក់ = សិស្សឆ្វេង.ប្រាក់ - សិស្សស្តាំ.ប្រាក់
    return លទ្ធផល
```

```
def __mul__(សិស្សឆ្វេង, សិស្សស្តាំ) :
    លទ្ធផល = ទឹកប្រាក់()
    លទ្ធផល.ប្រាក់ = សិស្សឆ្វេង.ប្រាក់ * សិស្សស្តាំ.ប្រាក់
    return លទ្ធផល

def __truediv__(សិស្សឆ្វេង, សិស្សស្តាំ) :
    លទ្ធផល = ទឹកប្រាក់()
    លទ្ធផល.ប្រាក់ = សិស្សឆ្វេង.ប្រាក់ / សិស្សស្តាំ.ប្រាក់
    return លទ្ធផល

def __floordiv__(សិស្សឆ្វេង, សិស្សស្តាំ) :
    លទ្ធផល = ទឹកប្រាក់()
    លទ្ធផល.ប្រាក់ = សិស្សឆ្វេង.ប្រាក់ // សិស្សស្តាំ.ប្រាក់
    return លទ្ធផល

def __mod__(សិស្សឆ្វេង, សិស្សស្តាំ) :
    លទ្ធផល = ទឹកប្រាក់()
    លទ្ធផល.ប្រាក់ = សិស្សឆ្វេង.ប្រាក់ % សិស្សស្តាំ.ប្រាក់
    return លទ្ធផល

def __pow__(សិស្សឆ្វេង, សិស្សស្តាំ) :
    លទ្ធផល = ទឹកប្រាក់()
    លទ្ធផល.ប្រាក់ = សិស្សឆ្វេង.ប្រាក់ ** សិស្សស្តាំ.ប្រាក់
    return លទ្ធផល

def __str__(សិស្ស) :
    return str(សិស្ស.ប្រាក់)
```

```
ថ្លៃលក់ = ទឹកប្រាក់(1000)
ថ្លៃទិញ = ទឹកប្រាក់(900)
print(ថ្លៃលក់ + ថ្លៃទិញ)
print(ថ្លៃលក់ - ថ្លៃទិញ)
print(ថ្លៃលក់ * ថ្លៃទិញ)
print(ថ្លៃលក់ / ថ្លៃទិញ)
print(ថ្លៃលក់ // ថ្លៃទិញ)
print(ថ្លៃលក់ % ថ្លៃទិញ)
print(ថ្លៃលក់ ** ថ្លៃទិញ)
```

print(ថ្លៃលក់ + ថ្លៃទិញ) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យធ្វើប្រមាណវិធីបូករវាងសិស្សនៃថ្នាក់ឈ្មោះ ទឹកប្រាក់ ។ ប្រការនេះធ្វើឲ្យវិធីពិសេសឈ្មោះ __add__ នៅក្នុងថ្នាក់នោះត្រូវយកមកប្រើដោយស្វ័យប្រវត្តិ។

print(ថ្លៃលក់ - ថ្លៃទិញ) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យធ្វើប្រមាណវិធីដករវាងសិស្សនៃថ្នាក់ឈ្មោះ ទឹកប្រាក់ ។ ប្រការនេះធ្វើឲ្យវិធីពិសេសឈ្មោះ __sub__ នៅក្នុងថ្នាក់នោះត្រូវយកមកប្រើដោយស្វ័យប្រវត្តិ។

print(ថ្លៃលក់ * ថ្លៃទិញ) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យធ្វើប្រមាណវិធីគុនរវាងសិស្សនៃថ្នាក់ឈ្មោះ ទឹកប្រាក់ ។ ប្រការនេះធ្វើឲ្យវិធីពិសេសឈ្មោះ __mul__ នៅក្នុងថ្នាក់នោះត្រូវយកមកប្រើដោយស្វ័យប្រវត្តិ។

print(*ថ្លៃលក់* / *ថ្លៃទិញ*) គឺជាបញ្ហាដែលនៅក្នុងនោះមានការតម្រូវឲ្យធ្វើប្រមាណវិធីចែករវាងសិស្សនៃថ្នាក់ឈ្មោះ ទឹកប្រាក់ ។ ប្រការនេះធ្វើឲ្យវិធីពិសេសឈ្មោះ __truediv__ នៅក្នុងថ្នាក់នោះត្រូវយកមកប្រើដោយស្វ័យប្រវត្តិ។

print(*ថ្លៃលក់* // *ថ្លៃទិញ*) គឺជាបញ្ហាដែលនៅក្នុងនោះមានការតម្រូវឲ្យធ្វើប្រមាណវិធីចែកបន្ថយរវាងសិស្សនៃថ្នាក់ឈ្មោះ ទឹកប្រាក់ ។ ប្រការនេះធ្វើឲ្យវិធីពិសេសឈ្មោះ __floordiv__ នៅក្នុងថ្នាក់នោះត្រូវយកមកប្រើដោយស្វ័យប្រវត្តិ។

print(*ថ្លៃលក់* % *ថ្លៃទិញ*) គឺជាបញ្ហាដែលនៅក្នុងនោះមានការតម្រូវឲ្យធ្វើប្រមាណវិធីចែកយកសំណល់រវាងសិស្សនៃថ្នាក់ឈ្មោះ ទឹកប្រាក់ ។ ប្រការនេះធ្វើឲ្យវិធីពិសេសឈ្មោះ __mod__ នៅក្នុងថ្នាក់នោះត្រូវយកមកប្រើដោយស្វ័យប្រវត្តិ។

print(*ថ្លៃលក់* ** *ថ្លៃទិញ*) គឺជាបញ្ហាដែលនៅក្នុងនោះមានការតម្រូវឲ្យធ្វើប្រមាណវិធីស្វ័យគុនរវាងសិស្សនៃថ្នាក់ឈ្មោះ ទឹកប្រាក់ ។ ប្រការនេះធ្វើឲ្យវិធីពិសេសឈ្មោះ __pow__ នៅក្នុងថ្នាក់នោះត្រូវយកមកប្រើដោយស្វ័យប្រវត្តិ។

ផ្ទុយមកវិញ បើនៅក្នុងថ្នាក់ណាមួយគ្មានវិធីពិសេសទាំងអស់នោះទេ រាល់ការសាកល្បងយកសិស្សនៃថ្នាក់នោះមកធ្វើប្រមាណវិធីនព្វន្ត នឹងបណ្តាលឲ្យភាពមិនប្រក្រតីកើតមានឡើង។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
class ទឹកប្រាក់() :
   def __init__(សិស្ស, ប្រាក់=0) :
      សិស្ស.ប្រាក់ = ប្រាក់

   def __str__(សិស្ស) :
```

```
        return str(សិស្ស.ប្រាក់)

ថ្លៃលក់ = ទឹកប្រាក់(1000)
ថ្លៃទិញ = ទឹកប្រាក់(900)
print(ថ្លៃលក់ + ថ្លៃទិញ)
```

print(ថ្លៃលក់ + ថ្លៃទិញ) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការសាកល្បងយកសិស្សនៃថ្នាក់ឈ្មោះ ទឹកប្រាក់ មកធ្វើប្រមាណវិធីបូក។ ដោយនៅក្នុងថ្នាក់នោះគ្មានវិធីពិសេសឈ្មោះ __add__ ដូច្នេះការសាកល្បងយកសិស្សនៃថ្នាក់នោះមកធ្វើប្រមាណវិធីបូកបង្កឲ្យមានភាពមិនប្រក្រតីកើតឡើង។

វិធីពិសេសសម្រាប់ប្រមាណវិធីប្រៀបធៀប

វិធីពិសេសប្រើសម្រាប់ធ្វើប្រមាណវិធីប្រៀបធៀបមាន៖

```
__lt__(សិស្សឆ្វេង, សិស្សស្តាំ)
__le__(សិស្សឆ្វេង, សិស្សស្តាំ)
__eq__(សិស្សឆ្វេង, សិស្សស្តាំ)
__ne__(សិស្សឆ្វេង, សិស្សស្តាំ)
__gt__(សិស្សឆ្វេង, សិស្សស្តាំ)
__ge__(សិស្សឆ្វេង, សិស្សស្តាំ)
```

ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
class ទឹកប្រាក់() :
    def __init__(សិស្ស, ប្រាក់=0) :
        សិស្ស.ប្រាក់ = ប្រាក់
```

```
    def __lt__(សិស្សឆ្វេង, សិស្សស្តាំ) :
        return (សិស្សឆ្វេង.ប្រាក់ < សិស្សស្តាំ.ប្រាក់)

    def __le__(សិស្សឆ្វេង, សិស្សស្តាំ) :
        return (សិស្សឆ្វេង.ប្រាក់ > សិស្សស្តាំ.ប្រាក់)

    def __eq__(សិស្សឆ្វេង, សិស្សស្តាំ) :
        return (សិស្សឆ្វេង.ប្រាក់ == សិស្សស្តាំ.ប្រាក់)

    def __ne__(សិស្សឆ្វេង, សិស្សស្តាំ) :
        return (សិស្សឆ្វេង.ប្រាក់ != សិស្សស្តាំ.ប្រាក់)

    def __gt__(សិស្សឆ្វេង, សិស្សស្តាំ) :
        return (សិស្សឆ្វេង.ប្រាក់ > សិស្សស្តាំ.ប្រាក់)

    def __ge__(សិស្សឆ្វេង, សិស្សស្តាំ) :
        return (សិស្សឆ្វេង.ប្រាក់ >= សិស្សស្តាំ.ប្រាក់)

    def __str__(សិស្ស) :
        return str(សិស្ស.ប្រាក់)

ថ្លៃលក់ = ទឹកប្រាក់(1000)
ថ្លៃទិញ = ទឹកប្រាក់(900)
print(ថ្លៃលក់ > ថ្លៃទិញ)
print(ថ្លៃលក់ < ថ្លៃទិញ)
print(ថ្លៃលក់ == ថ្លៃទិញ)
print(ថ្លៃលក់ != ថ្លៃទិញ)
print(ថ្លៃលក់ >= ថ្លៃទិញ)
```

print(ថ្ងៃលក់ <= ថ្ងៃទិញ)

print(ថ្ងៃលក់ > ថ្ងៃទិញ) គឺជាបញ្ហាដែលនៅក្នុងនោះមានការតម្រូវឲ្យធ្វើប្រមាណវិធីប្រៀបធៀបដោយប្រើប្រមាណសញ្ញា > រវាងសិស្សនៃថ្នាក់ឈ្មោះ ទឹកប្រាក់ ។ ប្រការនេះធ្វើឲ្យវិធីពិសេសឈ្មោះ __gt__ នៅក្នុងថ្នាក់ឈ្មោះ ទឹកប្រាក់ ត្រូវយកមកប្រើជាស្វ័យប្រវត្តិ។

print(ថ្ងៃលក់ < ថ្ងៃទិញ) គឺជាបញ្ហាដែលនៅក្នុងនោះមានការតម្រូវឲ្យធ្វើប្រមាណវិធីប្រៀបធៀបដោយប្រើប្រមាណសញ្ញា < រវាងសិស្សនៃថ្នាក់ឈ្មោះ ទឹកប្រាក់ ។ ប្រការនេះធ្វើឲ្យវិធីពិសេសឈ្មោះ __lt__ នៅក្នុងថ្នាក់ឈ្មោះ ទឹកប្រាក់ ត្រូវយកមកប្រើជាស្វ័យប្រវត្តិ។

print(ថ្ងៃលក់ == ថ្ងៃទិញ) គឺជាបញ្ហាដែលនៅក្នុងនោះមានការតម្រូវឲ្យធ្វើប្រមាណវិធីប្រៀបធៀបដោយប្រើប្រមាណសញ្ញា == រវាងសិស្សនៃថ្នាក់ឈ្មោះ ទឹកប្រាក់ ។ ប្រការនេះធ្វើឲ្យវិធីពិសេសឈ្មោះ __eq__ នៅក្នុងថ្នាក់ឈ្មោះ ទឹកប្រាក់ ត្រូវយកមកប្រើជាស្វ័យប្រវត្តិ។

print(ថ្ងៃលក់ != ថ្ងៃទិញ) គឺជាបញ្ហាដែលនៅក្នុងនោះមានការតម្រូវឲ្យធ្វើប្រមាណវិធីប្រៀបធៀបដោយប្រើប្រមាណសញ្ញា != រវាងសិស្សនៃថ្នាក់ឈ្មោះ ទឹកប្រាក់ ។ ប្រការនេះធ្វើឲ្យវិធីពិសេសឈ្មោះ __ne__ នៅក្នុងថ្នាក់ឈ្មោះ ទឹកប្រាក់ ត្រូវយកមកប្រើជាស្វ័យប្រវត្តិ។

print(ថ្ងៃលក់ >= ថ្ងៃទិញ) គឺជាបញ្ហាដែលនៅក្នុងនោះមានការតម្រូវឲ្យធ្វើប្រមាណវិធីប្រៀបធៀបដោយប្រើប្រមាណសញ្ញា >= រវាងសិស្សនៃថ្នាក់ឈ្មោះ ទឹកប្រាក់ ។ ប្រការនេះធ្វើឲ្យវិធីពិសេសឈ្មោះ __ge__ នៅក្នុងថ្នាក់ឈ្មោះ ទឹកប្រាក់ ត្រូវយកមកប្រើជាស្វ័យប្រវត្តិ។

print(ថ្ងៃលក់ <= ថ្ងៃទិញ) គឺជាបញ្ហាដែលនៅក្នុងនោះមានការតម្រូវឲ្យធ្វើប្រមាណវិធីប្រៀបធៀបដោយប្រើប្រមាណសញ្ញា <= រវាងសិស្សនៃថ្នាក់ឈ្មោះ ទឹកប្រាក់ ។ ប្រការនេះធ្វើឲ្យវិធីពិសេសឈ្មោះ __le__ នៅក្នុងថ្នាក់ឈ្មោះ ទឹកប្រាក់ ត្រូវយកមកប្រើជាស្វ័យប្រវត្តិ។

វិធីពិសេសសម្រាប់សមាសវត្ថុ

វិធីពិសេសសំខាន់ៗសម្រាប់សមាសវត្ថុមាន៖

```
__len__(សិស្ស)
__getitem__(សិស្ស, លេខរៀងឬកូនសោរ)
__setitem__(សិស្ស, លេខរៀងឬកូនសោរ, ធាតុឬតម្លៃ)
__delitem__(សិស្ស, លេខរៀងឬកូនសោរ)
__iter__(សិស្ស)
__contain__(សិស្ស, វត្ថុ)
```

ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
បង្ហិចប្រាក់ = {"ថ្លៃលក់":1000, "ថ្លៃទិញ":900, "សោហ៊ុយ":25}

class សំណុំប្រាក់() :
  def __init__(សិស្ស, ប្រាក់=0) :
    សិស្ស.ប្រាក់ = ប្រាក់

  def __len__(សិស្ស) :
    return len(សិស្ស.ប្រាក់)

  def __getitem__(សិស្ស, លេខរៀងឬកូនសោរ) :
    return សិស្ស.ប្រាក់[លេខរៀងឬកូនសោរ]

  def __setitem__(សិស្ស, លេខរៀងឬកូនសោរ, ធាតុឬតម្លៃ) :
    សិស្ស.ប្រាក់[លេខរៀងឬកូនសោរ] = ធាតុឬតម្លៃ

  def __delitem__(សិស្ស, លេខរៀងឬកូនសោរ) :
```

```
        del សិស្ស.ប្រាក់[លេខរៀងឬកូនសោរ]

    def __iter__(សិស្ស) :
        return iter(សិស្ស.ប្រាក់)

    def __contains__(សិស្ស, វត្ថុ) :
        return (វត្ថុ in សិស្ស.ប្រាក់)

    def __str__(សិស្ស) :
        return str(សិស្ស.ប្រាក់)

សំណុំប្រាក់រកស៊ី = សំណុំប្រាក់(បង្ខិចប្រាក់)
print(len(សំណុំប្រាក់រកស៊ី))
print(សំណុំប្រាក់រកស៊ី["ថ្ងៃលក់"])
សំណុំប្រាក់រកស៊ី["ថ្ងៃលក់"] = 5000
print(សំណុំប្រាក់រកស៊ី)
del សំណុំប្រាក់រកស៊ី["សោហ៊ុយ"]
print(សំណុំប្រាក់រកស៊ី)
for វត្ថុ in សំណុំប្រាក់រកស៊ី :
    print(វត្ថុ)
print("សោហ៊ុយ" in សំណុំប្រាក់រកស៊ី)
```

print(len(សំណុំប្រាក់រកស៊ី)) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកក្បួនមានស្រាប់ឈ្មោះ len មកប្រើដោយផ្តល់វត្ថុឈ្មោះ សំណុំប្រាក់រកស៊ី ដែលជាសិស្សនៃថ្នាក់ឈ្មោះ សំណុំប្រាក់ ទៅក្បួននោះ។ ប្រការនេះធ្វើឲ្យវិធីពិសេសឈ្មោះ __len__ នៅក្នុងថ្នាក់ឈ្មោះ សំណុំប្រាក់ នោះត្រូវយកមកប្រើជាស្វ័យប្រវត្តិ។

print(*សំណុំប្រាក់រកស៊ី*["*ថ្លៃលក់*"]) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យធ្វើប្រមាណវិធីលេខរៀងជាមួយនឹងសិស្សឈ្មោះ សំណុំប្រាក់រកស៊ី ។ ប្រការនេះធ្វើឲ្យវិធីពិសេសឈ្មោះ __getitem__ នៅក្នុងថ្នាក់ឈ្មោះ សំណុំប្រាក់ ត្រូវយកមកប្រើដោយស្វ័យប្រវត្តិ។

សំណុំប្រាក់រកស៊ី["*ថ្លៃលក់*"] = 5000 គឺជាបញ្ជាតម្រូវឲ្យធ្វើប្រមាណវិធីលេខរៀងជាមួយនឹងសិស្សឈ្មោះ សំណុំប្រាក់រកស៊ី ដើម្បីផ្លាស់ប្តូរតម្លៃជាប់នឹងកូនសោរ ថ្លៃលក់ ។ ប្រការនេះធ្វើឲ្យវិធីពិសេសឈ្មោះ __setitem__ នៅក្នុងថ្នាក់ឈ្មោះ សំណុំប្រាក់ ត្រូវបានយកមកប្រើដោយស្វ័យប្រវត្តិ។

del *សំណុំប្រាក់រកស៊ី*["*សោហ៊ុយ*"] គឺជាការប្រើបញ្ជា del ជាមួយនឹងសិស្សឈ្មោះ សំណុំប្រាក់រកស៊ី ដើម្បីលុបតម្លៃជាប់នឹងកូនសោរ សោហ៊ុយ ។ ប្រការនេះធ្វើឲ្យវិធីពិសេសឈ្មោះ __delitem__ នៅក្នុងថ្នាក់ឈ្មោះ សំណុំប្រាក់ ត្រូវយកមកប្រើដោយស្វ័យប្រវត្តិ។

for *វត្ថុ* in *សំណុំប្រាក់រកស៊ី* : គឺជាការប្រើបញ្ជា for ជាមួយនឹងសិស្សឈ្មោះ សំណុំប្រាក់រកស៊ី ។ ប្រការនេះធ្វើឲ្យវិធីពិសេសឈ្មោះ __iter__ នៅក្នុងថ្នាក់ឈ្មោះ សំណុំប្រាក់ ត្រូវយកមកប្រើដោយស្វ័យប្រវត្តិ។

print("*សោហ៊ុយ*" in *សំណុំប្រាក់រកស៊ី*) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការធ្វើប្រមាណវិធីរកធាតុដោយប្រើប្រមាណសញ្ញា in ជាមួយនឹងសិស្សឈ្មោះ សំណុំប្រាក់រកស៊ី ។ ប្រការនេះធ្វើឲ្យវិធីពិសេសឈ្មោះ __contains__ នៅក្នុងថ្នាក់ឈ្មោះ សំណុំប្រាក់ ត្រូវយកមកប្រើដោយស្វ័យប្រវត្តិ។

វិធីពិសេសសម្រាប់សិស្ស

វិធីពិសេសសំខាន់ៗសម្រាប់សិស្សមានដូចខាងក្រោមនេះ៖

```
__call__(សិស្ស)
__getattribute__(សិស្ស, សម្បត្តិ)
__setattr__(សិស្ស, សម្បត្តិ, វត្ថុ)
__delattr__(សិស្ស, សម្បត្តិ)
__dir__(សិស្ស)
__del__(សិស្ស)
```

ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
class ទឹកប្រាក់() :
    def __init__(សិស្ស, ប្រាក់=0) :
        សិស្ស.ប្រាក់ = ប្រាក់

    def __call__(សិស្ស) :
        print("សិស្សនៃថ្នាក់ឈ្មោះ ទឹកប្រាក់ ត្រូវបានយកទៅប្រើដូចជាក្បួនៗ")

    def __getattribute__(សិស្ស, សម្បត្តិ) :
        print("ការប៉ុនប៉ងយកសម្បត្តិឈ្មោះ", សម្បត្តិ, "ទៅប្រើ")

    def __setattr__(សិស្ស, សម្បត្តិ, វត្ថុ) :
        print("ការប៉ុនប៉ងបង្កើតសម្បត្តិឈ្មោះ", សម្បត្តិ, "ដែលជា", វត្ថុ)

    def __delattr__(សិស្ស, សម្បត្តិ) :
        print("ការប៉ុនប៉ងលុបសម្បត្តិឈ្មោះ", សម្បត្តិ)

    def __dir__(សិស្ស) :
        return [1, 2, 3]
```

```
    def __del__(សិស្ស) :
        print("សិស្សនៃថ្នាក់ឈ្មោះ ទឹកប្រាក់ ត្រូវបានលុបចោល។")

ថ្លៃលក់ = ទឹកប្រាក់(1000)
ថ្លៃលក់()
ថ្លៃលក់.ប្រាក់
del ថ្លៃលក់.ប្រាក់
print(dir(ថ្លៃលក់))
ថ្លៃលក់ = 2000
```

ថ្លៃលក់ = ទឹកប្រាក់(1000) គឺជាការបង្កើតសិស្សឈ្មោះ ថ្លៃលក់ ដោយយកថ្នាក់ឈ្មោះ ទឹកប្រាក់ មកប្រើ។ ប្រការនេះធ្វើឲ្យស្ថាបនិកនៃថ្នាក់ឈ្មោះ ទឹកប្រាក់ ត្រូវយកមកប្រើ ដែលជាកត្តាតម្រូវឲ្យអនុវត្តបញ្ជា សិស្ស.ប្រាក់ = ប្រាក់ ដែលជាបញ្ជាតម្រូវឲ្យបង្កើតសម្បត្តិឈ្មោះ ប្រាក់ ទុកនៅក្នុងសិស្សឈ្មោះ ថ្លៃលក់ នោះ។ ដោយនៅក្នុងថ្នាក់ឈ្មោះ ទឹកប្រាក់ មានវិធីពិសេសឈ្មោះ __setattr__ ដូចនេះគ្រប់ការបង្កើតសម្បត្តិសិស្សតម្រូវឲ្យវិធីពិសេសនេះត្រូវយកមកប្រើជាស្វ័យប្រវត្តិ។ បានន័យថា ការប្រើបញ្ជា សិស្ស.ប្រាក់ = ប្រាក់ ធ្វើឲ្យវិធីពិសេសឈ្មោះ __setattr__ ត្រូវយកមកប្រើជាស្វ័យប្រវត្តិ។

ថ្លៃលក់() គឺជាបញ្ជាតម្រូវឲ្យយកសិស្សឈ្មោះ ថ្លៃលក់ មកប្រើដូចជាក្បួនឬវិធីផ្សេងៗ។ ប្រការនេះធ្វើឲ្យវិធីពិសេសឈ្មោះ __call__ នៅក្នុងថ្នាក់ឈ្មោះ ទឹកប្រាក់ ត្រូវយកមកប្រើជាស្វ័យប្រវត្តិ។

ថ្លៃលក់.ប្រាក់ គឺជាបញ្ជាតម្រូវឲ្យយកសម្បត្តិឈ្មោះ ប្រាក់ មកប្រើតាមរយៈសិស្សឈ្មោះ ថ្លៃលក់ ។ ប្រការនេះធ្វើឲ្យវិធីពិសេសឈ្មោះ __getattribute__ នៅក្នុងថ្នាក់ឈ្មោះ ទឹកប្រាក់ ត្រូវបានយកមកប្រើដោយស្វ័យប្រវត្តិ។

del *ថ្លៃលក់.ប្រាក់* គឺជាការប្រើបញ្ជា del ដើម្បីលុបសម្បត្តិឈ្មោះ ប្រាក់ ។ ប្រការនេះធ្វើឲ្យវិធីពិសេសឈ្មោះ __delattr__ នៅក្នុងថ្នាក់ឈ្មោះ ទឹកប្រាក់ ត្រូវបានយកមកប្រើដោយស្វ័យប្រវត្តិ។

print(dir(*ថ្លៃលក់*)) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកក្បួនមានស្រាប់ឈ្មោះ dir មកប្រើដោយផ្តល់វត្ថុឈ្មោះ ថ្លៃលក់ ដែលជាសិស្សនៃថ្នាក់ឈ្មោះ ទឹកប្រាក់ ឲ្យទៅក្បួននោះ។ ប្រការនេះធ្វើឲ្យវិធីពិសេសឈ្មោះ __dir__ នៅក្នុងថ្នាក់ឈ្មោះ ទឹកប្រាក់ ត្រូវយកមកប្រើជាស្វ័យប្រវត្តិ។

ថ្លៃលក់ = 2000 គឺជាការយកឈ្មោះ ថ្លៃលក់ ទៅភ្ជាប់ជាមួយនឹងវត្ថុថ្មីដែលជាលេខ 2000 វិញម្តង។ កត្តានេះធ្វើឲ្យវត្ថុជាប់នឹងឈ្មោះ ថ្លៃលក់ ចាស់ត្រូវបានលុបចោលព្រោះវាត្រូវបានបាត់ឈ្មោះ។ ដោយវត្ថុចាស់នោះជាសិស្សនៃថ្នាក់ឈ្មោះ ទឹកប្រាក់ ដូចនេះនៅពេលដែលសិស្សនោះត្រូវលុបចោល វិធីពិសេសឈ្មោះ __del__ ត្រូវបានយកមកប្រើជាស្វ័យប្រវត្តិ។

សម្បត្តិពិសេស

បើយើងយកថ្នាក់និងឬសិស្សណាមួយមកពិនិត្យមើលដោយប្រើក្បួនមានស្រាប់ឈ្មោះ help យើងនឹងឃើញថានៅក្នុងវត្ថុទាំងនោះមានវត្ថុមួយចំនួនដែលជាសម្បត្តិអាចត្រូវយកមកប្រើតាមរយៈសិស្សឬថ្នាក់នោះបាន។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
class ទឹកប្រាក់() :
    def __init__(សិស្ស, ប្រាក់=0) :
        សិស្ស.ប្រាក់ = ប្រាក់

ថ្លៃលក់ = ទឹកប្រាក់(1000)
```

```
help(ទឹកប្រាក់)
```

help(ទឹកប្រាក់) គឺជាការយកក្បួនមានស្រាប់ឈ្មោះ help មកប្រើដើម្បីពិនិត្យមើលសម្បត្តិទាំងឡាយដែលអាចត្រូវយកមកប្រើតាមរយៈថ្នាក់ឈ្មោះ ទឹកប្រាក់ ។ នៅក្នុងចំណោមសម្បត្តិទាំងនោះ ក្រៅពីស្ថាបនិកដែលយើងបានបង្កើតឡើងដោយខ្លួនយើងផ្ទាល់ នៅមានវត្ថុមួយចំនួនទៀតដែលត្រូវបានបង្កើតឡើងជាស្វ័យប្រវត្តិ។ វត្ថុទាំងនោះគឺជា *សម្បត្តិពិសេស* (special attribute) ព្រោះវាមានសញ្ញា _ នេះពីរនៅអមសងខាង។

សម្បត្តិពិសេសឈ្មោះ __dict__

សម្បត្តិពិសេសឈ្មោះ __dict__ គឺជាវត្ថុដែលជាវចនានុក្រមមួយដែលនៅក្នុងនោះមានកូនសោរជាឈ្មោះរបស់សម្បត្តិផ្សេងៗ និងតម្លៃគឺជាវត្ថុដែលជាសម្បត្តិទាំងនោះផ្ទាល់តែម្តង។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
class ទឹកប្រាក់() :
    def __init__(សិស្ស, ប្រាក់=0) :
        សិស្ស.ប្រាក់ = ប្រាក់

ថ្លៃលក់ = ទឹកប្រាក់(1000)
print(ទឹកប្រាក់.__dict__)
print(ថ្លៃលក់.__dict__)
```

print(ទឹកប្រាក់.__dict__) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកសម្បត្តិពិសេសឈ្មោះ __dict__ នៅក្នុងថ្នាក់ឈ្មោះ ទឹកប្រាក់ មកសរសេរនៅលើបង្អួចបឋម។ យើងឃើញថាវចនានុក្រមដែលជាសម្បត្តិពិសេសឈ្មោះ __dict__ នោះមានកូនសោរជាឈ្មោះនៃសម្បត្តិផ្សេងៗនៅក្នុងថ្នាក់នោះ និងតម្លៃជាសម្បត្តិទាំងនោះផ្ទាល់តែម្តង។

print(*ថ្ងៃលក់*.__dict__) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកសម្បត្តិពិសេសឈ្មោះ __dict__ នៅក្នុងសិស្សឈ្មោះ ទឹកប្រាក់ មកសរសេរនៅលើបង្អួចបឋម។ យើងឃើញថា វចនានុក្រមដែលជាសម្បត្តិពិសេសឈ្មោះ __dict__ នោះមានកូនសោរជាឈ្មោះនៃសម្បត្តិផ្សេងៗរបស់សិស្សនោះ និងតម្លៃជាសម្បត្តិទាំងនោះផ្ទាល់តែម្តង។

ដូចនេះយើងអាចធ្វើការសន្និដ្ឋានបានថា សម្បត្តិពិសេសឈ្មោះ __dict__ គឺជាដែនកំណត់របស់ថ្នាក់ឬសិស្សណាមួយ ព្រោះវាជាវចនានុក្រមដែលមានធាតុគូជាសម្បត្តិរបស់វត្ថុទាំងនោះ។

ដោយហេតុថាសម្បត្តិពិសេសឈ្មោះ __dict__ គឺវចនានុក្រមដែលនៅក្នុងនោះមានសម្បត្តិរបស់សិស្សឬថ្នាក់ណាមួយ ដូចនេះយើងអាចយកសម្បត្តិទាំងនោះមកប្រើតាមរយៈ វចនានុក្រមឈ្មោះ __dict__ នេះបានដោយគ្មានបញ្ហាអ្វីឡើយ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
class ទឹកប្រាក់() :
    ប្រាក់ = 2000
    def __init__(សិស្ស, ប្រាក់=0) :
        សិស្ស.ប្រាក់ = ប្រាក់

ថ្ងៃលក់ = ទឹកប្រាក់(1000)
print(ទឹកប្រាក់.__dict__["ប្រាក់"])
print(ថ្ងៃលក់.__dict__["ប្រាក់"])
```

print(*ទឹកប្រាក់*.__dict__["*ប្រាក់*"]) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកសម្បត្តិឈ្មោះ ប្រាក់ នៅក្នុងថ្នាក់ឈ្មោះ ទឹកប្រាក់ មកប្រើការ។ ការយកសម្បត្តិនេះមកប្រើត្រូវបានធ្វើឡើងតាមរយៈវចនានុក្រមដែលជាសម្បត្តិពិសេសឈ្មោះ __dict__ ។

print(*ថ្លៃលក់*.__dict__["*ប្រាក់*"]) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកសម្បត្តិឈ្មោះ ប្រាក់ របស់សិស្សឈ្មោះ ថ្លៃលក់ មកប្រើការ។ ការយកសម្បត្តិនេះមកប្រើត្រូវបានធ្វើឡើងតាមរយៈវចនានុក្រមដែលជាសម្បត្តិពិសេសឈ្មោះ __dict__ ។

សម្បត្តិពិសេសឈ្មោះ __doc__

នៅក្នុងថ្នាក់មួយ និងនៅក្នុងខាងលើគេបង្អស់ បើសិនជាយើងបង្កើតកម្រងអក្សរគ្មានឈ្មោះណាមួយ ឈ្មោះ __doc__ របស់សម្បត្តិពិសេសនៅក្នុងថ្នាក់នោះ នឹងត្រូវយកទៅភ្ជាប់ទៅនឹងកម្រងអក្សរនោះ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
class ទឹកប្រាក់() :
    """គឺជាថ្នាក់ប្រើសម្រាប់បង្កើតវត្ថុដែលជាទឹកប្រាក់។"""
    ប្រាក់ = 2000
    def __init__(សិស្ស, ប្រាក់=0) :
        សិស្ស.ប្រាក់ = ប្រាក់

ថ្លៃលក់ = ទឹកប្រាក់(1000)
print(ទឹកប្រាក់.__doc__)
```

"""*គឺជាថ្នាក់ប្រើសម្រាប់បង្កើតវត្ថុដែលជាទឹកប្រាក់។*""" គឺជាការបង្កើតកម្រងអក្សរគ្មានឈ្មោះមួយនៅក្នុងថ្នាក់ឈ្មោះ ទឹកប្រាក់ និងនៅខាងលើគេបង្អស់។ ប្រការនេះធ្វើឲ្យឈ្មោះ __doc__ ដែលជាឈ្មោះរបស់សម្បត្តិពិសេសនៅក្នុងថ្នាក់នោះ ត្រូវបានយកទៅភ្ជាប់នឹងកម្រងអក្សរនោះ។

ជាទូទៅ គេច្រើនប្រើសម្បត្តិឈ្មោះ __doc__ នេះជាឯកសារសម្រាប់កត់ត្រាទុកនូវកំណត់ពន្យល់ផ្សេងៗនៅក្នុងថ្នាក់ណាមួយ។

ដូចគ្នាដែរ បើសិនជាយើងបង្កើតកម្រងអក្សរគ្មានឈ្មោះណាមួយនៅក្នុងក្បួនណាមួយ និងនៅខាងលើគេបង្អស់ សម្បត្តិពិសេសឈ្មោះ __doc__ នៅក្នុងក្បួននោះ នឹងត្រូវយកទៅភ្ជាប់នឹងកម្រងអក្សរនោះដោយស្វ័យប្រវត្តិ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
def បង្ហាញព័ត៌មាន() :
    """ក្បួនប្រើសម្រាប់បង្ហាញព័ត៌មាន។"""
    print("ព័ត៌មានទាំងឡាយនឹងត្រូវបង្ហាញនៅទីនោះ។")

print(បង្ហាញព័ត៌មាន.__doc__)
```

"""ក្បួនប្រើសម្រាប់បង្ហាញព័ត៌មាន។""" គឺជាការបង្កើតកម្រងអក្សរគ្មានឈ្មោះមួយនៅខាងលើគេបង្អស់នៅក្នុងក្បួនឈ្មោះ បង្ហាញព័តិមាន ។ ប្រការនេះធ្វើឲ្យឈ្មោះ __doc__ ដែលជាឈ្មោះរបស់សម្បត្តិពិសេសនៅក្នុងក្បួននោះ ត្រូវយកទៅភ្ជាប់នឹងកម្រងអក្សរនោះ។

សម្បត្តិពិសេសឈ្មោះ __slots__

បើសិនជាយើងចង់កំណត់ពីចំនួននិងឈ្មោះនៃសម្បត្តិរបស់សិស្សនៃថ្នាក់ណាមួយនោះ យើងត្រូវបង្កើតកម្រងអថេរមួយមានឈ្មោះថា __slots__ ដែលមានធាតុជាឈ្មោះរបស់សម្បត្តិសិស្សនៃថ្នាក់នោះ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
class ទឹកប្រាក់() :
    __slots__ = ["ថ្លៃលក់", "ថ្លៃទិញ", "សោហ៊ុយ"]
    def __init__(សិស្ស, ថ្លៃលក់=0, ថ្លៃទិញ=0, សោហ៊ុយ=0) :
        សិស្ស.ថ្លៃលក់ = ថ្លៃលក់
        សិស្ស.ថ្លៃទិញ = ថ្លៃទិញ
        សិស្ស.សោហ៊ុយ = សោហ៊ុយ
```

```
ប្រាក់រកស៊ី = ទឹកប្រាក់(1000, 900, 25)
print(ប្រាក់រកស៊ី.ថ្ងៃលក់)
print(ប្រាក់រកស៊ី.ថ្ងៃទិញ)
print(ប្រាក់រកស៊ី.សោហ៊ុយ)
```

__slots__ = ["ថ្ងៃលក់", "ថ្ងៃទិញ", "សោហ៊ុយ"] គឺជាការបង្កើតសម្បត្តិពិសេសដែលជាកម្រងអថេរឈ្មោះ __slots__ ។ ធាតុទាំងឡាយនៅក្នុងកម្រងអថេរនេះគឺជាសម្បត្តិរបស់សិស្សនៃថ្នាក់ឈ្មោះ ទឹកប្រាក់ ។ ហើយការបង្កើតសម្បត្តិសិស្សដែលមានឈ្មោះខុសពីឈ្មោះដែលជាធាតុនៅក្នុងកម្រងអថេរឈ្មោះ __slots__ នេះ នឹងបណ្តាលឲ្យភាពមិនប្រក្រតីកើតមានឡើង។

សម្បត្តិឯកជន

នៅក្នុងថ្នាក់មួយ បើសិនជាមានសម្បត្តិមួយចំនួនមានសញ្ញា _ នេះពីរនៅពីមុខ សម្បត្តិទាំងនោះនឹងមិនអាចត្រូវយកទៅប្រើនៅខាងក្រៅថ្នាក់ដោយប្រើឈ្មោះដូចនោះបានឡើយ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
class ទឹកប្រាក់() :
   def __init__(សិស្ស, ប្រាក់=0) :
      សិស្ស.__ប្រាក់ = ប្រាក់

   def បង្ហាញព័ត៌មាន(សិស្ស) :
      print(សិស្ស.__ប្រាក់)

ថ្ងៃលក់ = ទឹកប្រាក់(1000)
ថ្ងៃលក់.បង្ហាញព័ត៌មាន()
print(ថ្ងៃលក់.__ប្រាក់)
```

សិស្ស.__ប្រាក់ = ប្រាក់ គឺជាបញ្ជាតម្រូវឲ្យបង្កើតសម្បតិឈ្មោះ __ប្រាក់ មួយទុកនៅក្នុងសិស្ស ឈ្មោះ សិស្ស នៅក្នុងស្ថាបនិក។

print(*សិស្ស.__ប្រាក់*) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកសម្បត្តិឈ្មោះ __ប្រាក់ របស់សិស្សឈ្មោះ សិស្ស មកប្រើនៅក្នុងវិធីឈ្មោះ បង្ហាញព័ត៌មាន ។ ប្រការនេះអាចធ្វើទៅបានដោយគ្មានបញ្ហាអ្វីឡើយ ព្រោះសម្បត្តិនោះត្រូវបានយកទៅប្រើនៅក្នុងថ្នាក់របស់សិស្ស ដែលមានសម្បត្តិនោះ។

print(*ថ្លៃលក់.__ប្រាក់*) គឺជាបញ្ជាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកសម្បត្តិឈ្មោះ __ប្រាក់ របស់សិស្សឈ្មោះ ថ្លៃលក់ មកប្រើនៅខាងក្រៅថ្នាក់របស់សិស្សនោះ។ ប្រការនេះមិនអាចធ្វើទៅបានឡើយ កំហុសមួយបានកើតមានឡើង។

គ្រប់សម្បត្តិនៅក្នុងថ្នាក់ដែលមានសញ្ញា _ នេះពីរនៅពីមុខគឺជា *សម្បត្តិឯកជន* (private attribute) ពីព្រោះវាមិនអាចត្រូវយកទៅប្រើនៅខាងក្រៅថ្នាក់របស់វាបានឡើយ។ ក៏ប៉ុន្តែ បើយើងពិតជាចង់យកសម្បត្តិឯកជនទាំងនោះទៅប្រើនៅខាងក្រៅថ្នាក់មែន យើងត្រូវធ្វើដូចខាងក្រោមនេះ៖

```
class ទឹកប្រាក់() :
   def __init__(សិស្ស, ប្រាក់=0) :
      សិស្ស.__ប្រាក់ = ប្រាក់

   def បង្ហាញព័ត៌មាន(សិស្ស) :
      print(សិស្ស.__ប្រាក់)

ថ្លៃលក់ = ទឹកប្រាក់(1000)
```

```
ថ្លៃលក់.បង្ហាញព័ត៌មាន()
print(ថ្លៃលក់._ទឹកប្រាក់__ប្រាក់)
```

print(*ថ្លៃលក់._ទឹកប្រាក់__ប្រាក់*) គឺជាបញ្ហាដែលនៅក្នុងនោះមានការតម្រូវឲ្យយកសម្បត្តិឈ្មោះ __ប្រាក់ របស់សិស្សឈ្មោះ ថ្លៃលក់ មកសរសេរនៅលើបង្អួចបឋម។ ប្រការនេះអាចធ្វើទៅបានដោយគ្មានបញ្ហាអ្វីឡើយ។

ដូច្នេះ ដើម្បីអាចយកសម្បត្តិឯកជនរបស់សិស្សណាម្នាក់មកប្រើនៅខាងក្រៅថ្នាក់បាន យើងត្រូវសរសេរសញ្ញា _ នេះរួមផ្សំនឹងឈ្មោះរបស់ថ្នាក់បន្ថែមទៅលើឈ្មោះរបស់សម្បត្តិឯកជននោះ។

លក្ខណៈសម្បត្តិ

យើងបានដឹងរួចមកហើយថា នៅក្នុងថ្នាក់មួយ បើសិនជាយើងបង្កើតវិធីពិសេសឈ្មោះ __setattr__ និងវិធីពិសេសឈ្មោះ __getattribute__ នៅពេលដែលយើងបង្កើតនិងឬយកសម្បត្តិរបស់សិស្សនៃថ្នាក់នោះទៅប្រើ វិធីទាំងនោះនឹងត្រូវយកមកប្រើជាស្វ័យប្រវត្តិ។ ម៉្យាងទៀត សម្បត្តិថ្នាក់គឺជាវត្ថុដែលធ្វើឲ្យថ្នាក់និមួយៗមានលក្ខណៈខុសៗគ្នា ពីព្រោះគ្រប់ថ្នាក់ទាំងឡាយណាដែលមានសម្បត្តិមានលក្ខណៈខុសគ្នា នឹងបង្កើតសិស្សដែលមានសម្បត្តិមានលក្ខណៈខុសៗគ្នាដែរ។ ដោយហេតុថា វិធីពិសេសឈ្មោះ __setattr__ និងវិធីពិសេសឈ្មោះ __getattribute__ ជាវិធីដែលមានមុខងារជាអ្នកបង្កើតឬយកសម្បត្តិសិស្សមកប្រើដូចនេះ វិធីទាំងពីរនេះមានតួនាទីសំខាន់ណាស់ក្នុងការកំណត់ពីលក្ខណៈនៃសម្បត្តិរបស់សិស្ស។

ទន្ទឹមគ្នានេះដែរ នៅក្នុងថ្នាក់មួយ បើសិនជាយើងយកថ្នាក់មានស្រាប់ឈ្មោះ property មកប្រើ យើងនឹងទទួលបានវត្ថុម៉្យាងដែលត្រូវហៅថា *លក្ខណៈសម្បត្តិ* (property attribute) ដែលមានតួនាទីសំខាន់ណាស់ក្នុងការកំណត់ពីលក្ខណៈនៃសម្បត្តិរបស់សិស្ស។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
class ទឹកប្រាក់() :
   def បង្កើតសម្បត្តិសិស្ស(សិស្ស, សម្បត្តិ) :
      print("សម្បត្តិសិស្សដែលជា", សម្បត្តិ, "ត្រូវបានបង្កើតឡើង។")
      សិស្ស.សម្បត្តិ = សម្បត្តិ

   def យកសម្បត្តិសិស្ស(សិស្ស) :
      print("សម្បត្តិសិស្សដែលជា", សិស្ស.សម្បត្តិ, "ត្រូវបានយកទៅប្រើ។")
      return សិស្ស.សម្បត្តិ

   def លុបសម្បត្តិសិស្ស(សិស្ស) :
      print("សម្បត្តិសិស្សដែលជា", សិស្ស.សម្បត្តិ, "ត្រូវបានលុបចោល។")
      del សិស្ស.សម្បត្តិ

   គុណលក្ខណៈ = property(យកសម្បត្តិសិស្ស, បង្កើតសម្បត្តិសិស្ស, លុបសម្បត្តិសិស្ស)

ប្រាក់រកស៊ី = ទឹកប្រាក់()
ប្រាក់រកស៊ី.គុណលក្ខណៈ = 1000
ប្រាក់រកស៊ី.គុណលក្ខណៈ
del ប្រាក់រកស៊ី.គុណលក្ខណៈ
```

គុណលក្ខណៈ = property(យកសម្បត្តិសិស្ស, បង្កើតសម្បត្តិសិស្ស, លុបសម្បត្តិសិស្ស) គឺជាការយកថ្នាក់មានស្រាប់ឈ្មោះ property មកប្រើដោយផ្តល់វត្ថុដែលជាវិធីចំនួនបីឲ្យទៅ

ស្ថាបនិកនៃថ្នាក់នោះ។ ជាលទ្ធផល វត្ថុមួយមានឈ្មោះថា គុណលក្ខណៈ ត្រូវបានបង្កើតឡើង ។

ប្រាក់រកស៊ី.គុណលក្ខណៈ = 1000 គឺជាការបង្កើតសម្បត្តិសម្រាប់សិស្សឈ្មោះ ប្រាក់រកស៊ី ដោយយកវត្ថុដែលជាលក្ខណៈសម្បត្តិឈ្មោះ គុណលក្ខណៈ មកប្រើ។ ប្រការនេះធ្វើឲ្យវិធីឈ្មោះ បង្កើតសម្បត្តិសិស្ស ត្រូវបានយកមកប្រើ និងសិស្សឈ្មោះ ប្រាក់រកស៊ី នោះត្រូវបានផ្តល់ជាដំណឹងសម្រាប់ដំណាង សិស្ស ហើយលេខ 1000 ត្រូវបានផ្តល់ជាដំណឹងសម្រាប់ដំណាង សម្បត្តិ នៅក្នុងវិធីឈ្មោះ បង្កើតសម្បត្តិសិស្ស នោះ។

ប្រាក់រកស៊ី.គុណលក្ខណៈ គឺជាការយកសម្បត្តិរបស់សិស្សឈ្មោះ ប្រាក់រកស៊ី មកប្រើដោយយកវត្ថុដែលជាលក្ខណៈសម្បត្តិឈ្មោះ គុណសម្បត្តិ មកប្រើ។ ប្រការនេះធ្វើឲ្យវិធីឈ្មោះ យកសម្បត្តិសិស្ស ត្រូវយកមកប្រើជាស្វ័យប្រវត្តិ និងសិស្សឈ្មោះ ប្រាក់រកស៊ី នោះត្រូវបានផ្តល់ជាដំណឹងសម្រាប់ដំណាង សិស្ស នៅក្នុងវិធីនោះ។

del *ប្រាក់រកស៊ី.គុណលក្ខណៈ* គឺជាការលុបសម្បត្តិរបស់សិស្សឈ្មោះ ប្រាក់រកស៊ី ដោយយកវត្ថុដែលជាលក្ខណៈសម្បត្តិឈ្មោះ គុណលក្ខណៈ មកប្រើ។ ប្រការនេះធ្វើឲ្យវិធីឈ្មោះ លុបសម្បត្តិសិស្ស ត្រូវយកមកប្រើជាស្វ័យប្រវត្តិ និងសិស្សឈ្មោះ ប្រាក់រកស៊ី នោះត្រូវផ្តល់ជាដំណឹងសម្រាប់ដំណាង សិស្ស នៅក្នុងវិធីនោះ។

វិធីឯកោ

វិធីឯកោ (static method) គឺជាវិធីទាំងឡាយណាដែលមិនត្រូវការសិស្សសម្រាប់ដំណាងទីមួយ ។ ដើម្បីបង្កើតវិធីឯកោ យើងត្រូវធ្វើដូចខាងក្រោមនេះ៖

```
class ទឹកប្រាក់() :
```

```
        def បង្ហាញព័ត៌មាន() :
            print("ព័ត៌មានត្រូវបង្ហាញនៅទីនេះ។")

        បង្ហាញព័ត៌មាន = staticmethod(បង្ហាញព័ត៌មាន)

ប្រាក់រកស៊ី = ទឹកប្រាក់()
ទឹកប្រាក់.បង្ហាញព័ត៌មាន()
ប្រាក់រកស៊ី.បង្ហាញព័ត៌មាន()
```

បង្ហាញព័ត៌មាន = staticmethod(បង្ហាញព័ត៌មាន) គឺជាការយកក្បួនមានស្រាប់ឈ្មោះ staticmethod មកប្រើដើម្បីកែវិធីឈ្មោះ បង្ហាញព័ត៌មាន ឲ្យទៅជាវិធីដែលលែងត្រូវការសិស្សសម្រាប់ដំណាងទីមួយ។ វិធីនេះត្រូវហៅថាវិធីឯកោ។

ទឹកប្រាក់.បង្ហាញព័ត៌មាន() គឺជាការយកវិធីឯកោឈ្មោះ បង្ហាញព័ត៌មាន មកប្រើតាមរយៈថ្នាក់របស់វា ដោយមិនចាំបាច់ផ្តល់ដំណឹងជាសិស្សណាម្នាក់ឲ្យទៅវាឡើយ។

យើងគួររំលឹកឡើងវិញថា ការយកវិធីផ្សេងៗមកប្រើតាមរយៈថ្នាក់ មិនទាមទារឲ្យយើងចាំបាច់ត្រូវតែផ្តល់ដំណឹងជាសិស្សឲ្យទៅវិធីនោះទេ តែប្រការដែលចាំបាច់គឺយើងត្រូវផ្តល់ដំណឹងឲ្យមានចំនួនគ្រប់គ្រាន់សម្រាប់ដំណាងនៅក្នុងវិធីទាំងនោះ។ ក៏ប៉ុន្តែមានវិធីខ្លះត្រូវការដំណឹងជាសិស្សដើម្បីយកសម្បត្តិនៅក្នុងនោះទៅប្រើការផ្សេងៗ ដូច្នេះ យើងចាំបាច់ត្រូវតែផ្តល់ដំណឹងជាសិស្សឲ្យទៅវិធីនោះ ទោះបីជាយើងយកវាមកប្រើតាមរយៈថ្នាក់ក៏ដោយ។

ប្រាក់រកស៊ី.បង្ហាញព័ត៌មាន() គឺជាការយកវិធីឯកោឈ្មោះ បង្ហាញព័ត៌មាន មកប្រើតាមរយៈសិស្សឈ្មោះ ប្រាក់រកស៊ី ។ ក្នុងករណីនេះ សិស្សឈ្មោះ ប្រាក់រកស៊ី មិនត្រូវបានផ្តល់ជាដំណឹងឲ្យទៅវិធីឈ្មោះ បង្ហាញព័ត៌មាន សម្រាប់ដំណាងទីមួយនៅក្នុងនោះឡើយ ព្រោះវិធីនេះបានក្លាយទៅជាវិធីឯកោរួចទៅហើយ។ បានន័យថា ការយកវិធីឯកោមកប្រើតាម

រយៈសិស្ស មិនធ្វើឲ្យសិស្សនោះត្រូវផ្តល់ជាដំណឹងឲ្យទៅវិធីនោះឡើយ ព្រោះវិធីឯកោមិនត្រូវការសិស្ស។

សរុបមក វិធីឯកោមានសណ្ឋានដូចជាក្បួនធម្មតាដែលជាក្បួននៅខាងក្រៅថ្នាក់ ទោះជាវាត្រូវបានបង្កើតឡើងនៅក្នុងថ្នាក់ក៏ដោយ។ ហើយដោយសារតែវិធីឯកោត្រូវបានបង្កើតឡើងនៅក្នុងថ្នាក់ ដូចនេះគ្រប់ការយកវិធីឯកោមកប្រើ ត្រូវតែធ្វើឡើងតាមថ្នាក់ឬសិស្សនៃថ្នាក់របស់វា។

វិធីប្រចាំថ្នាក់

វិធីប្រចាំថ្នាក់ (class method) គឺជាវិធីទាំងឡាយណាដែលត្រូវការថ្នាក់សម្រាប់ដំណាងទីមួយនៅក្នុងនោះ។ ដើម្បីបង្កើតវិធីប្រចាំថ្នាក់ យើងត្រូវធ្វើដូចខាងក្រោមនេះ៖

```
class ទឹកប្រាក់() :
  def បង្ហាញព័ត៌មាន(ថ្នាក់) :
    print("ព័ត៌មានត្រូវបង្ហាញនៅទីនេះ។")

  បង្ហាញព័ត៌មាន = classmethod(បង្ហាញព័ត៌មាន)

ប្រាក់រកស៊ី = ទឹកប្រាក់()
ទឹកប្រាក់.បង្ហាញព័ត៌មាន()
ប្រាក់រកស៊ី.បង្ហាញព័ត៌មាន()
```

បង្ហាញព័ត៌មាន = classmethod(បង្ហាញព័ត៌មាន) គឺជាការយកក្បួនមានស្រាប់ឈ្មោះ classmethod មកប្រើដើម្បីកែវិធីឈ្មោះ បង្ហាញព័តិមាន ឲ្យទៅជាវិធីប្រចាំថ្នាក់ ដែលជាវិធីត្រូវការថ្នាក់ណាមួយសម្រាប់ដំណាងទីមួយនៅក្នុងនោះ។

ទឹកប្រាក់.បង្ហាញព័ត៌មាន() គឺជាការយកវិធីប្រចាំថ្នាក់ឈ្មោះ បង្ហាញព័ត៌មាន មកប្រើតាមរយៈថ្នាក់ឈ្មោះ ទឹកប្រាក់ ។ ប្រការនេះបណ្តាលឲ្យថ្នាក់ឈ្មោះ ទឹកប្រាក់ នោះត្រូវបានផ្តល់ជាដំណឹងសម្រាប់ដំណាងទីមួយនៅក្នុងវិធីនោះជាស្វ័យប្រវត្តិ។

ប្រាក់រកស៊ី.បង្ហាញព័ត៌មាន() គឺជាការយកវិធីប្រចាំថ្នាក់ឈ្មោះ បង្ហាញព័ត៌មាន មកប្រើតាមរយៈសិស្សឈ្មោះ ប្រាក់រកស៊ី ដែលជាសិស្សនៃថ្នាក់ឈ្មោះ ទឹកប្រាក់ ។ ប្រការនេះបណ្តាលឲ្យថ្នាក់ឈ្មោះ ទឹកប្រាក់ នោះត្រូវបានផ្តល់ជាដំណឹងសម្រាប់ដំណាងទីមួយនៅក្នុងវិធីនោះជាស្វ័យប្រវត្តិ។

សរុបមក គ្រប់ការយកវិធីប្រចាំថ្នាក់ណាមួយមកប្រើ បណ្តាលឲ្យថ្នាក់ណាមួយត្រូវផ្តល់ឲ្យទៅវិធីនោះជាស្វ័យប្រវត្តិ ទោះចង់ឬចង់មិនចង់ក្តី។

រចនាករ

រចនាករ (decorator) គឺជាវត្ថុទាំងឡាយណាដែលមានសម្ថភាពអាចយកក្បួនឬវិធីណាមួយមកកែលម្អឲ្យទៅជាវត្ថុផ្សេងទៀត។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
class ទឹកប្រាក់() :
    def កែឲ្យទៅជាវិធីឯកោ(វិធី) :
        វិធី = staticmethod(វិធី)
        return វិធី

    def កែឲ្យទៅជាវិធីប្រចាំថ្នាក់(វិធី) :
        វិធី = classmethod(វិធី)
        return វិធី
```

```
@កែឲ្យទៅជាវិធីប្រចាំថ្នាក់
def បង្កើតទិន្នន័យគម្រូ(ថ្នាក់, ទិន្នន័យ) :
    ថ្នាក់.ទិន្នន័យ = ទិន្នន័យ

@កែឲ្យទៅជាវិធីឯកោ
def បង្ហាញព័ត៌មាន() :
    print(ទឹកប្រាក់.ទិន្នន័យ)


ប្រាក់រកស៊ី = ទឹកប្រាក់()
ប្រាក់រកស៊ី.បង្កើតទិន្នន័យគម្រូ(1000)
ប្រាក់រកស៊ី.បង្ហាញព័ត៌មាន()
```

@កែឲ្យទៅជាវិធីប្រចាំថ្នាក់ គឺជាបញ្ជាតម្រូវឲ្យយកវិធីឈ្មោះ កែឲ្យទៅជាវិធីប្រចាំថ្នាក់ មកប្រើជារចនាករដើម្បីកែវិធីឈ្មោះ បង្កើតទិន្នន័យគម្រូ នៅខាងក្រោមនោះឲ្យទៅជាវិធីប្រចាំថ្នាក់។

@កែឲ្យទៅជាវិធីឯកោ គឺជាបញ្ជាតម្រូវឲ្យយកវិធីឈ្មោះ កែឲ្យទៅជាវិធីឯកោ មកប្រើជារចនាករដើម្បីកែវិធីឈ្មោះ បង្ហាញព័ត៌មាន នៅខាងក្រោមនោះឲ្យទៅជាវិធីឯកោ។

ប្រាក់រកស៊ី.បង្កើតទិន្នន័យគម្រូ(1000) គឺជាការយកវិធីឈ្មោះ បង្កើតទិន្នន័យគម្រូ មកប្រើតាមរយៈសិស្សឈ្មោះ ប្រាក់រកស៊ី ។ ប្រការនេះធ្វើឲ្យថ្នាក់របស់សិស្សនោះត្រូវបានផ្តល់ជាដំណឹងសម្រាប់ដំណាងទីមួយនៅក្នុងនោះ ព្រោះវិធីនោះត្រូវបានកែឲ្យទៅជាវិធីប្រចាំថ្នាក់រួចទៅហើយ។

ប្រាក់រកស៊ី.បង្ហាញព័ត៌មាន() គឺជាការយកវិធីឈ្មោះ បង្ហាញព័ត៌មាន មកប្រើតាមរយៈសិស្ស ឈ្មោះ ប្រាក់រកស៊ី ។ ប្រការនេះមិនបានធ្វើឲ្យសិស្សនោះត្រូវផ្តល់ជាដំណឹងសម្រាប់ដំណាងទីមួយនៅក្នុងនោះ ព្រោះវិធីនោះត្រូវបានកែឲ្យទៅជាវិធីឯកោរួចទៅហើយ។

ថ្នាក់មេអរូបី

ថ្នាក់មេអរូបី (abstract superclass) គឺជាថ្នាក់មេទាំងឡាយណាដែលទាមទារឲ្យយើងបង្កើតវិធីមួយចំនួននៅក្នុងថ្នាក់រងរបស់វា ដើម្បីឲ្យវិធីមួយចំនួននៅក្នុងថ្នាក់មេនោះអាចត្រូវយកប្រើការបាន។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
class ទឹកប្រាក់() :
    def បង្ហាញប្រាក់ចំណេញ(សិស្ស) :
        សិស្ស.រកប្រាក់ចំណេញ()

class ប្រាក់រកស៊ី(ទឹកប្រាក់) :
    def __init__(សិស្ស, ថ្ងៃលក់, ថ្ងៃទិញ) :
        សិស្ស.ថ្ងៃលក់ = ថ្ងៃលក់
        សិស្ស.ថ្ងៃទិញ = ថ្ងៃទិញ

    def រកប្រាក់ចំណេញ(សិស្ស) :
        print(សិស្ស.ថ្ងៃលក់ - សិស្ស.ថ្ងៃទិញ)

ប្រាក់រកស៊ីលក់អង្ករ = ប្រាក់រកស៊ី(1000, 900)
ប្រាក់រកស៊ីលក់អង្ករ.បង្ហាញប្រាក់ចំណេញ()
```

សិស្ស.រកប្រាក់ចំណេញ() គឺជាការយកវិធីមិនទាន់ត្រូវបានបង្កើតឈ្មោះ រកប្រាក់ចំណេញ មកប្រើនៅក្នុងវិធីឈ្មោះ បង្ហាញប្រាក់ចំណេញ ដែលជាវិធីនៅក្នុងថ្នាក់ឈ្មោះ ទឹកប្រាក់ ។

យើងគួររំលឹកឡើងវិញជាថ្មីម្តងទៀតថា ការយកវត្ថុមិនទាន់ត្រូវបានបង្កើតមកប្រើនៅពេលបង្កើតក្បួននិងឬវិធីណាមួយ អាចធ្វើទៅបាន ពីព្រោះក្រុមបញ្ជានៅក្នុងក្បួននិងឬវិធីទាំងនោះមិនទាន់ត្រូវយកទៅអនុវត្តនៅឡើយទេ នៅពេលដែលក្បួននិងឬវិធីទាំងនោះត្រូវបានបង្កើតឡើង។ ក្រុមបញ្ជាទាំងនោះត្រូវយកទៅអនុវត្តតែនៅពេលណាដែលក្បួននិងឬវិធីទាំងនោះត្រូវយកទៅប្រើតែប៉ុណ្ណោះ។

def *រកប្រាក់ចំណេញ(សិស្ស)* : គឺជាការបង្កើតវិធីឈ្មោះ រកប្រាក់ចំណេញ នៅក្នុងថ្នាក់រងឈ្មោះ ប្រាក់រកស៊ី ។

ប្រាក់រកស៊ីលក់អង្ករ.បង្ហាញប្រាក់ចំណេញ() គឺជាការយកវិធីឈ្មោះ បង្ហាញប្រាក់ចំណេញ នៅក្នុងថ្នាក់មេឈ្មោះ ទឹកប្រាក់ មកប្រើតាមរយៈសិស្សនៃថ្នាក់រងឈ្មោះ ប្រាក់រកស៊ីលក់អង្ករ ។ ប្រការនេះធ្វើឲ្យវិធីឈ្មោះ រកប្រាក់ចំណេញ នៅក្នុងថ្នាក់រងក៏ត្រូវយកមកប្រើដែរ។ ដោយហេតុថាវិធីឈ្មោះ រកប្រាក់ចំណេញ នេះត្រូវបានបង្កើតរួចហើយ ដូចនេះ ការយកវិធីឈ្មោះ បង្ហាញប្រាក់ចំណេញ នៅក្នុងថ្នាក់មេឈ្មោះ ទឹកប្រាក់ មកប្រើមិនបង្កឲ្យមានបញ្ហាអ្វីឡើយ។

សរុបមក ថ្នាក់មេឈ្មោះ ទឹកប្រាក់ គឺជាថ្នាក់មេអរូបី ព្រោះវាទាមទារឲ្យយើងបង្កើតវិធីមួយឈ្មោះ រកប្រាក់ចំណេញ នៅក្នុងរងរបស់វាដើម្បីអាចឲ្យវិធីរបស់វាឈ្មោះ បង្ហាញប្រាក់ចំណេញ អាចត្រូវយកទៅប្រើការបាន។

ស្វ័យសេវា

ស្វ័យសេវា (recursion) គឺជាការយកក្បួនឬវិធីណាមួយមកប្រើនៅក្នុងក្បួនឬវិធីដដែលនោះ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
ប៉ុន្មានដង = 0
def បង្ហាញព័ត៌មាន() :
   global ប៉ុន្មានដង
   ប៉ុន្មានដង += 1
   print("ក្បួនឈ្មោះបង្ហាញព័ត៌មានត្រូវបានយកទៅប្រើចំនួន {0} ដង".format(ប៉ុន្មានដង))
   if ប៉ុន្មានដង < 10 :
      បង្ហាញព័ត៌មាន()
   print("បញ្ហានេះត្រូវបានទុកចោលជាលើកទី {0}".format(ប៉ុន្មានដង))
   ប៉ុន្មានដង -= 1

បង្ហាញព័ត៌មាន()
```

បង្ហាញព័ត៌មាន() គឺជាការយកក្បួនឈ្មោះ បង្ហាញព័ត៌មាន មកប្រើនៅក្នុងក្បួនឈ្មោះ បង្ហាញព័ត៌មាន ដដែលនោះ។ ដូចនេះ នៅពេលដែលក្រុមបញ្ហានៅក្នុងក្បួននោះត្រូវបានយកទៅអនុវត្តជាបន្តបន្ទាប់រហូតដល់កន្លែងដែលមានបញ្ជា បង្ហាញព័ត៌មាន() ក្បួនដដែលនោះក៏ត្រូវបានយកមកប្រើ ដែលជាប្រការធ្វើឲ្យបញ្ហានៅខាងក្រោមនោះត្រូវបានទុកចោល ហើយក្រុមបញ្ហានៅក្នុងក្បួនដដែលនោះ ក៏ត្រូវបានយកមកអនុវត្តសារជាថ្មីម្តងទៀត។ ទង្វើរបៀបនេះ ត្រូវបានប្រព្រឹត្តទៅជាដដែលៗរហូតដល់បញ្ជា បង្ហាញព័ត៌មាន() លែងត្រូវបានយកមកអនុវត្ត ដែលជាប្រការធ្វើឲ្យបញ្ហាដែលត្រូវបានទុកចោលនោះត្រូវបានយកមក

អនុវត្តវិញម្តង។ ហើយការអនុវត្តន៍បញ្ជាដែលត្រូវបានទុកចោលនោះ ត្រូវប្រព្រឹត្តទៅជាច្រើនលើកច្រើនសារ។ ពោលគឺបើបញ្ជាទាំងនោះត្រូវបានទុកចោលចំនួន n ដង ការអនុវត្តន៍បញ្ជាទាំងនោះក៏ត្រូវធ្វើឡើងចំនួន n ដងដែរ។

ការចម្លងសមាសវត្ថុ

ការចម្លងសើរើ

យើងអាចយកសមាសវត្ថុផ្សេងៗដែលមានធាតុជាសមាសវត្ថុមួយចំនួនទៀតមកចម្លងបង្កើតជាសមាសវត្ថុថ្មី ដោយធ្វើដូចខាងក្រោមនេះ៖

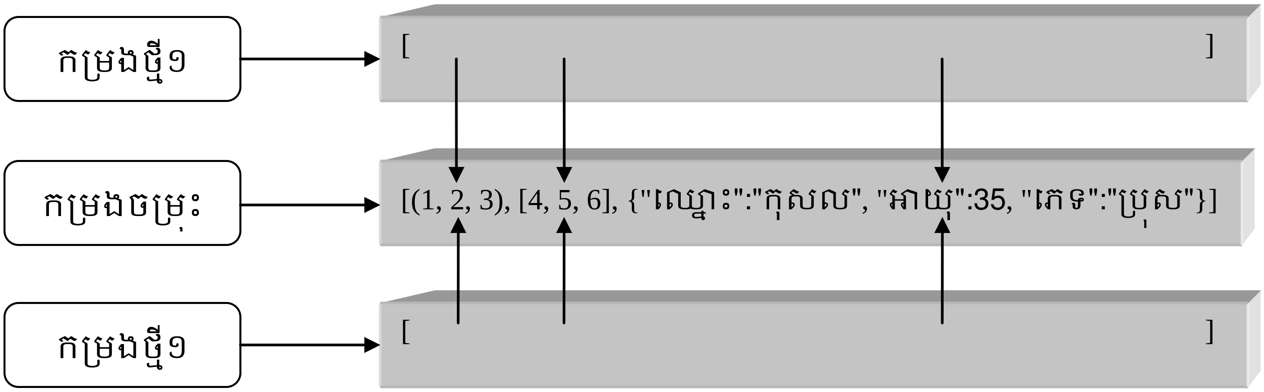
```
កម្រងចម្រុះ = [(1, 2, 3), [4, 5, 6], {"ឈ្មោះ":"កុសល", "អាយុ":35, "ភេទ":"ប្រុស"}]
កម្រងថ្មី១ = list(កម្រងចម្រុះ)
កម្រងថ្មី២ = កម្រងចម្រុះ[:]
print(កម្រងចម្រុះ)
print(កម្រងថ្មី១)
print(កម្រងថ្មី២)
```

កម្រងថ្មី១ = list(កម្រងចម្រុះ) គឺជាការយកកម្រងអថេរឈ្មោះ កម្រងចម្រុះ មកចម្លងបង្កើតជាកម្រងអថេថ្មីមួយទៀតមានឈ្មោះថា កម្រងថ្មី១ ដោយប្រើថ្នាក់មានស្រាប់ឈ្មោះ list ។

កម្រងថ្មី២ = កម្រងចម្រុះ[:] គឺជាការយកកម្រងអថេរឈ្មោះ កម្រងចម្រុះ មកចម្លងបង្កើតជាកម្រងអថេថ្មីមួយទៀតមានឈ្មោះថា កម្រងថ្មី២ ដោយធ្វើប្រមាណវិធីកាត់ចម្លង។

ការយកសមាសវត្ថុមកចម្លងបង្កើតជាសមាសវត្ថុថ្មីតាមរបៀបដូចខាងលើនេះ មិនមែនជាការបង្កើតសមាសវត្ថុថ្មីដោយឡែកផ្សេងទៀតឡើយ គឺវាគ្រាន់តែជាការបង្កើតសមាសវត្ថុថ្មី

ដែលមានធាតុជាធាតុរបស់សមាសវត្ថុចាស់តែប៉ុណ្ណោះ។ ជាក់ស្តែង នៅក្នុងកម្មវិធីខាងលើនេះ កម្រងអថេរឈ្មោះ កម្រងចម្រុះ កម្រងថ្មី១ និង កម្រងថ្មី២ មានធាតុទាំងឡាយជាសមាសវត្ថុតែមួយដូចគ្នា ពោលគឺធាតុរបស់កម្រងអថេរទាំងបីនោះគឺជាកម្រងថេរ (1, 2, 3) កម្រងអថេរ [4, 5, 6] និងវចនានុក្រម {"ឈ្មោះ":"កុសល", "អាយុ":35, "ភេទ":"ប្រុស"} តែមួយដូចគ្នា។



ការចម្លងបង្កើតសមាសវត្ថុថ្មីតាមរបៀបដូចខាងលើនេះហៅថា *ការចម្លងសើរលើ* (shallow copy) ពីព្រោះវាជាការចម្លងដែលបង្កើតវត្ថុថ្មីផ្សេងទៀតដែលមានធាតុជាធាតុរបស់សមាសវត្ថុចាស់ដែលត្រូវបានយកមកចម្លង។ មួយវិញទៀត បើសិនជាធាតុទាំងនោះត្រូវបានកែប្រែ ធាតុនៅក្នុងសមាសវត្ថុដែលទាក់ទងទាំងអស់ក៏ត្រូវបានកែប្រែដែរ ព្រោះធាតុទាំងនោះជាវត្ថុតែមួយ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
កម្រងចម្រុះ = [(1, 2, 3), [4, 5, 6], {"ឈ្មោះ":"កុសល", "អាយុ":35, "ភេទ":"ប្រុស"}]
កម្រងថ្មី១ = list(កម្រងចម្រុះ)
កម្រងថ្មី២ = កម្រងចម្រុះ[:]
កម្រងចម្រុះ[1][1] = True
កម្រងថ្មី១[2]["ឈ្មោះ"] = "សុខ"
កម្រងថ្មី២[2]["អាយុ"] = 34
```

```
print(កម្រងចម្រុះ)
print(កម្រងថ្មី១)
print(កម្រងថ្មី២)
```

កម្រងចម្រុះ[1][1] = True គឺជាការផ្លាស់ប្តូរធាតុមានលេខរៀង 1 នៅក្នុងកម្រងអថេរ [4, 5, 6] ដែលជាធាតុមានលេខរៀង 1 នៅក្នុងកម្រងអថេរឈ្មោះ កម្រងចម្រុះ ។ ប្រការនេះបានធ្វើឲ្យធាតុនៅក្នុងកម្រងអថេរឈ្មោះ កម្រងថ្មី១ និងកម្រងអថេរឈ្មោះ កម្រងថ្មី២ ក៏ត្រូវបានផ្លាស់ប្តូរដែរ ពីព្រោះកម្រងអថេរទាំងនោះមានធាតុជាកម្រងអថេរ [4, 5, 6] តែមួយដូចគ្នា។

កម្រងថ្មី១[2]["*ឈ្មោះ*"] = "*សុខ*" គឺជាការផ្លាស់ប្តូរតម្លៃជាប់នឹងកូនសោរ ឈ្មោះ នៅក្នុងវចនានុក្រម {"*ឈ្មោះ*":"*កុសល*", "*អាយុ*":35, "*ភេទ*":"*ប្រុស*"} ដែលជាធាតុមានលេខរៀង 2 នៅក្នុងកម្រងអថេរឈ្មោះ កម្រងថ្មី១ ។ ប្រការនេះបានធ្វើឲ្យធាតុនៅក្នុងកម្រងអថេរឈ្មោះ កម្រងចម្រុះ និងកម្រងអថេរឈ្មោះ កម្រងថ្មី២ ក៏ត្រូវបានផ្លាស់ប្តូរដែរ ពីព្រោះកម្រងអថេរទាំងនោះមានធាតុជាវចនានុក្រម {"*ឈ្មោះ*":"*កុសល*", "*អាយុ*":35, "*ភេទ*":"*ប្រុស*"} តែមួយដូចគ្នា ។

កម្រងថ្មី២[2]["*អាយុ*"] = 34 គឺជាការផ្លាស់ប្តូរតម្លៃជាប់នឹងកូនសោរ អាយុ នៅក្នុងវចនានុក្រម {"*ឈ្មោះ*":"*កុសល*", "*អាយុ*":35, "*ភេទ*":"*ប្រុស*"} ដែលជាធាតុមានលេខរៀង 2 នៅក្នុងកម្រងអថេរឈ្មោះ កម្រងថ្មី២ ។ ប្រការនេះបានធ្វើឲ្យធាតុនៅក្នុងកម្រងអថេរឈ្មោះ កម្រងចម្រុះ និងកម្រងអថេរឈ្មោះ កម្រងថ្មី១ ក៏ត្រូវបានផ្លាស់ប្តូរដែរ ពីព្រោះកម្រងអថេរទាំងនោះមានធាតុជាវចនានុក្រម {"*ឈ្មោះ*":"*កុសល*", "*អាយុ*":35, "*ភេទ*":"*ប្រុស*"} តែមួយដូចគ្នា ។

ក៏ប៉ុន្តែ យើងត្រូវធ្វើការកត់សំគាល់ផងដែរថា បើសិនជាយើងធ្វើការផ្លាស់ប្តូរធាតុទាំងមូលនៅក្នុងកម្រងអថេរណាមួយ កម្រងអថេរដទៃទៀតនឹងមិនទទួលរងនូវផលប៉ះពាល់អ្វីឡើយ ពីព្រោះការផ្លាស់ប្តូរធាតុទាំងមូល មិនមែនជាការយកធាតុមកកែប្រែទេ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
កម្រងចម្រុះ = [(1, 2, 3), [4, 5, 6], {"ឈ្មោះ":"កុសល", "អាយុ":35, "ភេទ":"ប្រុស"}]
កម្រងថ្មី១ = list(កម្រងចម្រុះ)
កម្រងថ្មី២ = កម្រងចម្រុះ[:]
កម្រងចម្រុះ[1] = True
កម្រងថ្មី១[2] = "សុខ"
កម្រងថ្មី២[2] = 34
print(កម្រងចម្រុះ)
print(កម្រងថ្មី១)
print(កម្រងថ្មី២)
```

កម្រងចម្រុះ[1] = True គឺជាការផ្លាស់ប្តូរធាតុមានលេខរៀង 1 នៅក្នុងកម្រងអថេរឈ្មោះ កម្រងចម្រុះ ។ ប្រការនេះមិនបានធ្វើឲ្យមានផលប៉ះពាល់ដល់កម្រងអថេរឈ្មោះ កម្រងថ្មី១ និង កម្រងថ្មី២ ឡើយ។

កម្រងថ្មី១[2] = "សុខ" គឺជាការផ្លាស់ប្តូរធាតុមានលេខរៀង 2 នៅក្នុងកម្រងអថេរឈ្មោះ កម្រងថ្មី១ ។ ប្រការនេះមិនបានធ្វើឲ្យមានផលប៉ះពាល់ដល់កម្រងអថេរឈ្មោះ កម្រងចម្រុះ និង កម្រងថ្មី២ ឡើយ។

កម្រងថ្មី២[2] = 34 គឺជាការផ្លាស់ប្តូរធាតុមានលេខរៀង 2 នៅក្នុងកម្រងអថេរឈ្មោះ កម្រងថ្មី២ ។ ប្រការនេះមិនបានធ្វើឲ្យមានផលប៉ះពាល់ដល់កម្រងអថេរឈ្មោះ កម្រងចម្រុះ និង កម្រងថ្មី១ ឡើយ។

ការចម្លងទាំងស្រុង

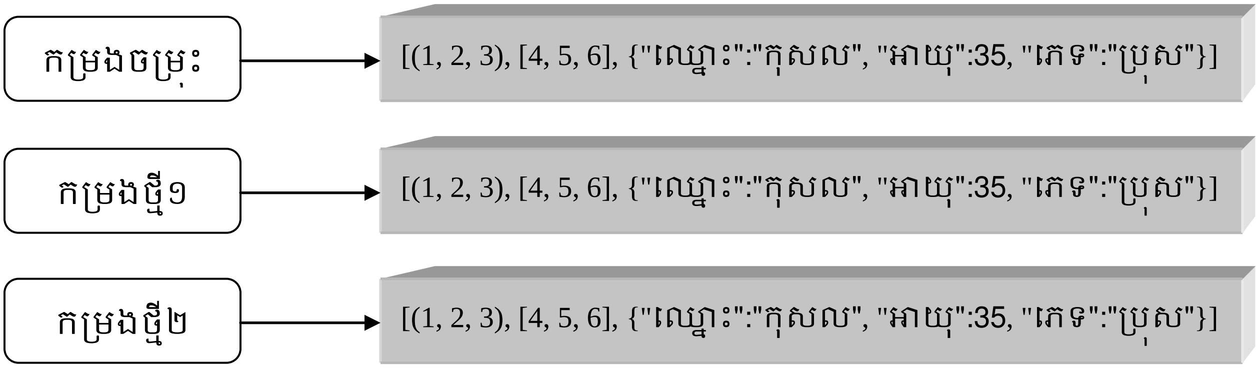
ផ្ទុយទៅវិញ បើសិនជាយើងចង់យកសមាសវត្ថុផ្សេងៗមកចម្លងបង្កើតជាសមាសវត្ថុថ្មីទៀត ដែលមានធាតុដាច់ដោយឡែករៀងៗខ្លួន យើងត្រូវធ្វើដូចខាងក្រោមនេះ៖

```
from copy import deepcopy as ចម្លងទាំងស្រុង

កម្រងចម្រុះ = [(1, 2, 3), [4, 5, 6], {"ឈ្មោះ":"កុសល", "អាយុ":35, "ភេទ":"ប្រុស"}]
កម្រងថ្មី១ = ចម្លងទាំងស្រុង(កម្រងចម្រុះ)
កម្រងថ្មី២ = ចម្លងទាំងស្រុង(កម្រងចម្រុះ)
print(កម្រងចម្រុះ)
print(កម្រងថ្មី១)
print(កម្រងថ្មី២)
```

from copy import deepcopy as *ចម្លងទាំងស្រុង* គឺជាការចម្លងយកក្បួនឈ្មោះ deepcopy នៅក្នុងសាស្ត្រាឈ្មោះ copy មកដាក់ឈ្មោះថាជាភាសាខ្មែរថា ចម្លងទាំងស្រុង ។ សាស្ត្រាឈ្មោះ copy គឺជាសាស្ត្រាដែលស្ថិតនៅក្នុងបណ្ណាល័យមជ្ឈឹម។

កម្រងថ្មី១ = *ចម្លងទាំងស្រុង*(*កម្រងចម្រុះ*) គឺជាការយកក្បួនឈ្មោះ ចម្លងទាំងស្រុង ឬ deepcopy នៅក្នុងសាស្ត្រាឈ្មោះ copy មកប្រើដើម្បីយកកម្រងអថេរឈ្មោះ កម្រងចម្រុះ មកចម្លងបង្កើតជាកម្រងអថេរថ្មីឈ្មោះ កម្រងថ្មី១ ដែលមានធាតុដាច់ដោយឡែកតែឯង។

កម្រងថ្មី២ = ចម្លងទាំងស្រុង(កម្រងចម្រុះ) គឺជាការយកក្បួនឈ្មោះ ចម្លងទាំងស្រុង ឬ deepcopy នៅក្នុងសាស្ត្រាឈ្មោះ copy មកប្រើដើម្បីយកកម្រងអថេរឈ្មោះ កម្រងចម្រុះ មកចម្លងបង្កើតជាកម្រងអថេរថ្មីឈ្មោះ កម្រងថ្មី២ ដែលមានធាតុដាច់ដោយឡែកតែឯង។



នៅក្នុងករណីមានការយកសមាសវត្ថុមកចម្លងបង្កើតជាសមាសវត្ថុថ្មីដែលមានធាតុដាច់ដោយឡែកតែឯង ការយកធាតុនៅក្នុងសមាសវត្ថុណាមួយមកកែប្រែ មិនធ្វើឲ្យធាតុដូចគ្នានៅក្នុងសមាសវត្ថុផ្សេងទៀតត្រូវបានកែប្រែឡើយ ពីព្រោះធាតុទាំងនោះជាវត្ថុខុសៗគ្នា។ ម៉្យាងទៀត ការយកសមាសវត្ថុមកចម្លងដោយប្រើក្បួនឈ្មោះ ចម្លងទាំងស្រុង ឬ deepcopy នេះ ហៅថា *ការចម្លងទាំងស្រុង* (deepcopy) ពីព្រោះធាតុទាំងឡាយនៅក្នុងសមាសវត្ថុដើមត្រូវបានចម្លងទាំងស្រុងដើម្បីបង្កើតសមាសវត្ថុថ្មី។

អាណាព្យាបាល និងបញ្ជា *with*

អាណាព្យាបាល

អាណាព្យាបាល (cotext manager) គឺជាវត្ថុម៉្យាងដែលជាសិស្សនៃថ្នាក់ទាំងឡាយណាដែលនៅក្នុងនោះមានវិធីពិសេសឈ្មោះ __enter__ និងវិធីពិសេសឈ្មោះ __exit__ ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
class ការរ៉ាប់រង() :
    def __enter__(សិស្ស) :
        print("វិធីពិសេសឈ្មោះ __enter__ ត្រូវបានយកទៅប្រើ។")

    def __exit__(សិស្ស, ប្រភេទនៃកំហុស, កំហុស, ព័ត៌មានស្ដីពីកំហុស) :
        print("វិធីពិសេសឈ្មោះ __exit__ ត្រូវបានយកទៅប្រើ។")
        print(ប្រភេទនៃកំហុស)
        print(កំហុស)
        print(ព័ត៌មានស្ដីពីកំហុស)

ធានារ៉ាប់រង១ = ការរ៉ាប់រង()
```

def __enter__(សិស្ស) : គឺជាការបង្កើតវិធីពិសេសឈ្មោះ __enter__ ដែលជាវិធីមិនត្រូវការដំណឹងណាផ្សេងក្រៅពីសិស្សនៃថ្នាក់របស់វាឡើយ។

def __exit__(សិស្ស, ប្រភេទនៃកំហុស, កំហុស, ព័ត៌មានស្ដីពីកំហុស) : គឺជាការបង្កើតវិធីពិសេសឈ្មោះ __exit__ ដែលជាវិធីត្រូវការដំណឹងបីផ្សេងទៀតក្រៅពីសិស្សនៃថ្នាក់របស់វា។

ធានារ៉ាប់រង១ = ការរ៉ាប់រង() គឺជាការបង្កើតសិស្សនៃថ្នាក់ឈ្មោះ ការរ៉ាប់រង ម្នាក់មានឈ្មោះថា ធានារ៉ាប់រង១ ។ ដោយហេតុថានៅក្នុងថ្នាក់ឈ្មោះ ការរ៉ាប់រង មានវិធីពិសេសឈ្មោះ __enter__ និងវិធីពិសេសឈ្មោះ __exit__ ដូចនេះសិស្សនៃថ្នាក់នោះត្រូវហៅថាអាណាព្យាបាល។ ដូចនេះវត្ថុឈ្មោះ ធានារ៉ាប់រង គឺជាអាណាព្យាបាលម្នាក់។

បញ្ជា with

បញ្ជា with គឺជាបញ្ជាដែលត្រូវយកទៅប្រើជាមួយនឹងអាណាព្យាបាលណាមួយ ក្នុងគោលបំណងតម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងបញ្ជា with នោះ។ ហើយនៅពេលដែលក្រុមបញ្ជានៅក្នុងបញ្ជា with ត្រូវយកទៅអនុវត្ត វិធីពិសេសសេឈ្មោះ __enter__ នៅក្នុងថ្នាក់របស់អាណាព្យាបាលនឹងត្រូវយកមកប្រើជាស្វ័យប្រវត្តិ។ ហើយលុះដល់ក្រុមបញ្ជានៅក្នុងបញ្ជា with ត្រូវបានអនុវត្តបានចប់សព្វគ្រប់អស់ហើយ វិធីពិសេសឈ្មោះ __exit__ នៅក្នុងថ្នាក់របស់អាណាព្យាបាលក៏ត្រូវយកប្រើជាស្វ័យប្រវត្តិដែរ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
class ការរ៉ាប់រង() :
    def __enter__(សិស្ស) :
        print("វិធីពិសេសឈ្មោះ __enter__ ត្រូវបានយកទៅប្រើ។")

    def __exit__(សិស្ស, ប្រភេទនៃកំហុស, កំហុស, ព័ត៌មានស្តីពីកំហុស) :
        print("វិធីពិសេសឈ្មោះ __exit__ ត្រូវបានយកទៅប្រើ។")
        print(ប្រភេទនៃកំហុស)
        print(កំហុស)
        print(ព័ត៌មានស្តីពីកំហុស)

ធានារ៉ាប់រង១ = ការរ៉ាប់រង()
with ធានារ៉ាប់រង១ :
    print("ក្រុមបញ្ជានៅក្នុងបញ្ជា with ត្រូវបានយកទៅអនុវត្ត។")
```

with ធានារ៉ាប់រង១ : គឺជាការប្រើបញ្ជា with ជាមួយនឹងអាណាព្យាបាលឈ្មោះ ធានារ៉ាប់រង១ ដែលជាសិស្សនៃថ្នាក់ឈ្មោះ ការរ៉ាប់រង ។

យើងឃើញថា ការប្រើបញ្ជា with មិនតម្រូវឲ្យមានលក្ខខ័ណ្ឌអ្វីទាំងអស់ ដូចនេះការអនុវត្ត បញ្ជា with នឹងត្រូវធ្វើឡើងនៅក្នុងករណីទាំងអស់ ដូចគ្នាទៅនឹងការអនុវត្តន៍បញ្ជា class និងឬ def ដែរ។ ម៉្យាងទៀតនៅពេលដែលបញ្ជា with ត្រូវយកទៅអនុវត្ត វិធីពិសេសឈ្មោះ __enter__ នៅក្នុងថ្នាក់ឈ្មោះ ការរ៉ាប់រង ត្រូវយកមកប្រើជាស្វ័យប្រវត្តិ តាមរយៈ អាណាព្យាបាលឈ្មោះ ធានារ៉ាប់រង១ ។ បន្ទាប់មកទៀត នៅពេលដែលក្រុមបញ្ជានៅក្នុង បញ្ជា with ត្រូវបានអនុវត្តបានចប់សព្វគ្រប់អស់ហើយ វិធីពិសេសឈ្មោះ __exit__ នៅក្នុង ថ្នាក់ឈ្មោះ ការរ៉ាប់រង ក៏ត្រូវយកមកប្រើជាស្វ័យប្រវត្តិតាមរយៈអាណាព្យាបាលឈ្មោះ ធានារ៉ាបរង១ នោះដែរ ហើយដំណឹងដែលជាមោឃៈវត្ថុ None ចំនួនបីត្រូវផ្តល់ឲ្យទៅដំណាង ដែលមិនមែនជា សិស្ស នៅក្នុងវិធីនោះ។

មួយវិញទៀត ការយកវិធីពិសេសឈ្មោះ __exit__ មកប្រើក៏ត្រូវធ្វើឡើងជាស្វ័យប្រវត្តិដែរ នៅពេលណាដែលការអនុវត្តន៍ក្រុមបញ្ជានៅក្នុងបញ្ជា with បណ្តាលឲ្យមានភាពមិនប្រក្រតីកើតមានឡើង។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
class ការរ៉ាប់រង() :
  def __enter__(សិស្ស) :
    print("វិធីពិសេសឈ្មោះ __enter__ ត្រូវបានយកទៅប្រើ។")
    return សិស្ស

  def __exit__(សិស្ស, ប្រភេទនៃកំហុស, កំហុស, ព័ត៌មានស្តីពីកំហុស) :
    print("វិធីពិសេសឈ្មោះ __exit__ ត្រូវបានយកទៅប្រើ។")
    print(ប្រភេទនៃកំហុស)
    print(កំហុស)
    print(ព័ត៌មានស្តីពីកំហុស)
```

```
ធានារ៉ាប់រង១ = ការរ៉ាប់រង()
with ធានារ៉ាប់រង១ :
    raise SyntaxError()
    print("ក្រុមបញ្ជានៅក្នុងបញ្ជា with ត្រូវបានយកទៅអនុវត្ត។")
```

raise SyntaxError() គឺជាបញ្ជាតម្រូវឲ្យបង្កើតភាពមិនប្រក្រតីប្រភេទ SyntaxError ។ ដូចនេះ នៅពេលដែលក្រុមបញ្ជានៅក្នុបញ្ជា with ត្រូវយកទៅអនុវត្ត ភាពមិនប្រក្រតីប្រភេទ SyntaxError ត្រូវកើតមានឡើង។ ប្រការនេះបណ្តាលឲ្យវិធីពិសេសឈ្មោះ __exit__ នៃថ្នាក់ឈ្មោះ ការរ៉ាប់រង ត្រូវយកមកប្រើជាស្វ័យប្រវត្តិតាមរយៈអាណាព្រាបាលឈ្មោះ ធានារ៉ាប់រង១ ហើយប្រភេទនៃភាពមិនប្រក្រតី វត្ថុនៃភាពមិនប្រក្រតី និងវត្ថុដែលជាសិស្សនៃថ្នាក់ឈ្មោះ traceback នឹងត្រូវផ្តល់ជាដំណឹងរៀងគ្នាសម្រាប់ដំណាងនៅក្នុងវិធីឈ្មោះ __exit__ នោះ។

ម៉្យាងទៀត យើងសង្កេតឃើញថា នៅពេលភាពមិនប្រក្រតីកើតមានឡើង វិធីពិសេសឈ្មោះ __exit__ ត្រូវយកមកប្រើជាស្វ័យប្រវត្តិមែន តែភាពមិនប្រក្រតីមិនត្រូវបានទទួលយកឡើយ។ ប្រការនេះបណ្តាលឲ្យកម្មវិធីត្រូវឈប់លែងដំណើរការនៅត្រឹមនោះ។ ក្នុងករណីនេះ ដើម្បីឲ្យភាពមិនប្រក្រតីត្រូវទទួលយក វិធីពិសេសឈ្មោះ __exit__ ត្រូវតែបញ្ជូនតក្កវត្ត True ទៅកាន់កន្លែងណាដែលវាត្រូវបានយកទៅប្រើ។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
class ការរ៉ាប់រង() :
    def __enter__(សិស្ស) :
        print("វិធីពិសេសឈ្មោះ __enter__ ត្រូវបានយកទៅប្រើ។")
        return សិស្ស

    def __exit__(សិស្ស, ប្រភេទនៃកំហុស, កំហុស, ព័ត៌មានស្តីពីកំហុស) :
        print("វិធីពិសេសឈ្មោះ __exit__ ត្រូវបានយកទៅប្រើ។")
        print(ប្រភេទនៃកំហុស)
```

```
        print(កំហុស)
        print(ព័ត៌មានស្តីពីកំហុស)
        return True

ជានាវ៉ាប់រង១ = ការវ៉ាប់រង()

with ជានាវ៉ាប់រង១ :
    raise SyntaxError()
    print("ក្រុមបញ្ជានៅក្នុងបញ្ជា with ត្រូវបានយកទៅអនុវត្ត។")
```

return True គឺជាការបញ្ជូនតក្កវត្ថុ True ទៅកាន់កន្លែងដែលវិធីពិសេសឈ្មោះ __exit__ ត្រូវយកទៅប្រើជាស្វ័យប្រវត្តិ។ ប្រការនេះធ្វើឲ្យភាពមិនប្រក្រតីត្រូវទទួលយក ហើយកម្មវិធីត្រូវចប់ទៅដោយសម្រួល។

បញ្ជា with/as

បញ្ជា with/as គឺជាបញ្ជាដែលត្រូវយកទៅប្រើជាមួយនឹងអាណាព្យាបាលណាមួយ ក្នុងគោលបំណងតម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងនោះ និងភ្ជាប់ឈ្មោះណាមួយទៅនឹងវត្ថុដែលវិធីពិសេសឈ្មោះ __enter__ បញ្ជូនមក។ ពិនិត្យកម្មវិធីខាងក្រោមនេះ៖

```
class ការវ៉ាប់រង() :
    def __enter__(សិស្ស) :
        print("វិធីពិសេសឈ្មោះ __enter__ ត្រូវបានយកទៅប្រើ។")
        return "វត្ថុក្បួន __enter__ បញ្ជូនមក។"

    def __exit__(សិស្ស, ប្រភេទនៃកំហុស, កំហុស, ព័ត៌មានស្តីពីកំហុស) :
        print("វិធីពិសេសឈ្មោះ __exit__ ត្រូវបានយកទៅប្រើ។")
        print(ប្រភេទនៃកំហុស)
        print(កំហុស)
```

```
        print(ព័ត៌មានស្តីពីកំហុស)
        return True

ធានារ៉ាប់រង១ = ការរ៉ាប់រង()
with ធានារ៉ាប់រង១ as វត្ថុ :
    print("ក្រុមបញ្ជានៅក្នុងបញ្ជា with ត្រូវបានយកទៅអនុវត្ត។")
    print(វត្ថុ)
```

with ធានារ៉ាប់រង១ as វត្ថុ : គឺជាការប្រើបញ្ជា with/as ជាមួយនឹងអាណាព្យាបាលឈ្មោះ ធានារ៉ាប់រង១ ដើម្បីតម្រូវឲ្យអនុវត្តក្រុមបញ្ជានៅក្នុងនោះ និងភ្ជាប់ឈ្មោះ វត្ថុ ទៅនឹងវត្ថុដែលវិធីពិសេសឈ្មោះ __enter__ បញ្ជូនមក។

ពាក្យបច្ចេកទេស

កញ្ចប់ 251
កន្សោមប្រមាណវិធី 27
កន្សោមផលិតករ 286
កម្រង 45
កម្រងថេរ 54
កម្រងអក្សរ 40
កម្រងអថេរ 63
កម្រងអថេររូបមន្ត 283
ការចម្លងទាំងស្រុង 326
ការចម្លងសើលើ 322
ការទទួលយកភាពមិនប្រក្រតី 261
ការបន្តថ្នាក់ 185
ការបន្តថ្នាក់រាងចតុកោណស្មើ 210
ការប្រមូលផ្តុំដំណឹង 136
ការបំបែកដំណឹង 135
ការយកថ្នាក់មកប្រើ 161
ការយកក្បួនមកប្រើ 122
កូនសោរ 73
កែ 220
ក្បាលក្បួន 119
ក្បួន 119
ក្បួនផលិតករ 276
ក្បួនមានស្រាប់ 153
ក្បួនអនាមិក 271
ក្រុមបញ្ជា 103
កំណត់ពន្យល់ 53
ចំនួនពិត 31
ចំនួនគត់ 25
ជាអប្បបរិមា 19
ឈ្មោះ 20
ដែនកំណត់សកល 140
ដែនកំណត់ 140
ដែនកំណត់ចារឹកក្នុង 142
ដែនកំណត់ចារឹកក្រៅ 142
ដែនកំណត់ដោយឡែក 141
ដែនកំណត់ទូទៅ 143
ដំណាង 128
ដំណឹង 128
ដំណឹងតាមដំណាង 131
ដំណឹងតាមលេខរៀង 130
ដំណឹងមានស្រាប់ 132
ដំណើរការ 17
ដំណោះស្រាយ 14
តក្កវត្ថុ 32
តម្លៃ 73
តាមរបៀបរចនានុក្រម 50

តួក្បួន ... 119
ថ្នាក់ ... 158
ថ្នាក់មេអរូបី ... 318
ថ្នាក់មានស្រាប់ ... 217
ថ្នាក់មេ ... 186
ថ្នាក់រង ... 186
ទិន្នន័យ ... 19
ទិន្នន័យគំរូ ... 159
ធាតុ ... 44
ធាតុគូ ... 73
បង្អួចបឋម ... 21
បច្ចេកទេសជាន់ខ្ពស់ ... 271
បញ្ហា ... 18
បញ្ហាចាត់តាំង ... 92
បណ្ណាល័យមជ្ឈឹម ... 256
បាំង ... 198
ប្រភេទ ... 25
ប្រមាណវិធីអត្តសញ្ញាណ ... 38
ប្រមាណវិធីកាត់ចម្លង ... 46
ប្រមាណវិធីគុន ... 27
ប្រមាណវិធីចែក ... 28
ប្រមាណវិធីចែកបន្ថយ ... 28
ប្រមាណវិធីចែកយកសំណល់ ... 28
ប្រមាណវិធីដក ... 27
ប្រមាណវិធីតក្កវិទ្យា ... 33
ប្រមាណវិធីនព្វន្ត ... 26
ប្រមាណវិធីបូក ... 27
ប្រមាណវិធីបូកបន្ត ... 43
ប្រមាណវិធីប្រជុំ ... 80
ប្រមាណវិធីប្រជុំធាតុខុសគ្នា ... 83
ប្រមាណវិធីប្រៀបធៀប ... 36
ប្រមាណវិធីរកធាតុ ... 52
ប្រមាណវិធីលេខរៀង ... 45
ប្រមាណវិធីប្រសព្វ ... 82
ប្រមាណសញ្ញា ... 26
ប្រមាណអង្គ ... 27
ផលិតករ ... 277
ផ្នែកទន់បកប្រែ ... 18
ពហុបន្តថ្នាក់ ... 202
ពាក្យពិសេស ... 20
ភាពមិនប្រក្រតី ... 258
ភ្ជាប់ ... 20
មិនអាចដោះដូរបាន ... 70
មោឃៈវត្ថុ ... 39
យន្តការបោសសម្អាត ... 23
រចនាករ ... 316
រោងជាង ... 15
លក្ខណៈប្រែរូប ... 199
លក្ខណៈសម្បត្តិ ... 312
លេខរៀង ... 44

វចនានុក្រម 73
វដ្តកម្ម 117
វដ្តកម្មជានិរន្តន៍ 110
វឌ្ឍនករ... 273
វត្ថុ 19
វត្ថុមានស្រាប់ 143
វិធី 159
វិធីប្រចាំថ្នាក់.. 315
វិធីពិសេស 287
វិធីឯកោ... 313
វិធីត្រូវគេបាំង 198
វិធីបាំងគេ.. 198
សតិ 19
សមមូល 35
សមាសវត្ថុ 44
សមាសវត្ថុអរូបី 281
សម្បត្តិពិសេស 305
សម្បត្តិឯកជន 310
សម្បត្តិថ្នាក់ 159
សម្បត្តិសាស្ត្រា....................................246
សម្បត្តិសិស្ស 177
សាស្ត្រា..244
សិស្ស...162
ស្ថាបនិក..166
ស្វ័យប្រមាណវិធី...................................96
ស្វ័យសេវា...320
សំណុំ...79
សំណុំមេ...86
សំណុំរង..86
អនុវត្ត...14
អាចដោះដូរបាន....................................70
អាណាព្យាបាល 326
អាទិភាពនៃប្រមាណសញ្ញា 29

ឯកសារស្រាវជ្រាវ

Mark Lutz, *Learning Python*, Sebastopol : O'Reilly Media, 2007

Tony Gasddis, *Starting Out with Python*, Boston : Addison Wesley, 2008

Magnus Lie Hetland, *Beginning Python : From Novice to Professional*, Apress, 2005

Wesley Chun, *Core Python Programming*, Prentice Hall, 2006

Stef Maruch, Aahz Maruch, *Python For Dummies*, For Dummies, 2006

Mark Pilgrim, *Dive Into Python*, Apress, 2004

Michael Dawson, *Python Programming for the Absolute Beginner*, Course Technology, 2003

Peter C. Norton, Alex Samuel, Dave Aitel, Eric Foster-Johnson, *Beginning Python*, Wrox, 2005

Dave Brueck, Stephen Tanner, *Python 2.1 Bible*, Wiley, 2001

Michael H Goldwasser, David Letscher, *Object-Oriented Programming in Python*, Prentice Hall, 2007

Alex Martelli, Anna Ravenscroft, David Ascher, *Python Cookbook*, O'Reilly Media, 2005

Alex Martelli, *Python in a Nutshell*, O'Reilli Media, 2006

David M. Beazley, *Python Essential Reference*, Sams, 2006