

Unit 21.

Class

ຈຸດປະສົງການຮຽນຮູ້

- ✓ ເຊົ້າໃຈແນວຄວາມຄິດພື້ນຖານຂອງການຂຽນໂປຣແກຣມແບບວັດຖຸ (object-oriented programming) ແລະ ຄວາມຈຳເປັນຂອງມັນ.
- ✓ ເຊົ້າໃຈແນວຄວາມຄິດພື້ນຖານຂອງ ການຫຼຸມຫໍ່(encapsulation) ແລະ ສ້າງໂຄດສໍາຫຼັບການຫຼຸມຫໍ່(encapsulation) ເພື່ອຄວາມສະຖານຂອງການເຮັດວຽກຂອງໂປຣແກຣມ.
- ✓ ເຊົ້າໃຈແນວຄວາມຄິດຂອງ class instances, objects ແລະ ສະແດງໃຫ້ເຫັນເຖິງຄວາມແຕກຕ່າງ.
- ✓ ການສ້າງ class, ເຊົ້າໃຈວິທີການສ້າງ objects ແລະ ການສ້າງ objects.
- ✓ ເຊົ້າໃຈແນວຄວາມຄິດພື້ນຖານຂອງ constructor ແລະ ນໍາໃຊ້ self parameter ຂອງ class ເປັນໂຕສ້າງ.
- ✓ ເຊົ້າໃຈໂຄງສ້າງຂອງ method ແລະ ວິທີການກຳນົດ.
- ✓ ເຊົ້າໃຈແນວຄວາມຄິດພື້ນຖານຂອງການອ້າງອີງແລະ ອະທິບາຍກົນໄກພາຍໃນຂອງການກຳນົດ class instances ໃຫ້ກັບຕົວປິ່ງ.
- ✓ ກຳນົດ class ໃໝ່ໜໍ້ສືບທອດມາຈາກ class ທີ່ມີຢູ່ກ່ອນແລ້ວ.

ພາບລວມການຮຽນຮູ້

- ✓ ສຶກສາການຂຽນໂປຣແກຣມແບບວັດຖຸ (object-oriented programming) ແລະ ເຂົ້າໃຈຄວາມຈຳເປັນຂອງມັນ.
- ✓ ເຂົ້າໃຈແນວຄວາມຄິດພື້ນຖານຂອງການຫຼຸມທີ່ (encapsulation) ແລະ ນຳໃຊ້ແນວຄວາມຄິດນີ້ເຂົ້າໃນການຂຽນໂຄດສໍາລັບ ຄວາມສະຖານຂອງ ການເຮັດວຽກຂອງໂປຣແກຣມ.
- ✓ ຮຽນຮູ້ແນວຄວາມຄິດພື້ນຖານຂອງ objects ແລະ class instances.
- ✓ ການສ້າງ class, ເຂົ້າໃຈວິທີການສ້າງ objects ແລະ ການສ້າງ objects.
- ✓ ກຳນົດ class object ໃຫ້ກັບຕົວປ່ຽນໂດຍ assigning operator. ຮຽນຮູ້ແນວຄວາມຄິດຂອງການອ້າງອີງທີ່ຖືກນຳມາໃຊ້.
- ✓ ຮຽນຮູ້ວິທີກຳນົດ class ໃຫ່ມທີ່ສືບທອດມາຈາກ class ຫີ່ມີຢູ່ກ່ອນເລື່ອ.

ແນວຄວາມຄິດທີ່ຈະຕ້ອງຮູ້ຈາກບົດຮຽນທີ່ຜ່ານມາ

- ✓ ການສ້າງfunction ທີ່ຊ້ອນກັນແລະເອັນໃຊ້ function ພາຍໃນ function.
- ✓ ການນຳໃຊ້ nonlocal keyword, ເຊິ່ງຊອກຫາ local keyword ຫີ່ໄກ້ທີ່ສຸດ, ແລະ gobal keyword.
- ✓ ນຳໃຊ້ indentation ແລະ ຄໍາສົ່ງປລັອກ.

Keywords

Object

Class

Instance

Constructor

self Parameter

ການສືບທອດຄຸນລັກສະນະ
(Inheritance)

Mission

1. ບັນຫາໃນໄລກຄວາມເປັນຈີງ

1.1. ທະນາຄານ ແລະ ລະບົບຄອມພິວຕີ



- ▶ ທະນາຄານເປັນສະຖາບັນການເງິນຫຼັກຂອງສັງຄົມທີ່ບໍລິຫານຈັດການ ການກຸ້ມືມເງິນ, ການຝາກເງິນ ແລະ ອື່ນໆ.
- ▶ ທະນາຄານໃນທຸກມື້ນີ້ໃຊ້ລະບົບຄອມພິວຕີອັດຕະໂນມັດສໍາລັບວຽກງານສ່ວນໃຫຍ່ເນື່ອງຈາກການພັດທະນາຂອງ IT.
- ▶ ທະນາຄານຕ່າງໆໃນປັດຈຸບັນໄດ້ກຳລັງຂະໜາຍທຸລະກິດຈາກການຝາກເງິນ ແລະ ການກຸ້ມືມເງິນໄປສູ່ການປະກັນໄຟ, ຫຼັກຊັບ ແລະ ທະນາຄານເອກະຊົນ. ເນື່ອງຈາກທຸລະກິດເຫຼົ່ານີ້ມີຄວາມສັບຊອນຫຼາຍເຂັ້ມ, ທະນາຄານຈຶ່ງປະສົບບັນຫາໃນການໃຊ້ງານລະບົບຄອມພິວຕີ.
- ▶ ເພື່ອໃຫ້ສອດຄ່ອງກັບຄວາມຕ້ອງການຂອງສັງຄົມ, ພວກເຮົາຈະສ້າງໂຄດສໍາລັບລະບົບຄອມພິວຕີຂອງທະນາຄານ.
- ▶ ການຮຽນຮູ້ເຕັກນິກການສ້າງ class ຈະເປັນຂັ້ນຕອນທຳອິດເພື່ອການຂຽນໂປຣແກຣມລະດັບສູງສໍາລັບຄວາມຕ້ອງການທີ່ສັບຊອນ.

1. ບັນຫາໃນໄລກຄວາມເປັນຈິງ

1.2. ປະຫວັດຄວາມເປັນມາຂອງທະນາຄານ



- ▶ ທະນາຄານແມ່ນສະຖາບັນການເງິນທີ່ຈໍາເປັນ ຫຼືມີການຮັບຝາກເງິນ ແລະ ໃຫ້ການກຸ້ຍືມເງິນ.
- ▶ ການທະນາຄານແມ່ນເລີ່ມຕົ້ນໂດຍພໍ່ຄ້າໃນສະໄໝມບູຮານຕັ້ງແຕ່ສະໄໝມ ບາບີໂລນ ແລະ ເມ ໂຊໂປຕາເມບ. ເປັນທີ່ຮູ້ຈັກກັນວ່າຊາວນ ແລະ ພໍ່ຄ້າທີ່ຈັດຫາສະບຽງລະຫວ່າງເມືອງຕ່າງໆ ມີ ການໃຫ້ກຸ້ຍືມເມັດຜົນລະບຸກ. ຄາດຄະເນວ່າແມ່ນອາດຈະປະມານ 2000 ປີກ່ອນຄືສະ ສັກກາຣາດ.
- ▶ ຕໍ່ມາປະເທດເກົຮັກບູຮານ ແລະ ຈັກກະພັດໂຮມນຂະຫຍາຍທຸລະກິດດັ່ງກ່າວ.
- ▶ ນອກຈາກນັ້ນ, ຍັງມີການຊຸດຄົ້ນຫຼັກຖານທາງໂບຮານຄະດີກ່ຽວກັບການກຸ້ຍືມໃນສະໄໝມ ບູຮານຂອງຈິນແລະອິນເດຍ.

2. Mission

2.1. ປິດເຜີກຫັດການຮັດບັນຊີທະນາຄານ

- | ໃນປິດເຜີກຫັດນີ້, ພວກເຮົາຈະສ້າງ class ທີ່ມີຊື່ວ່າ BankAccount ແລະ ດຳເນີນການໃຊ້ function ຕ່າງໆໂດຍໃຊ້ຄຸນສົມບັດຂອງ OOP (object-oriented programming).
- | ສ້າງ class ມີຊື່ວ່າ BankAccount, ສ້າງບັນຊີ account1 ທີ່ມີຂໍ້ມູນ ເຊັ່ນ: ຊື່ລູກຄ້າ 'David', ແລະ ຈຳນວນເງິນຝາກ 2000 ວອນ.
- | ໃຫ້ພິມຂໍ້ມູນບັນຊີ, ຈາກນັ້ນຖອນເງິນ 500 ວອນ ແລະ ພິມຂໍ້ມູນບັນຊີດັ່ງກ່າວອອກມາ.
- | ຖອນເງິນ 5000 ວອນ ຈາກບັນຊີດັ່ງກ່າວ, ຫຼັງຈາກນັ້ນໃຫ້ຂຽນໂຄດພິມຍອດເງິນໃນບັນຊີ ແລະ ຂໍຄວາມທີ່ບອກວ່າການຖອນເງິນຖືກປະຕິເສດ.
- | ໂຄດຂອງປິດເຜີກຫັດນີ້ຮັດວຽກໂດຍການສິ່ງຜ່ານ methods ໄປຫາ objects, ເຊິ່ງແຕກຕ່າງຈາກຂະບວນການຂອງ methods .

```
account1 = BankAccount("David", "1234-0001")
print(account1)
account1.deposit(2000)
print(account1)
account1.withdraw(500)
print(account1)
account1.withdraw(5000)
```

2. Mission

2.1. ຜົນການປະຕິບັດပິດແຜာຫັດການເຮັດບັນຊີທະນາຄານ

```
1 class BankAccount:
2     """A class example in python that models a bank account"""
3     def __init__(self, name, account_num, balance = 0):
4         self.name = name
5         self.account_num = account_num
6         self.balance = balance
7
8     def get_name():
9         return self.name
10
11    def get_account_num():
12        return self.account_num
13
14    def get_balance():
15        return self.balance
16
17    def deposit(self, amount):
18        self.balance += amount
19        print('{} won has been deposited. The balance is {} won.'.format(amount,\n20                                         self.balance))
21        return self.balance
22
23    def withdraw(self, amount):
24        if self.balance - amount > 0 :
25            self.balance -= amount
26        else:
27            print('The account balance is {} won. This is smaller than the requested amount of withdrawal' \
28                  '{} won.'.format(self.balance, amount))
```

2. Mission

2.1. ຜົນການປະຕິບັດပິດແຜာຫຼັດການເຮັດບັນຊີທະນາຄານ

```
30 def __str__(self):
31     return 'The balance of {}'s account {} is {:.2} won.'.format(self.name,\n32                                         self.account_num, self.balance)\n33\n34 account1 = BankAccount("David", "1234-0001")\n35 print(account1)\n36 account1.deposit(2000)\n37 print(account1)\n38 account1.withdraw(500)\n39 print(account1)\n40 account1.withdraw(5000)
```

The balance of David's account 1234-0001 is 0 won.
2000 won has been deposited. The balance is 2000 won.
The balance of David's account 1234-0001 is 2000 won.
The balance of David's account 1234-0001 is 1500 won.
The account balance is 1500 won. This is smaller than the requested amount of withdrawal 5000 won.

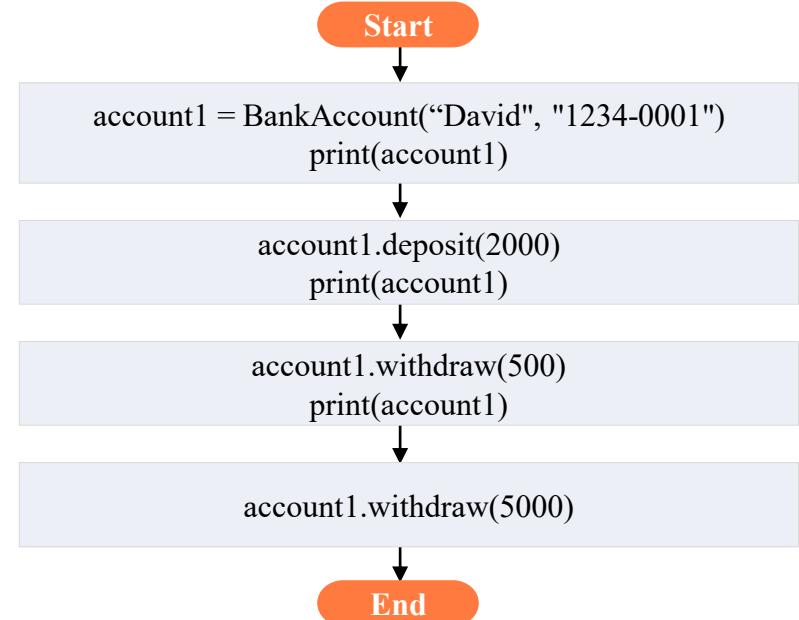
2. Mission

2.3. របៀបការងារនៃកម្មវិធី

Pseudocode

- [1] តើមពី
- [2] ស្វែងរកជួយរបៀបបង្កើតឱ្យលើកម្មវិធី
មុនបំផុត. ឲ្យរាយការណ៍, ឯកសារនិងរបៀបបង្កើតឱ្យលើកម្មវិធី។
- [3] សម្រាប់រាយការណ៍ ឯកសារនិងរបៀបបង្កើតឱ្យលើកម្មវិធី
របៀបបង្កើតឱ្យលើកម្មវិធី។
- [4] ទូទាត់រាយការណ៍ ឯកសារនិងរបៀបបង្កើតឱ្យលើកម្មវិធី។
- [5] ទូទាត់រាយការណ៍ ឯកសារនិងរបៀបបង្កើតឱ្យលើកម្មវិធី។
- [6] ចំណាំ

Flowchart



2. Mission

2.4. ແຜນຜັງຈຸດປະສົງຂອງບັນຊີທະນາຄານ

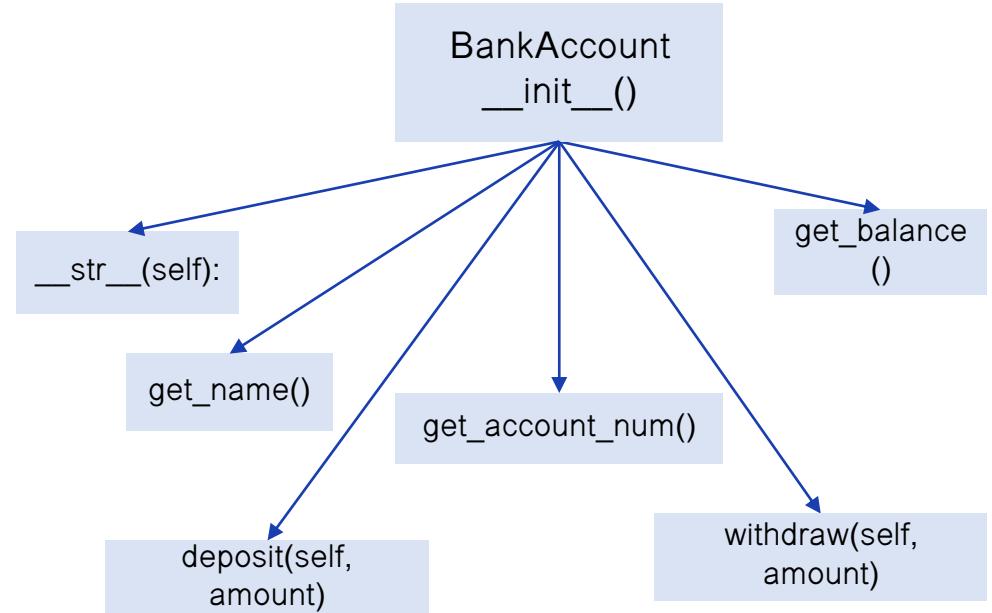
- | ມາຮອດຕອນນີ້, ພວກເຮົາໄດ້ອະທິບາຍການເຮັດວຽກຂອງໂຄດດ້ວຍ flowchart.
 - | ແນວໃດກໍ່ຕາມວິທີການຂຽນໂປຣແກຣມຕາມຂັ້ນຕອນທີ່ອີງຕາມການໃຫ້ຂອງຂໍ້ມູນແມ່ນບໍ່ເຫມາະສົມສໍາລັບການພັດທະນາຊອບແວຂະໜາດໃຫຍ່ທີ່ສະລັບສັບຊື່ອນ.
 - | ການພັດທະນາປະສິດທິພາບຮາດແວຢ່າງວ່ອງໄວຮຽກຮ້ອງໃຫ້ຕ້ອງໃຊ້ຊອບແວຂະໜາດໃຫຍ່ ແລະ ສັບຊ້ອນໜ້າຢ. ຢ່າງໃດກໍ່ຕາມ, ວິທີການຂຽນໂປຣແກຣມຕາມຂັ້ນຕອນທີ່ມີຢູ່ນີ້ບໍ່ສາມາເຮັດວຽກຂອງຮັບການພັດທະນາຂອງຊອບແວເຫຼົານີ້ໄດ້.
 - | ນີ້ແມ່ນເຫດຜົນທີ່ວ່າແຜນຜັງທີ່ສະແດງຢູ່ໃນໜັນຕໍ່ໄປໄດ້ຖືກສ້າງຂຶ້ນ.
 - | Class diagram ຂອງແຜນຜັງສະບັບນີ້ປະກອບດ້ວຍ attributes ແລະ methods ຂອງ Class .
 - | ນີ້ແມ່ນຫົວຂໍ້ຫຼັກຂອງບົດນີ້.

2. Mission

2.4. ထောက်ပွဲအုပ်စုနည်းပညာ

| Bank Account class diagram

class name : BankAccount
BankAccount attributes
self.name : name of the account holder
self.account_num : account number
self.balance : balance
BankAccount methods
def get_name(): returns the name of the account holder.
def get_account_num(): returns the account number.
def get_balance(): returns the balance.
def deposit(self, amount): returns the sum of account balance and deposit amount.
def withdraw(self, amount): returns the balance subtracting the withdrawal amount. If the withdrawal amount is bigger than the balance, it prints that the withdrawal is bigger.
def __str__(self): prints the name, account, and balance.



2. Mission

2.5. Code ဆုတ္တာယခ်ပါတန္ဒေသပံ့ပိုးများ #1

```
1 class BankAccount:  
2     """A class example in python that models a bank account"""  
3     def __init__(self, name, account_num, balance = 0):  
4         self.name = name  
5         self.account_num = account_num  
6         self.balance = balance  
7  
8     def get_name():  
9         return self.name  
10  
11    def get_account_num():  
12        return self.account_num  
13  
14    def get_balance():  
15        return self.balance  
16  
17    def deposit(self, amount):  
18        self.balance += amount  
19        print('{} won has been deposited. The balance is {} won.'.format(amount,\n              self.balance))  
20  
21    def withdraw(self, amount):  
22        if self.balance - amount > 0 :  
23            self.balance -= amount  
24        else:
```

2. Mission

2.5. Code ဆုတ္တာယခုခြင်ပါတန္ထေရာပို့မည့်အကျဉ်းချုပ် #1

```
27     print('The account balance is {} won. This is smaller than the requested amount of withdrawal'\n28         '{} won.'.format(self.balance, amount))\n29\n30     def __str__(self):\n31         return 'The balance of {}'s account {} is {:.2f} won.'.format(self.name,\n32                                         self.account_num, self.balance)\n33\n34 account1 = BankAccount("David", "1234-0001")\n35 print(account1)\n36 account1.deposit(2000)\n37 print(account1)\n38 account1.withdraw(500)\n39 print(account1)\n40 account1.withdraw(5000)
```

| Key concept

1. ນິຍາມ ແລະ ແນວຄວາມຄິດຂອງ Object

1.1. ການຂຽນໂປຣແກຣມ ແລະ object

| ພວກເຮົາຈະສຶກສາເຈົ້າຈຶ່ງດໍານາ Object.

```
1 animals = ['lion', 'tiger', 'cat', 'dog']
2 animals.sort()
3 animals

['cat', 'dog', 'lion', 'tiger']
```

```
1 animals.append('rabbit')
2 animals

['cat', 'dog', 'lion', 'tiger', 'rabbit']
```

```
1 animals.reverse()
2 animals

['rabbit', 'tiger', 'lion', 'dog', 'cat']
```

- ▶ ບັນຊີລາຍຊື່ສັດແມ່ນ Object ທີ່ປະກອບມີ 'lion', 'tiger', 'cat', 'dog' ເປັນອົງປະກອບ (attributes).
- ▶ ນອກຈາກນີ້ຢູ່ມີການຈັດລຽງ, ເພີ່ມເຂົ້າ, ຢ້ອນກັບ ປະກອບເປັນ functions (methods) (ວິທີການເອີ້ນໃຊ້ ໂດຍເຄື່ອງໝາຍ . (ຈ້າເມັດ))
 - Functions ຕ່າງໆເປັນຂອງ animal object ແມ່ນເອີ້ນວ່າ methods.
- ▶ Object
 - ດັ່ງນັ້ນ, ອົງປະກອບຕ່າງໆໃນລະບົບຄອມພິວເຕີ່ມີ attributes ແລະ methods ເອີ້ນວ່າ Object .

ຄືກໜຶ່ງຂັ້ນຕອນ

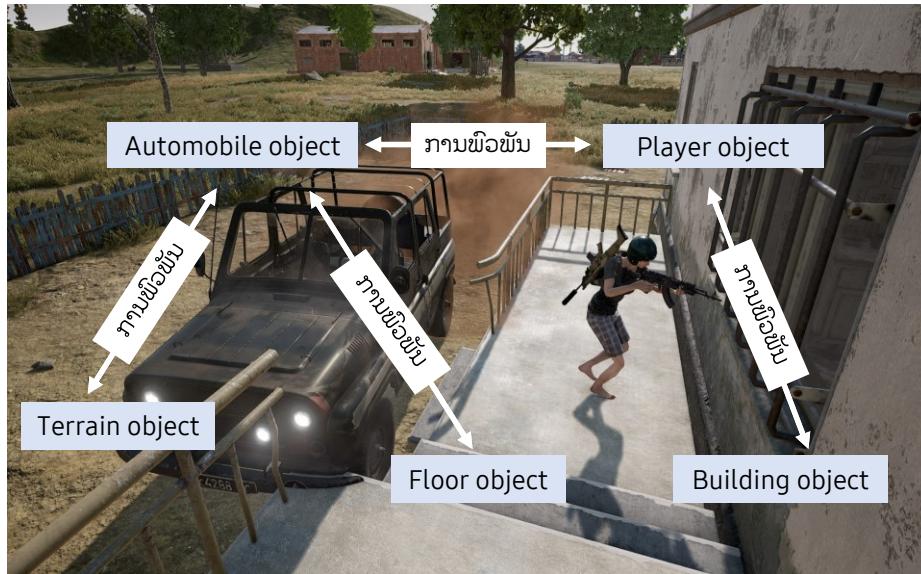
- | ຂ້າງລຸ່ມນີ້ແມ່ນການປຽບທຽບຂອງການຊຽນໂປຣແກຣມແບບວັດຖຸ ແລະ ການຊຽນໂປຣແກຣມຕາມຂັ້ນຕອນ.
 - | ການຊຽນໂປຣແກຣມແບບວັດຖຸ (OOP)
 - ▶ ເຕັກນິກທີ່ສ້າງແບບຈໍາລອງໂປຣແກຣມໃຫ້ຄ້າຍຄືກັບໂລກຄວາມເປັນຈິງ
 - ▶ ສະແດງວຽກງານທີ່ດໍາເນີນການດ້ວຍຄອມພິວເຕີເປັນ *interactions among objects*.
 - ▶ ແນວດວາມຄິດທີ່ມີຈຸດມຸ່ງໝາຍຕັ້ງໃຈຈະພັດທະນາຊອບແວໂດຍຊຸດຂອງ classes ຫຼື objects.
 - ▶ ຖືກນຳມາໃຊ້ໂດຍພາສາໂປຣແກຣມທີ່ຄືນນິຍົມໃຊ້ຊຽນໂຄດຢ່າງຫຼວງຫຼາຍເຊັ່ນ Java, Python, C++, C#, Swift ແລະ ອື່ນໆ.
 - | ການຊຽນໂປຣແກຣມຕາມຂັ້ນຕອນ
 - ▶ ວິທີການທີ່ດໍາເນີນການໂດຍການເຮັ້ນ functions ແລະ models ຕາມລຳດັບຄໍາສັ່ງໃນການແກ້ບັນຫາ.
 - ▶ ບືກນຳໃຊ້ໂດຍພາສາການຊຽນໂປຣແກຣມແບບດັ່ງເດີມເຊັ່ນ C, Fortran, Basic ແລະ ອື່ນໆ.
 - ▶ ບໍ່ສາມາດແກ້ໄຂບັນຫາກ່ຽວກັບລະບົບຂອງອີງປະກອບດ້ານການຟິກຕ່າງໆເຊັ່ນ ລະບົບການໄຕ້ຕອບກັບຜູ້ໃຊ້ໂດຍການຟິກ(GUI) .

1. ນິຍາມແລະ ແນວຄວາມຄິດຂອງ Object

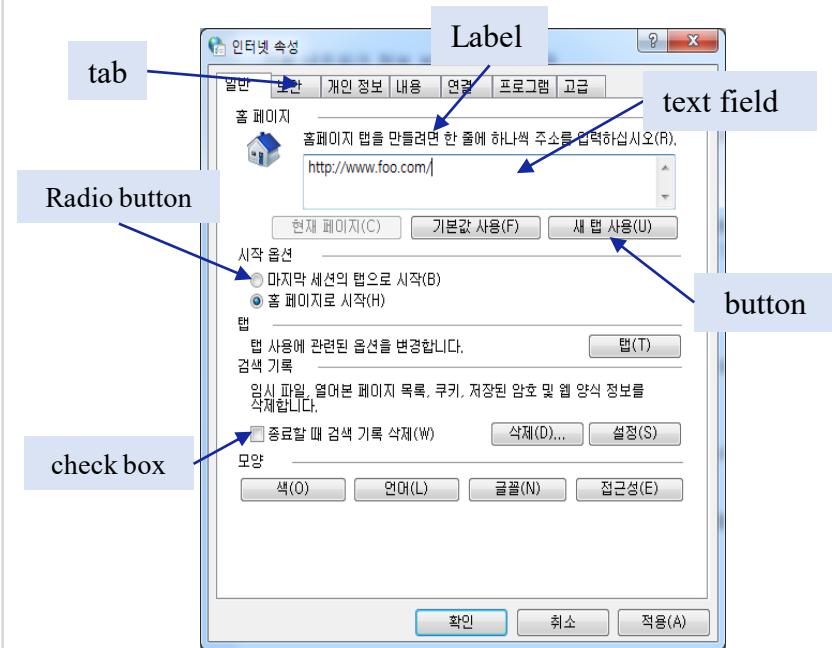
1.1. ການຂຽນໂປຣແກຣມ ແລະ object

| ແນວຄວາມຄິດຂອງ object : ອີງປະກອບຈໍານວນຫຼາຍອັນທີມີການພົວພັນກັນໃນເກມໄດ້ທີ່ກອ້ັນວ່າ objects. ອີງປະກອບຂອງການໂຕຕອບກັບຜູ້ໃຊ້ທີ່ໃຊ້ກາພືກກຳເອັນວ່າ objects.

Interactions among objects ในເກມ



Objects ໃນການໂຕຕອບກັບຜູ້ໃຊ້ໂດຍໃຊ້ກາພືກ



1. ນິຍາມ ແລະ ແນວຄວາມຄິດຂອງ Object

1.2. ການປຽບທຽບການຂຽນໂປຣແກຣມຕາມຂັ້ນຕອນ ແລະ ການຂຽນໂປຣແກຣມແບບວັດຖຸ

| ການຂຽນໂປຣແກຣມຕາມຂັ້ນຕອນ

- ▶ ຕ້ອງການຈຳນວນລູກສອນ (arrow) ຢ່າງຫຼວງຫຼາຍ ແລະ ເອັນໃຊ້ຝັງຊັ້ນຫຼາຍຖ້າຫາກຂຶ້ນ ແລະ ພັງຊັ້ນຂະຫຍາຍແລະເພີ່ມຂຶ້ນ.
- ▶ ຍາກທີ່ຈະຈັດການໂຄງການຂະໜາດໃຫຍ່.

| ການຂຽນໂປຣແກຣມແບບວັດຖຸ (OOP)

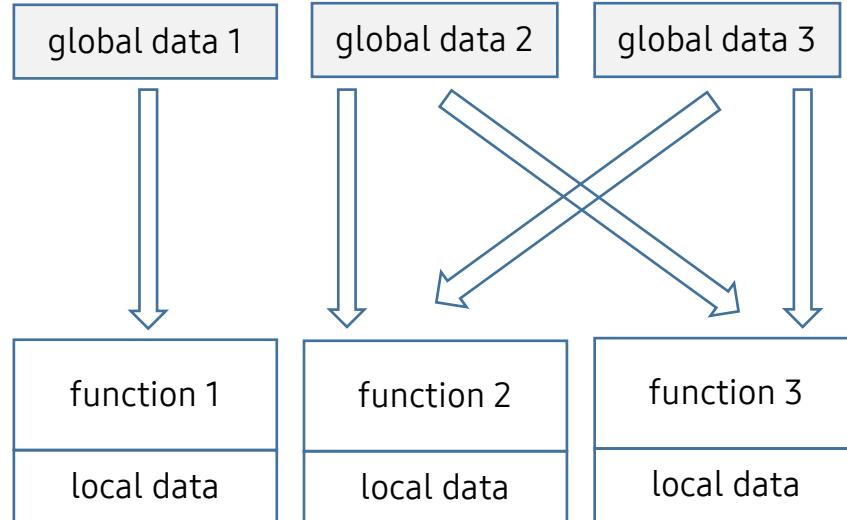
- ▶ ວິທີການສ້າງ objects ໂດຍ classes ທີ່ສ້າງຂຶ້ນ ຢ່າງມີປະສິດທິຜົນ.
- ▶ Classes ຖືກສ້າງໃຫ້ມີ actions ແລະ attributes . OOP ນຳໃຊ້ Classes ເຫຼື້ນີ້ກັບການຂຽນໂປຣແກຣມໂດຍການສ້າງ objects ທີ່ມີການໄຕຕອບ.

| OOP ຕ້ອງການຄ່າໃຊ້ຈ່າຍໃນການບໍາລຸງຮັກສາພຽງເລັກນ້ອຍໃນກໍລະນີຂອງການພັດທະນາ ຫຼື ການອັບເດດຊອບແວ. ການຂຽນໂປຣແກຣມໃນປັດຈຸບັນສ່ວນຫຼາຍມັກນຳໃຊ້ OOP.

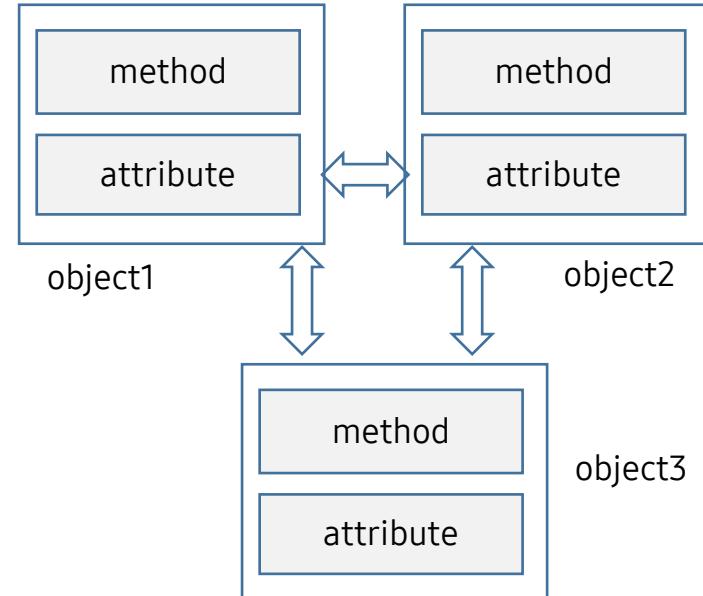
1. ນິຍາມ ແລະ ແນວຄວາມຄິດຂອງ Object

1.2. ການປຽບທຽບການຂຽນໂປຣແກຣມຕາມຂັ້ນຕອນ ແລະ ການຂຽນໂປຣແກຣມແບບວັດຖຸ

| ຂ້າງລຸ່ມນີ້ແມ່ນແຜນວາດການເຮັດວຽກຂອງຂໍ້ມູນ ແລະ function (method) ຂອງການຂຽນໂປຣແກຣມແບບຂັ້ນຕອນ ແລະ ການຂຽນໂປຣແກຣມແບບວັດຖຸ.



ການຂຽນໂປຣແກຣມແບບຂັ້ນຕອນ

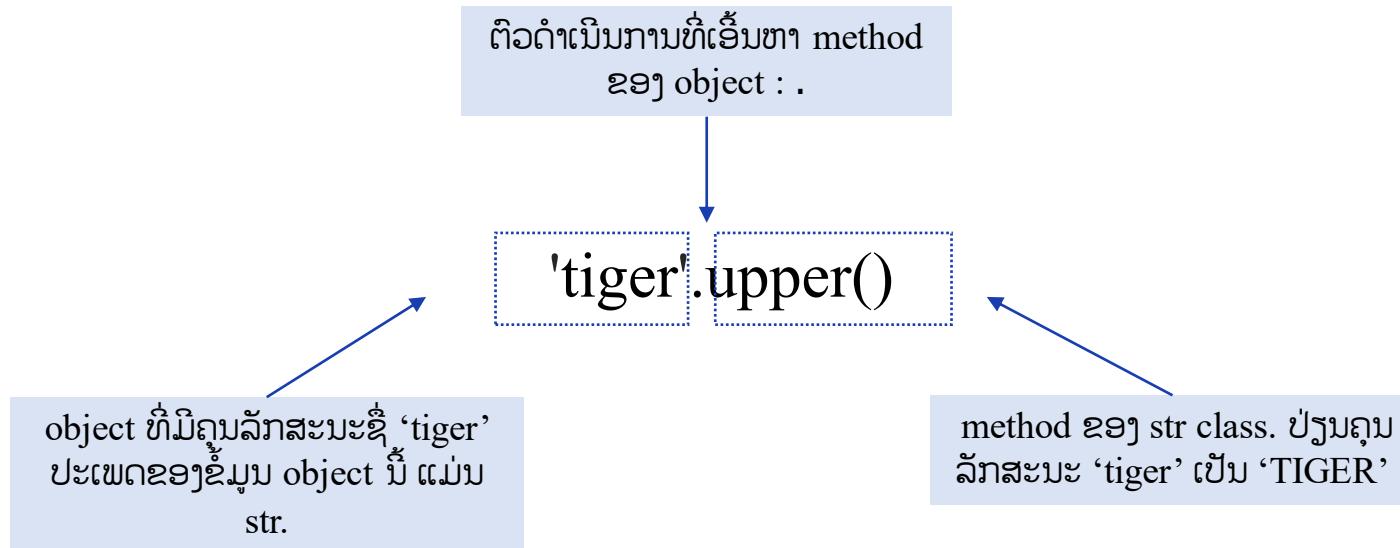


ການຂຽນໂປຣແກຣມແບບວັດຖຸ.

1. ນິຍາມ ແລະ ແນວຄວາມຄິດຂອງ Object

1.3. Syntax ທີ່ໃຊ້ເອັນ Python's objects ແລະ methods

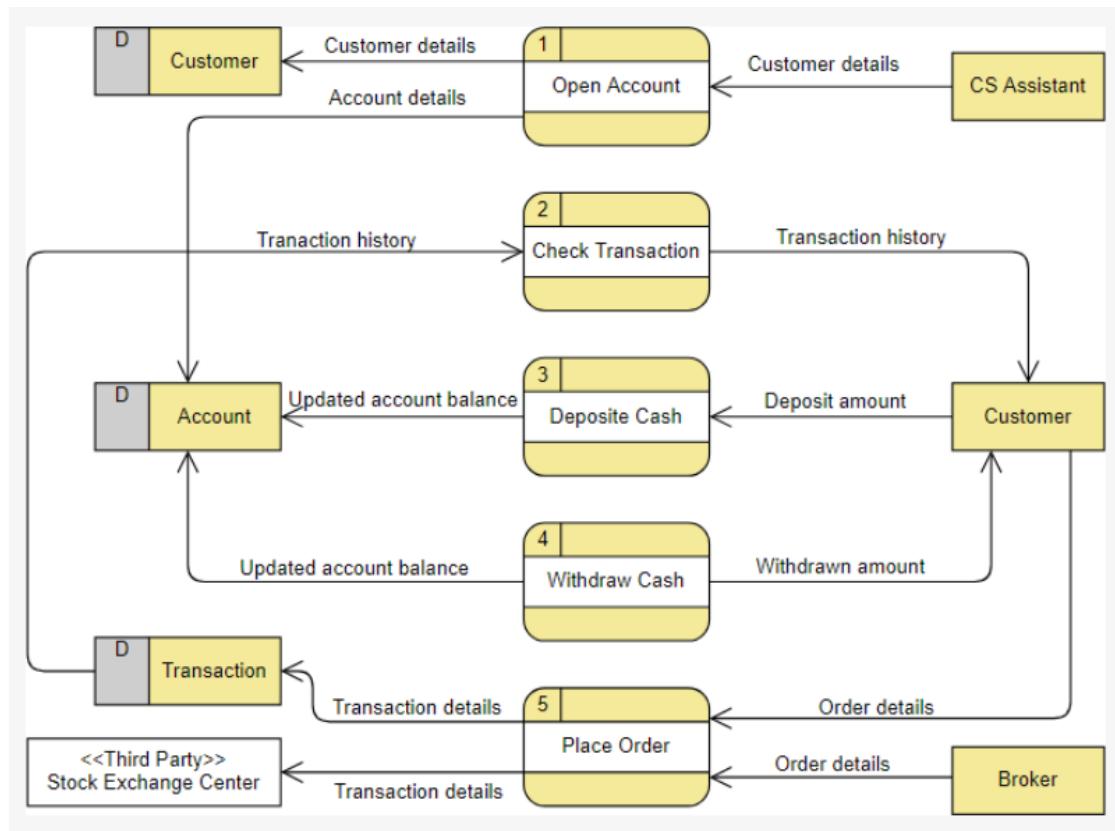
| ໃຊ້ດໍາເນີນການ .(ຈຳເມັດ) ເພື່ອເອັນ methods ສໍາລັບ object.



```
1 'tiger'.upper()  
'TIGER'
```

1. ນິຍາມ ແລະ ແນວຄວາມຄິດຂອງ Object

1.4. ຂໍ້ຈໍາກັດຂອງ flowchart



- ຖ້າຊອບແວທີ່ສັບສົນກິນໄປ, ການສະແດງຜົນດ້ວຍ flowchart ມີນຈະບໍ່ສະດວກ.
- ເພື່ອແກ້ໄຂບັນຫາດັ່ງກ່າວ, ສາມາດນຳໃຊ້ແຜນວາດການໃຫ້ຂອງຂໍ້ມູນ, ດັ່ງທີ່ສະແດງຢູ່ເບື້ອງຊ້າຍ, ເພື່ອອະທິບາຍການຮັດວຽກຂອງຂໍ້ມູນ.

<https://developpaper.com/how-to-create-a-data-flow-diagram-dfd-online/>

💡 ອີກໜຶ່ງຂັ້ນຕອນ

- | ທຸກໆຢ່າງໃນ Python ກໍາຄື object.
- | ສີກສາລະຫັດຂ້າງລຸ່ມນີ້.
 - ▶ 'tiger'.upper()
 - ▶ animals.append('rabbit')
- | animals.append('rabbit')
- | string(ຂໍ້ມູນປະເພດຂໍ້ວາມ) ‘tiger’ ແລະ animals ສ່ວນຕ່າງໆດໍາເນີນການໂດຍໃຊ້ methods ເຊັ່ນ upper ແລະ append. ອີງປະກອບດັ່ງກ່າວຂອງ ການຂຽນໂປຣແກຣມເອັນວ່າ objects.
- | Python ແມ່ນພາສາການຂຽນໂປຣແກຣມແບບວັດຖຸ ແລະ ຂໍ້ມູນທັງໝົດທີ່ໃຊ້ໃນ Python ແມ່ນ objects.
- | n, 100 ຫຼື 200, ທີ່ຖືກນຳໃຊ້ໃນສົມຜົນ ຫຼື ການເຮັດວຽກຂອງໂຄດ, ກໍຍັງແມ່ນ objects ຫຼື ບໍ່ ?
 - ▶ $n = 100 + 200$
- | ແມ່ນແລ້ວ, ໂຄດທາງເທິງຄືກັບໂຄດຕໍ່ໄປນີ້.
 - ▶ $n = (100).__add__(200)$
- | ນັ້ນຄືໃນໂຄດທາງເທິງ, ວັດຖຸ integer 100 ເພີ່ມວັດຖຸ 200 ໂດຍໃຊ້ methode __add__
- | ວັດຖຸນີ້ທີ່ກາເຂົ້າເຖິງໂດຍຕົວປ່ຽນ n ຈາກມຸມມອງນີ້, method ເອັນໃຊ້ code ແລະ ຂັ້ນຕອນການດໍາເນີນການຂອງຕົວດໍາເນີນການ add ແມ່ນຄ້າຍຄືກັນກັບ animals.append('rabbit').

2. ຕົວຢ່າງຂອງການນຳໃຊ້ Class ໃນບົດຮຽນທີ່ຜ່ານມາ

2.1. ຕົວຢ່າງຂອງການນຳໃຊ້ classes ແລະ objects

- | ຈຸດແຂງຂອງ OOP ແມ່ນການກຳນົດວິທີການທີ່ມີປະສິດທິພາບ ແລະ ນຳໃຊ້ວິທີການເຫຼື່ອນັ້ນໄດ້ຢ່າງຈ່າຍດາຍ.
- | ໃນໂຄດຕໍ່ໄປນີ້ ຈະແຍກ string s ຜ່ານ animal.pop().
- | Method ຕ່າງໆເຊັ່ນ upper, find ແມ່ນຖືກກຳນົດໄວ້ໃນ str data types.
- | ເຊັ່ນດຽວກັນ, ເຮົາສາມາດນຳໃຊ້ວິທີການທີ່ຫລາກຫລາຍໃນ OOP.

```
1 s = animals.pop()  
2 s  
'cat'
```

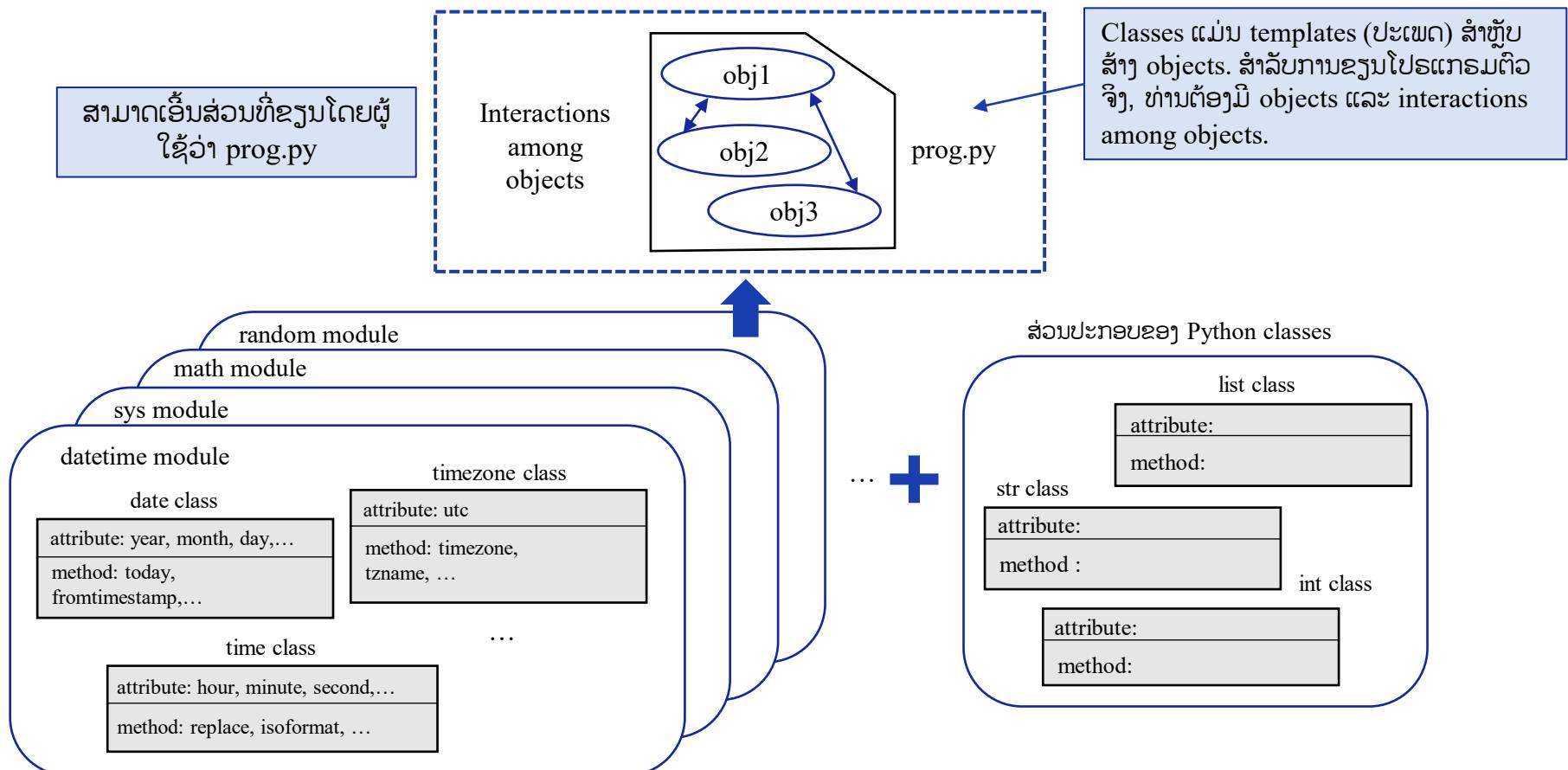
```
1 s.upper()  
'CAT'
```

```
1 s.find('a')
```

1

2. ຕົວຢ່າງຂອງການນຳໃຊ້ Class ໃນບົດຮຽນທີ່ຜ່ານມາ

2.2. ບັນດາ functions ແລະ classes ຕ່າງໆມື່ລວມຍູ້ໃນ modules



2. ຕົວຢ່າງຂອງການນຳໃຊ້ Class ໃນບົດຮຽນທີ່ຜ່ານມາ

2.3. ຫນ້າທີ່ສະເພາະຂອງຝັງຊັນ type ແລະ ຝັງຊັນ id

- | ຝັງຊັນ type ຈະສື່ປະເພດຂໍ້ມູນຂອງວັດຖຸ.
- | ແຕ່ລະ object, ເມື່ອສ້າງຈະມີຄ່າ id ຂອງຕົນເອງ. ສາມາດກວດສອບ id ໂດຍໃຊ້ຝັງຊັນ id.

```
1 animals = ['lion', 'tiger', 'cat', 'dog']
2 type(animals)
```

list

```
1 id(animals)
```

2911697529408

```
1 s = 'tiger'
2 type(s)
```

str

```
1 id(s)
```

2911697427632

2. ຕົວຢ່າງຂອງການນຳໃຊ້ Class ໃນບົດຮຽນທີ່ຜ່ານມາ

2.3. ຫນ້າທີ່ສະເພາະຂອງຝັງຊັນ type ແລະ ຝັງຊັນ id

| ໂດຍການເຮັດວຽກຂອງຝັງຊັນ id, ເຮົາສາມາດກວດສອບວ່າຕົວປຸງນຳລັງອ້າງອີງພື້ນທີ່ທ່ານ່ວຍຄວາມຈຳດູວກັນຫຼືໆບໍ່.

- ▶ ແຕ່ລະ object ມີຄ່າ id ຂອງຕົນເອງ.
- ▶ ຖ້າ object ດີກັນ, id ກໍດີກັນ.
- ▶ ບາງຄັ້ງເຮົາສາມາດອ້າງອີງ object ດຽວກັນໂດຍຕົວປຸງທີ່ມີຊື່ແຕກຕ່າງກັນ. ສາມາດກວດເບິ່ງດ້ວຍຝັງຊັນ id.

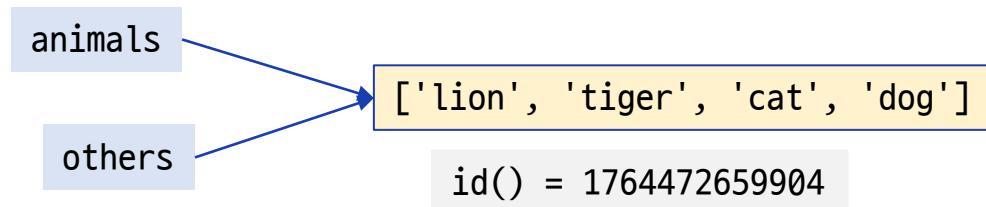
```
1 animals = ['lion', 'tiger', 'cat', 'dog']
2 others = animals
```

```
1 id(animals)
```

1764472659904

```
1 id(others)
```

1764472659904



2. ຕົວຢ່າງຂອງການນຳໃຊ້ Class ໃນບົດຮຽນທີ່ຜ່ານມາ

2.4. ຕົວຢ່າງຂອງ str class ແລະ methods ຂອງມັນ

| ຕົວຢ່າງ : str class ມີ methods ຕ່າງໆຕາມທີ່ສະແດງຂ້າງລຸ່ມນີ້.

- upper
- lower
- capitalize
- startswith
- strip
- find
- split
- join
- casefold
- center
- count
- endswith
- format
- index
- isalnum
- isalpha
- isdecimal
- islower
-

str class ພຽງແຕ່ສະເໜີ methods ຈຳນວນຂະໜາດໃຫຍ່.
ໂດຍ Python ຕອບສະໜອງຕໍ່ default, int, list, tuple, dic,
date, time ແລະ classes ອື່ນງອີກ, ແລະ methods ເຫຼົ້ານັ້ນ
ແມ່ນ inexhaustible.



2. ຕົວຢ່າງຂອງການນຳໃຊ້ Class ໃນບົດຮຽນທີ່ຜ່ານມາ

2.5. ຕົວຢ່າງຂອງ int class ແລະ methods ຂອງມັນ

- __add__
- __sub__
- __mult__
- __truediv__
- __mod__
- __pow__
- __lshift__
- __rshift__, ...

code ດຳເນີນການເຮັດວຽກທາງຄະນິດສາດ (+, -, *, @, /, //, %, **, <<, >>, &, ^, |) ໂດຍການນຳໃຊ້ methods ຕ່າງໆເຫຼົ່ານີ້. ຕົວດຳເນີນການທີ່ພວກເຮົາໄດ້ສິກສາຢູ່ໃນບົດທີ 3 ມີພິດຕິກຳໃນການເອັນໃຊ້ methods ເຫຼົ່ານີ້ພາຍໃນ.



| ເຂົ້າໃຈຄວາມແຕກຕ່າງລະຫວ່າງ __div__ ແລະ __truediv__ method.

- ▶ ຕົວດຳເນີນການ /, ເຊິ່ງດຳເນີນການຫານໃນ Python, ໄດ້ຖືກອະທິບາຍວ່າເປັນວິທີການ __div__ ໃນ Python 2, ແຕ່ໃນ Python 3, ມັນຖືກອະທິບາຍວ່າເປັນວິທີການ __truediv__. ດັ່ງນັ້ນ, 200/100 ຖືກຕິຄວາມໝາຍເປັນ (200).__truediv__(100) ໃນ Python 3 ແລະ ສິ່ງຄໍາ float.
- ▶ ທ້າກໍານົດ (200). __div__(100) ກໍ່ໃຫ້ເກີດຂໍຜິດພາດໃນ Python 3.

3. ຄວາມແຕກຕ່າງລະຫວ່າງ Instances ແລະ Objects

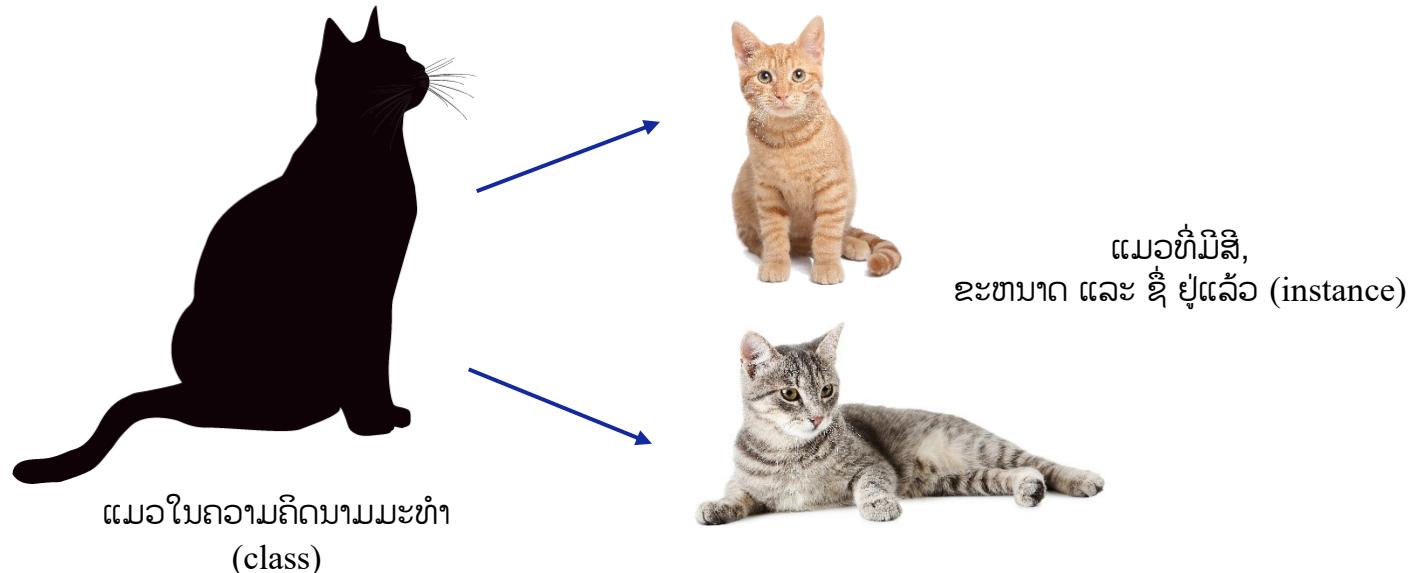
3.1. ການກຳນົດ instances ແລະ objects

| Class

- ▶ ແນວຄວາມຄືດນາມມະທຳທີ່ສະແດງເຖິງຊຸດຂອງ attributes ແລະ actions ທີ່ໃຊ້ໃນໂປຣແກຣມ.

| Instance

- ▶ object ແຕ່ລະລາຍການທີ່ຖືກສ້າງຂຶ້ນຈາກ class ຈະມີຄໍາ attribute ສະເພາະຂອງມັນ.



3. ຄວາມແຕກຕ່າງລະຫວ່າງ Instances ແລະ Objects

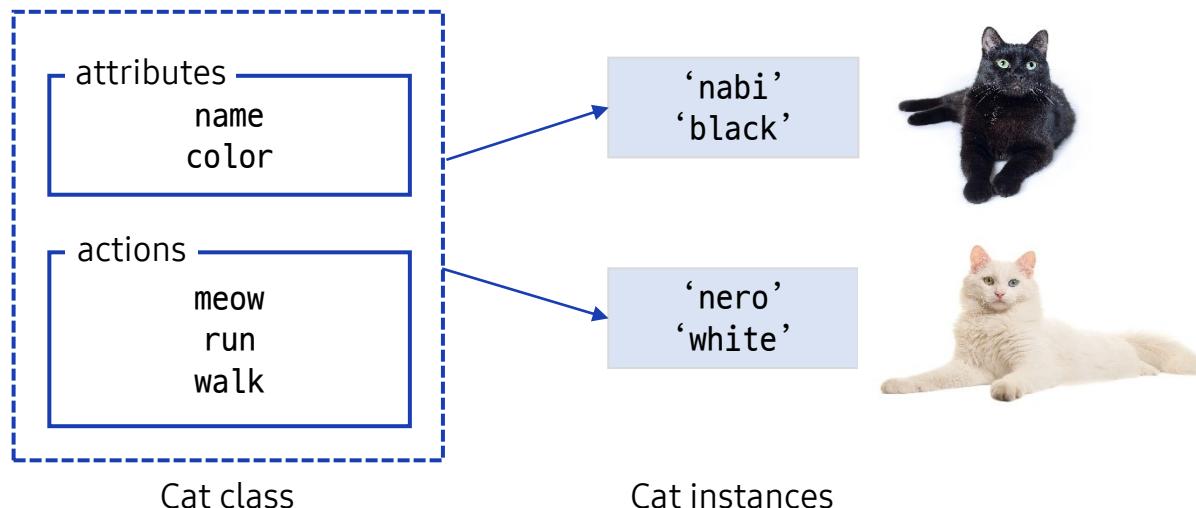
3.1. ການກຳນົດ instances ແລະ objects

| Class

- ▶ ຊຸດອັງ attribute ແລະ action ທີ່ໃຊ້ໃນໂປຣແກຣມ.
- ▶ plan, template ຫຼື blueprint ຂອງ objects

| Instance

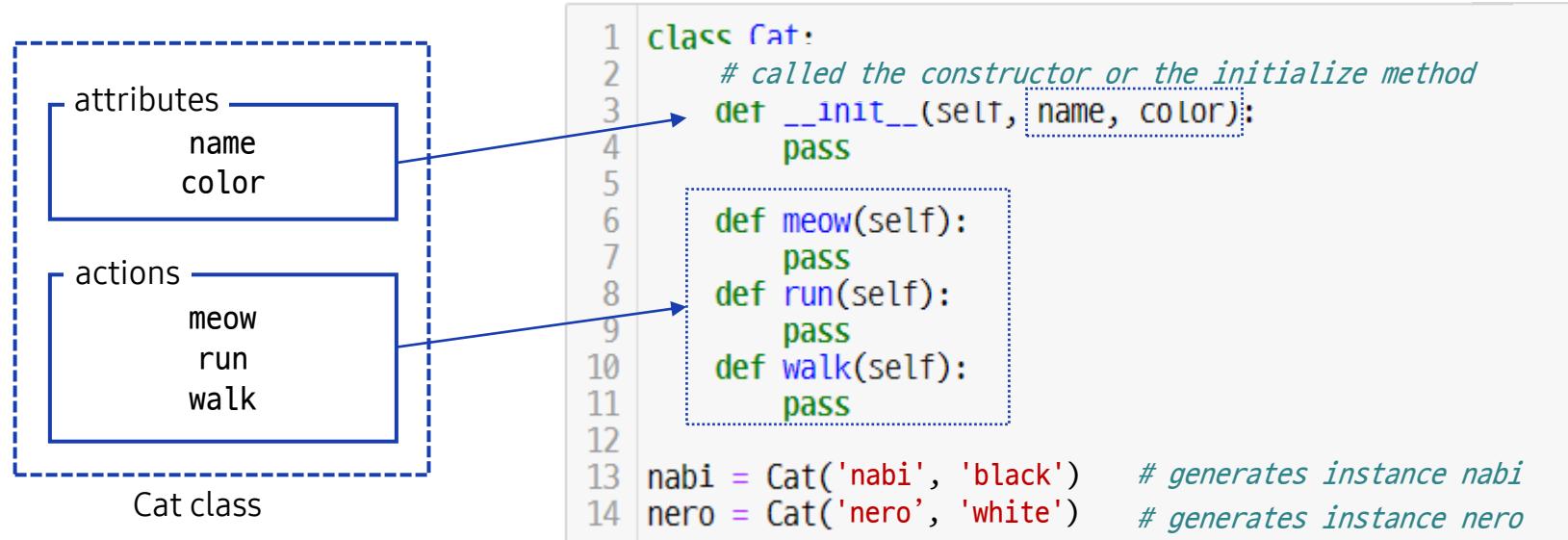
- ▶ object ແຕ່ລະລາຍການທີ່ສ້າງຂຶ້ນຈາກ class .
- ▶ instances ທີ່ຕ່າງກັນສາມາດມີຄໍາ attribute ຕ່າງກັນ.



3. ຄວາມແຕກຕ່າງລະຫວ່າງ Instances ແລະ Objects

3.2. Syntax ສໍາຫຼັບການກຳນົດ Class

| ການປະກາດ class, ຕ້ອງຊຽນຄໍາສັບ class ໄວຕໍ່ໜ້າຊື່ class, ສ່ວນຄໍາສັບ def ທີ່ເປັນ method ໃນ class ແມ່ນໃຫ້ຫຍັບເຂົ້າ, ດັ່ງສະແດງລຸ່ມນີ້



3. ຄວາມແຕກຕ່າງລະຫວ່າງ Instances ແລະ Objects

3.3. ຄວາມແຕກຕ່າງທີ່ຊັດເຈນລະຫວ່າງ object ແລະ instance

- | ເອກະສານຈໍານວນຫຼາຍປະສົມ object ແລະ instance ເປັນຂໍ້ກຳນົດຂອງແນວຄວາມຄິດທີ່ຄ້າຍຄືກັນ.
- | ທັງສອງຢ່າງນັ້ນແມ່ນມີຄວາມຄ້າຍຄືກັນ, ແຕ່ເພື່ອໃຫ້ຊັດເຈນກວ່າ, object ສາມາດຖືກກຳນົດໃຫ້ເປັນສ່ວນປະກອບຫຼືສິ່ງໄດ້ທີ່, ໃນຂະນະທີ່ instance ຖືກກຳນົດເປັນສ່ວນປະກອບ ຫຼື ສິ່ງທີ່ສ້າງຂຶ້ນໂດຍ class.
- | ຕົວຢ່າງ, ຈາກແຜນວາດທີ່ຜ່ານມາ, Cat ແມ່ນ class ເຮັດທີ້ເປັນ framework. ສ່ວນປະກອບທີ່ nabi ສ້າງໂດຍ class ແມ່ນ " object " ແລະ ໃນເວລາ ດຽວກັນກຳເປັນ " instance ຂອງ class Cat . "
- | ເພື່ອສະຫຼຸບທັງໝົດ, ຄໍາອະທິບາຍຕໍ່ໄປນີ້ແມ່ນຖືກຕ້ອງ.
 - 1) object ທັງໝົດຂອງ Python ມີປະເພດຂໍ້ມູນຂອງມັນ.
 - 2) instance ຂອງ Python ແມ່ນສ້າງຂຶ້ນຈາກ class.
 - 3) object ຂອງ Python ແມ່ນສ້າງຈາກ class.
 - 4) class ຂອງ Python ແມ່ນ object.
 - 5) nabi ເປັນ object.
 - 6) instance ຂອງ class cat ແມ່ນ nabi.
 - 7) 100 ແມ່ນ object ປະເພດ integer .

4. ການປະກາດ Class, Constructor ແລະ self Parameter

4.1. Syntax ຫົວໄປສໍາລັບການກຳນົດ class

- | ການກຳນົດ class ໂດຍຫົວໄປ ແລະ ຫົດສອບການ run.
- | ຂຽນ keyword ຊື່ຂອງ class ແລະ ການຕັ້ງຊື່ໃຫ້ class.
 - ▶ ຫຼັງຈາກນີ້ນີ້, ຂຽນ attributes ແລະ methods ທີ່ຈໍາເປັນໂດຍສອດຄ່ອງກັບ syntax Python.
- | pass statement ແມ່ນ Python statement ທີ່ບໍ່ຕ້ອງຮັດຫຍັງເລີຍ..
- | ເນື່ອງຈາກຟັງຊັ້ນພາຍໃນຂອງ class ບໍ່ໄດ້ກີກດໍາເນີນການ , ໃຫ້ເຮົາໃຊ້ pass statement.

```

1 class Cat:           # Define the Cat class
2     pass
3
4 nabi = Cat()         # Generate an instance of the Cat
5 print(nabi)

```

<__main__.Cat object at 0x000002A5EEAE71F0>


Line 4, 5

- ວິງເລັບແມ່ນຈຳເປັນເພື່ອ Cat instance.
- ເນື່ອດໍາເນີນການ print(nabi), ຄ່າ id ຂອງ Object ຕີກພິມອອກເປັນເລກຖານສືບທຶກ.

4. ການປະກາດ Class, Constructor ແລະ self Parameter

4.1. Syntax ທີ່ວໄປສໍາລັບການກຳນົດ class



| ຫຼັກການ ການຕັ້ງຂຶ້ນສໍາລັບຂຶ້ນ class ໃນ Python ແມ່ນດັ່ງຕໍ່ໄປນີ້.

- ▶ ໃຊ້ຕົວພິມນ້ອຍສໍາລັບຕົວອັກສອນທຳອິດຂອງ function, object ຫຼື variable.
- ▶ ໃຊ້ຕົວພິມໃຫຍ່ສໍາລັບຕົວອັກສອນທຳອິດຂອງ class ,
- ▶ ຖ້າຊື່ມີຫຼາຍກວ່າສອງຄໍາ, ໃຫ້ອັກສອນທຳອິດຂອງຄໍາທີ່ສອງເປັນຕົວພິມໃຫຍ່.
- ▶ ເພື່ອບ້ອງກັນ object ຫຼື class, ເນື້ອກໍານົດ attribute ໃຫ້ເລີ່ມຕົ້ນຄໍາທຳອິດເຕັ້ງວ່ານີ້ (_). (attribute ພົບທີ່ໃຫຍ່ມີຫຼາຍກວ່າສອງຄໍາທີ່ສອງເປັນຕົວພິມໃຫຍ່ ໂດຍກິຈຈາກ class ຫຼື object ພາຍນອກ).
- ▶ ຖ້າຕ້ອງການໃຊ້ຊື່ຂອງຄໍາທີ່ສະຫງວນໄວ້ເປັນຊື່ຂອງຕົວປ່ຽນ, ໃຫ້ເພີ່ມເຄື່ອງໝາຍຂິດກ້ອງຫຼັງຄໍາທີ່ສະຫງວນນັ້ນ.
- ▶ private attribute ມີໂຄງສ້າງທີ່ມີການແກ້ໄຂຊື່ພາຍໃນເພື່ອໃຫ້ເຂົ້າເຖິງຈາກພາຍນອກຖືກປະຕິເສດ. ຖ້າເພີ່ມ __ (double underscore), ຊື່ _class ຈະຕາມມາໂດຍອັດຕະໂນມັດ.
- ▶ ສໍາລັບ special attributes ທີ່ສະເພາະເຈາະຈີ່ ຫຼື methods ພາຍໃນ Python, ໃຫ້ເພີ່ມ __ ໃນດ້ານທັນໆ ແລະ ຫ້າຍຂອງຊື່.

4. ການປະກາດ Class, Constructor ແລະ self Parameter

4.2. ນໍາໃຊ້ methods ແລະ self

- | ສຶກສາ methods ແລະ self ຜ່ານຕົວຢ່າງຕໍ່ໄປນີ້ຂອງ Cat class.
- | Method ຫຼື ບັນດາ function ຕ່າງໆແມ່ນຟັງຊັນທີ່ດຳນິດໄວ້ພາຍໃນ class ທີ່ຖືກນໍາໃຊ້ໂດຍ class ຫຼື class instances.
- | Self parameter ຂອງ method meow ເປັນຕົວປຸງທີ່ອ້າງອີງຖິງໂຕມັນເອງ. ມັນເປັນຂໍ້ບັງຄັບທຳອິດຂອງ parameter ໃນ method.

```

1 class Cat:
2     def meow(self):
3         print('meow meow~~~')
4
5 nabi = Cat()
6 nabi.meow()

```

method ທີ່ເປັນຟັງຊັນພາຍໃນ Cat class

ສ້າງ object ຂອງ Cat class ຜ່ານ Cat
ຕອນນີ້ເຮົາສາມາດເຮັ້ນໃຊ້ method ຜ່ານ nabi.meow

meow meow~~~

4. ການປະກາດ Class, Constructor ແລະ self Parameter

4.3. Syntax ສໍາລັບການເອັນໃຊ້ methods

- | ໃຊ້ . (ຈຳເນັດ) ເປັນຕົວດຳເນີນການໃນການເອັນໃຊ້ methods ສໍາຫຼັບ instance.
 - ▶ ຈາກນັ້ນຂຽນ attributes ແລະ methods ທີ່ຈຳເປັນຕາມ syntax ຂອງ Python.
- | ຕາມດ້ານລຸ່ມນີ້ instance nabi ສາມາດເອັນໃຊ້ method ທີ່ຂຶ້ວ່າ meow ໃນ Cat class ໄດ້.

```
1 class Cat:  
2     def meow(self):  
3         print('meow meow~~~')  
4  
5 nabi = Cat()  
6 nabi.meow() ← nabi object ດຳເນີນການໃຊ້ meow  
7 nero = Cat()  
8 nero.meow() ← nero object ດຳເນີນການໃຊ້ meow  
9 mimi = Cat()  
10 mimi.meow() ← mimi object ດຳເນີນການໃຊ້ meow  
  
meow meow ~~~  
meow meow ~~~  
meow meow ~~~
```

4. ການປະກາດ Class, Constructor ແລະ self Parameter

4.4. Constructor method `__init__` ແລະ `__str__`

| Constructor `__init__`

- ▶ ເປັນ method ທີ່ກຳນົດຄ່າເລີ່ມຕົ້ນກັບຕົວປຸງພາຍໃນ instance ເວລາສ້າງ object.
- ▶ ຊຶ່ຂອງມັນແມ່ນ `__int__`.
- ▶ ມັນຈະຮັດວຽກອັດຕະໂນມັດເມື່ອ object ຕີກສ້າງ .

| `__str__` method

- ▶ ເປັນ method ທີ່ສະແດງຂໍ້ມູນຂອງ object ໃນຮູບແບບ string .
- ▶ ຕົວຢ່າງ: ພາຍໃນ Cat class : `__str__` method ສະແດງຊື່ ແລະ ສີຂອງເມືວ.

4. ການປະກາດ Class, Constructor ແລະ self Parameter

4.5. ນິຍາມ Class ແລະ `__init__`, `__str__` method ສະເພາະ

- | ເພີ່ມ `__str__` method ໃສ່ Cat class ຕາມຮູບຂ້າງລຸ່ມນີ້. method ນີ້ຈະຖືກເອີ້ນອັດຕະໂນມັດໂດຍຝຶ່ງຊັນ print.
- | `__str__` method ຈະສໍ່ງ string ທີ່ພິມໄດ້ຂອງ object ເປັນຢ່າງດີ.
 - ▶ ຖ້າ run `print(nabi)` ເພື່ອພິມ name ແລະ color ຂອງ Cat, `print(nabi.__str__())` ຈະຖືກເອີ້ນໂດຍອັດຕະໂນມັດ.

```

1  class Cat:
2      def __init__(self, name, color):
3          self.name = name
4          self.color = color
5
6      # A string expression format of the Cat class
7      def __str__(self):
8          return 'Cat(name='+self.name+', color='+self.color+')'
9
10
11 nabi = Cat('nabi', 'black') # generate instance nabi
12 nero = Cat('nero', 'white') # generate instance nero
13
14 print(nabi)
15 print(nero)

```

ເລີ່ມຕົ້ນເຮັດຕ່າງໆ ເມື່ອ Cat instances ອີກສ້າງ
ຂຶ້ນ.

method ສະເພາະ ທີ່ກຳນົດຮູບແບບ
ການສະແດງອອກຂອງ string ໃນ Cat class

Cat(name=nabi, color=black)
Cat(name=nero, color=white)

4. ການປະກາດ Class, Constructor ແລະ self Parameter

4.6. Constructor `__init__` ແລະ self parameter

- | Parameter ຕົວທໍາອິດ ຊື່ self ໃນ method ນີ້ອ້າງອິງເຕິງ instances ຂອງ class ປັດຈຸບັນ.
- | parameter ຕົວທີ່ສອງ ຊື່ name ແລະ parameter ຕົວທີ່ສາມ ຊື່ color ແມ່ນຕົວປ່ຽນທີ່ກຳນົດ ຊື່ ແລະ ສີ ຂອງແມວ ແຊ້ງແມ່ນ attributes ຂອງ instance.

```
class Cat:  
    def __init__(self, name, color):  
        ...  
  
nabi =Cat('nabi', 'black')  
nero =Cat('nero', 'white')  
mimi =Cat('mimi', 'brown')
```

4. ການປະກາດ Class, Constructor ແລະ self Parameter

4.7. ເມື່ອໄດ້ທີ່ method ສະເພາະ `__str__` ທີ່ກາເອັນໃຊ້ ?

| ໂດຍທົ່ວໄປ, methods ສະເພາະ `__str__` ຈະຖືກເອັນໃຊ້ໂດຍອັດຕະໂນມັດ.

- ▶ ໃນໂຄດລຸ່ມນີ້, ຂໍ້ມູນຖືກພິມອອກໂດຍ format method ກັບເຄື່ອງໝາຍ {}, method `__str__` ຈະຖືກເອັນໃຊ້ ແລະ ດຳເນີນການໂດຍອັດຕະໂນມັດ.

```
1 print('information of nabi : {}' .format(nabi))
```

```
information of nabi : Cat (name=nabi, color=black)
```

5. ໂຄງສໍາງຂອງ Method

5.1. ນຶ່ຍາມຂອງ method ແລະ ຕົວຢ່າງ code ຂອງການເອີ້ນໃຊ້ method

```

1 class Cat:
2     # Called as a constructor or the initialize method
3     def __init__(self, name, color):
4         self.name = name      # Generates an instance variable named name
5         self.color = color    # Generates an instance variable named color
6
7     # The method that prints the information of a cat
8     def meow(self):
9         print('My name is {}, color is {}, meow meow~~' .format(self.name, self.color))
10
11 nabi = Cat('nabi', 'black')      # Generates the instance Nabi
12 nero = Cat('nero', 'white')      # Generates the instance Nero
13 mimi = Cat('mimi', 'brown')      # Generates the instance Mimi
14
15 nabi.meow()
16 nero.meow()
17 mimi.meow()

```

ການກຳນົດ meow ຂອງ class Cat
ຜ່ານ keyword: def.

meow(), method ຂອງ class Cat, ສາມາດເອີ້ນໃຊ້ໄດຍ່
instances ຂອງ class Cat, Nabi, Nero ແລະ Mimi.

My name is Nabi, color is black, meow meow~~

My name is Nero, color is white, meow meow~~

My name is Mimi, color is brown, meow meow~~

5. ໂຄງສ້າງຂອງ Method

5.2. Instance variables, member variables, fields

- | ສຶກສາຂຶ້ນກຳນົດຂອງ instance variable, member variable ແລະ field.
- | ແຕ່ລະຂຶ້ນກຳນົດຫຼູ້ນີ້ທາມເຖິງຕົວປ່ຽນທີ່ເວັບຮັກສາຄຸນລັກສະນະທີ່ແຕກຕ່າງກັນຂອງແຕ່ລະ instances.

self.name = name
self.color = color
self ແມ່ນເຮັດວຽກປະມານນັ້ນສໍາລັບແຕ່ລະ instance.

nabi
instance

nabi.name = 'Nabi'
nabi.color = 'black'

nero
instance

nero.name = 'Nero'
nero.color = 'white'

mimi
instance

mimi.name = 'Mimi'
mimi.color = 'brown'

5. ໂຄງສໍາງຂອງ Method

5.3. ຕົວຢ່າງ code ຂອງການເອີ້ນໃຊ້ methods

```

1 class Cat:
2     # 생성자 # Called a constructor or the initialize method
3     def __init__(self, name, color):
4         self.name = name # Generates an instance variable named name
5         self.color = color # Generates an instance variable named color
6
7     # The method that prints the information of a cat
8     def meow(self):
9         print('My name is {}, color is {}, meow meow~~' .format(self.name, self.color))
10
11 nabi = Cat('nabi', 'black') # Generates the instance Nabi
12 nero = Cat('nero', 'white') # Generates the instance Nero
13 mimi = Cat('mimi', 'brown') # Generates the instance Mimi
14
15 nabi.meow()
16 nero.meow()
17 mimi.meow()

```

My name is Nabi, color is black, meow meow~~

My name is Nero, color is white, meow meow~~

My name is Mimi, color is brown, meow meow~~



Line 4, 5

- Nabi, Nero, Mimi ແມ່ນ instances ຂອງ class Cat. ສະນັ້ນ, ເຮົາສາມາດເອີ້ນໃຊ້ meow, ຊຶ່ງເປັນ method ຫຼື່ຈອງ class Cat ກັບຕົວທຳເນີນການ . (ຈຳເປັດ) .

6. ຕົວປັບປຸງ Class ແລະ Methods

6.1. instance variables ແມ່ນຫຍັງ?

- | ສຶກສາກ່ຽວກັບ instance variable, ເຊິ່ງຖືກນຳໃຊ້ໃນຂະນະທີ່ກຳນົດ class.
- | ເນື່ອຈາກຄຸນລັກສະນະຂອງ class ແຕກຕ່າງກັນໄປຕາມແຕ່ລະ instances, ພວກມັນສາມາດມີຄ່າທີ່ແຕກຕ່າງກັນ.
- | ສັງເຫຼີນເອັນວ່າ instance variables. ກວດເບິ່ງ class Circle ທີ່ຂຽນຂ້າງລຸ່ມນີ້.

```

1  class Circle:
2      def __init__(self, name, radius, PI):
3          self.__name = name      # instance variable
4          self.__radius = radius  # instance variable
5          self.__PI = PI         # instance variable
6
7      # Compute the area by multiplying radius**2 to current instance's PI
8      def area(self):
9          return self.__PI * self.__radius ** 2
10
11
12 c1 = Circle("C1", 4, 3.14)
13 print("Area of c1:", c1.area())
14 c2 = Circle("C2", 6, 3.141)
15 print("Area of c2:", c2.area())
16 c3 = Circle("C3", 5, 3.1415)
17 print("Area of c3:", c3.area())

```

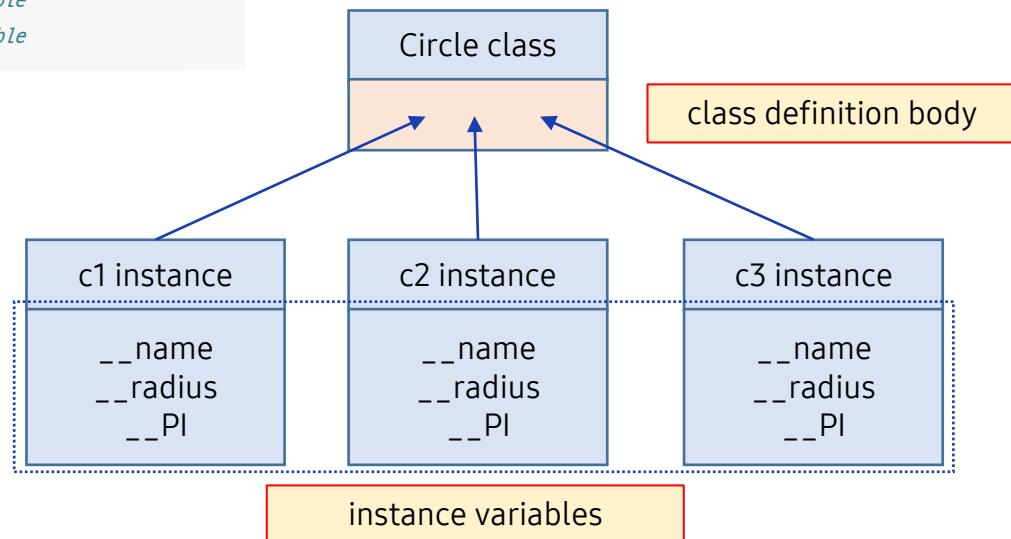
Area of c1: 50.24
 Area of c2: 113.076
 Area of c3: 78.53750000000001

6. ຕົວປັບປຸງ Class ແລະ Methods

6.1. instance variables ແມ່ນຫຍັງ?

- | __name, __radius, __PI ໃນ code ຂ້າງລຸ່ມນີ້ມີຄ່າທີ່ແຕກຕ່າງກັນສໍາລັບແຕ່ລະ instance.
- | ຕົວປັບປຸງດັ່ງກ່າວເອີ້ນວ່າ instance variables.
- | ເຫດຜົນສໍາລັບເຄືອຂາຍ __ ແມ່ນເພື່ອຊື່ອງ instance ເຫຼື້ນຈາກການເຂົ້າເຖິງພາຍນອກ.

```
class Circle:
    def __init__(self, name, radius, PI):
        self.__name = name      # instance variable
        self.__radius = radius   # instance variable
        self.__PI = PI           # instance variable
```



6. ຕົວປ່ຽນ Class ແລະ Methods

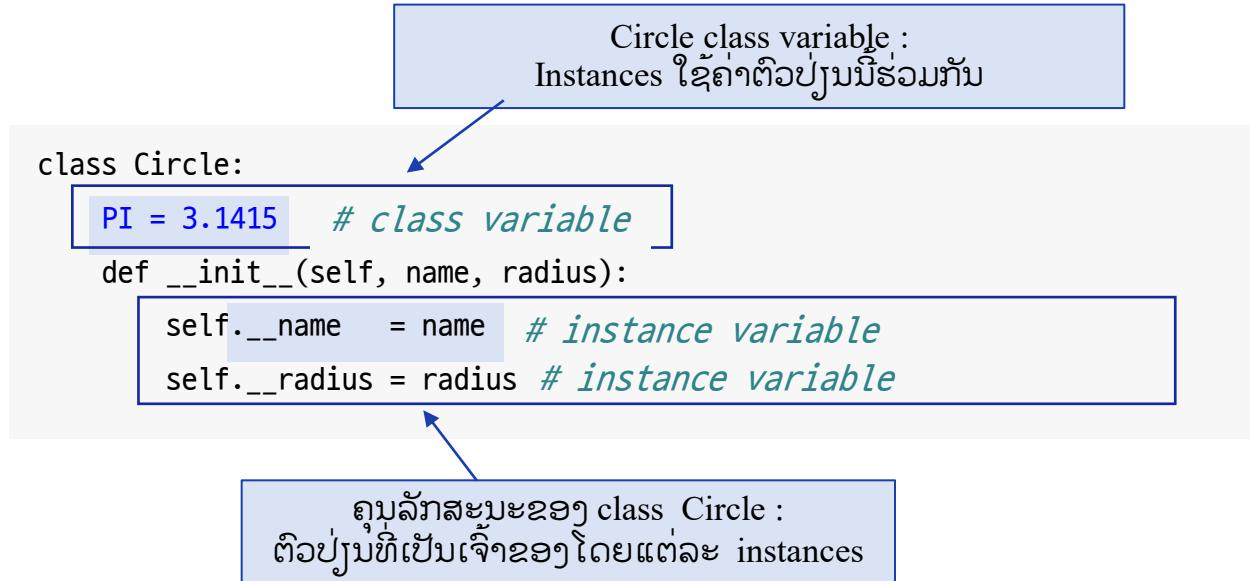
6.2. Class variables ແມ່ນຫຍັງ?

- | Instance variables ອາດມີຄຸນລັກສະນະທີ່ໄປ ທີ່ຕ້ອງໃຊ້ຮ່ວມກັນໃນ class.
- | ໃນຕົວຢ່າງ code, PI ແມ່ນຄຸນລັກສະນະຂອງປະເພດດັ່ງກ່າວ. ຖ້າຄຸນລັກສະນະນີ້ຖືກໃຊ້ຮ່ວມກັນໂດຍແຕ່ລະ instances, ການຊໍາກັນຂອງຂໍ້ມູນຈະຫຼຸດລົງ, ແລະ ການກວດສອບຄວາມຜິດພາດຈະຢ່າຍຂຶ້ນ.
- | ຖ້າຄ່າຂອງ self.__PI ຂອງ class Circle ມີຄ່າຄົງທີ່, ສະແດງວ່າ c1, c2 ແລະ c3 ແມ່ນໃຊ້ຄ່າ PI ຮ່ວມກັນ .
- | ໃນທີ່ນີ້, ຄວນໃຊ້ class variable.

6. ຕົວປຽນ Class ແລະ Methods

6.2. Class variables ແມ່ນຫຍັງ?

I class variable ຂອງ Circle, PI ເປັນຕົວປຽນທີ່ໃຊ້ຮ່ວມກັນໂດຍ instances ຂອງ class ນີ້. class attributes: __name ແລະ __radius ແມ່ນ instance variables ທີ່ເປັນເຈົ້າຂອງໂດຍແຕ່ລະ instances.



6. ຕົວປຸງ Class ແລະ Methods

6.3. ການກຳນົດ class variables

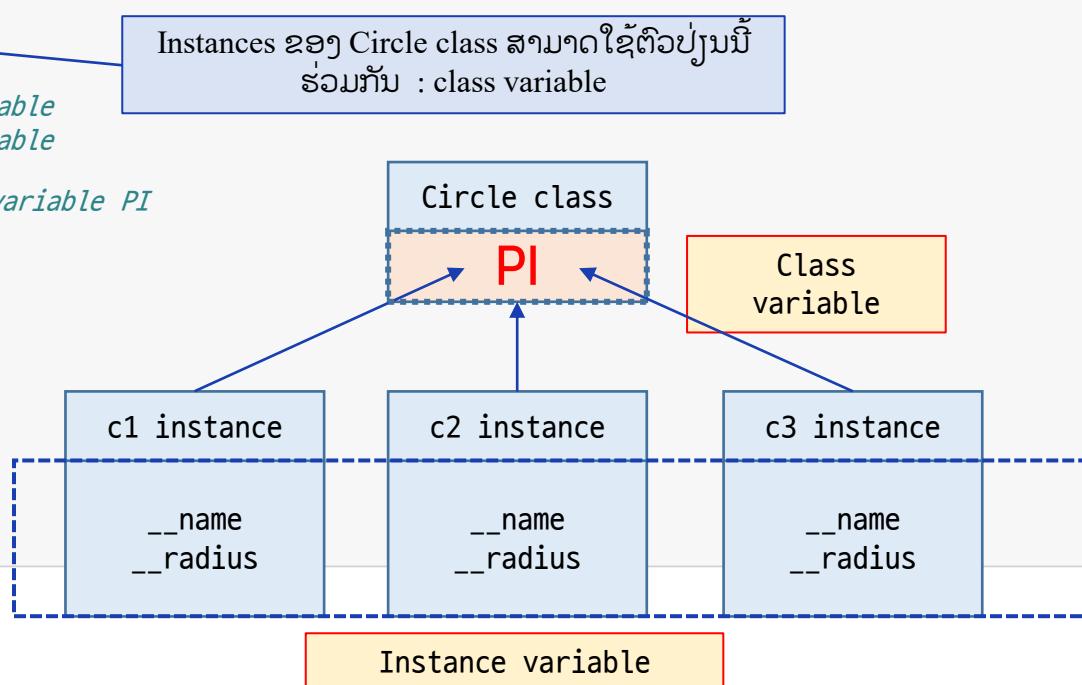
- | Class variables ທີ່ກຳນົດຢູ່ພາຍນອກ methods ທີ່ສະແດງລຸ່ມນີ້. class variable ທີ່ຊື່ PI ນີ້ທີ່ໃຊ້ຮ່ວມກັນໂດຍ instances c1, c2 ແລະ c3.
- | ຜ້າຕ້ອງການກຳນົດຄ່າທີ່ຖືກຕ້ອງຫຼາຍຂຶ້ນໃຫ້ກັບຄ່າ PI , ກຳນຽງແຕ່ຕ້ອງບ່ຽນແປງຄ່າຕົວແປເຊັ່ນ PI=3.1141592.

```

1 class Circle:
2     PI = 3.1415 # class variable
3     def __init__(self, name, radius):
4         self.__name = name # instance variable
5         self.__radius = radius # instance variable
6
7     #Compute the area by using Circle's class variable PI
8     def area(self):
9         return Circle.PI * self.__radius ** 2
10
11
12 c1 = Circle("C1", 4)
13 print("Area of c1:", c1.area())
14 c2 = Circle("C2", 6)
15 print("Area of c2:", c2.area())
16 c3 = Circle("C3", 5)
17 print("Area of c3:", c3.area())

```

Area of c1: 50.264
 Area of c2: 113.09400000000001
 Area of c3: 78.53750000000001



6. ຕົວປຽນ Class ແລະ Methods

6.4. Code ທີ່ໃຊ້ class variable PI

| ພຶນທີ່ຂອງ class Circle ໃນ code ຂ້າງລຸ່ມນີ້ແມ່ນ method ທີ່ສາມາດເອັນໄດ້ໂດຍ instances ຂອງ class Circle. Method ນີ້ໃຊ້ class variable PI ຜ່ານ Circle.PI.

```

1 class Circle:
2     PI = 3.1415    # class variable
3     def __init__(self, name, radius):
4         self.__name = name      # instance variable
5         self.__radius = radius  # instance variable
6
7     # Computes the area by using Circle's class variable PI
8     def area(self):
9         return Circle.PI * self.__radius ** 2
10
11
12 c1 = Circle("C1", 4)
13 print("Area of c1:", c1.area())
14 c2 = Circle("C2", 6)
15 print("Area of c2:", c2.area())
16 c3 = Circle("C3", 5)
17 print("Area of c3:", c3.area())

```

method ຂອງ class Circle.
Instances ອ້າງອີງ class variable ຜ່ານ Circle.PI
ເພື່ອຈະໃຊ້ method ນີ້.

Area of c1: 50.264
Area of c2: 113.09400000000001
Area of c3: 78.53750000000001

6. ຕົວປັນ Class ແລະ Methods

6.5. __dict__ ເຊິ່ງສະແດງຂໍ້ມູນຄຸນລັກສະນະຂອງ class objects

- | __dict__ ແມ່ນຄຸນລັກສະນະທີ່ເປັນປະໂຫຍດສໍາລັບໂປຣແກຣມ.
- | ເນື້າສາມາດປັນຕົວປັນຂອງ class object ເປັນ dictionary type.
- | ເນື້າສາມາດເອີ້ນຄໍາ attribute ດັ່ງປ່າງຈ່າຍ ໂດຍການປັນເປັນ dictionary type.

```

1 class Circle:
2     PI = 3.14
3     def __init__(self, name, radius):
4         self.name = name
5         self.radius = radius
6
7 c1 = Circle("C1",4)
8 print("Attributes of c1:", c1.__dict__)
9 # You can obtain the value with dic_[key] format
10 print("Value of c1's name variable:", c1.__dict__['name'])
11 print("Value of c1's radius variable:", c1.__dict__['radius'])

```

- ປັນປະເພດຂອງ object ເປັນ dictionary type.
ຈາກນັ້ນສະແດງຄໍາ C1 ສໍາລັບ name.
- ສະແດງ 4, ຄໍາສໍາລັບຕົວປັນ radius.

Attributes of c1 :{'name': 'C1', 'radius':4}
Value of c1's name variable: C1
Value of c1's radius variable: 4

<https://minimilab.tistory.com/58> [MINIMI LAB]

6. ຕົວປຽນ Class ແລະ Methods

6.5. `__dict__` ເຊິ່ງສະແດງຂໍ້ມູນຄຸນລັກສະນະຂອງ class objects

- | ຖ້າມີ instance variable ກັບ `(ຂີດກ້ອງ)`, ໃຫ້ໃຊ້ method ຕໍ່ໄປນີ້.
 - ▶ class Circle ມີຕົວປຽນ `__radius` ແລະ `__name`.
- | ເຮົາສາມາດດຶງເອົາຂໍ້ຂອງ instance variable ດ້ວຍ `.__dict__` attribute ຂອງ c1. ໃຊ້ຕົວປຽນນີ້ເປັນ dictionary's key.
 - ▶ ຕົວປຽນ `__name` ສາມາດໃຊ້ `_Circle__name` ເປັນ key.
 - ▶ ຕົວປຽນ `__radius` ສາມາດໃຊ້ `_Circle__radius` ເປັນ key.

```

1 class Circle:
2     PI = 3.14
3     def __init__(self, name, radius):
4         self.__name = name
5         self.__radius = radius
6
7 c1 = Circle("C1",4)
8 print("Attributes of c1:", c1.__dict__)
9
10 # You can obtain the value with dic_[key] format
11 print("Value of c1's name variable:", c1.__dict__['Circle__name'])
12 print("Value of c1's radius variable:", c1.__dict__['Circle__radius'])

```

Attributes of C1 :{'_Circle__name': 'C1', '_Circle__radius':4}

Value of C1's__name variable: C1

Value of C1's__radius variable: 4

7. Class Inheritance

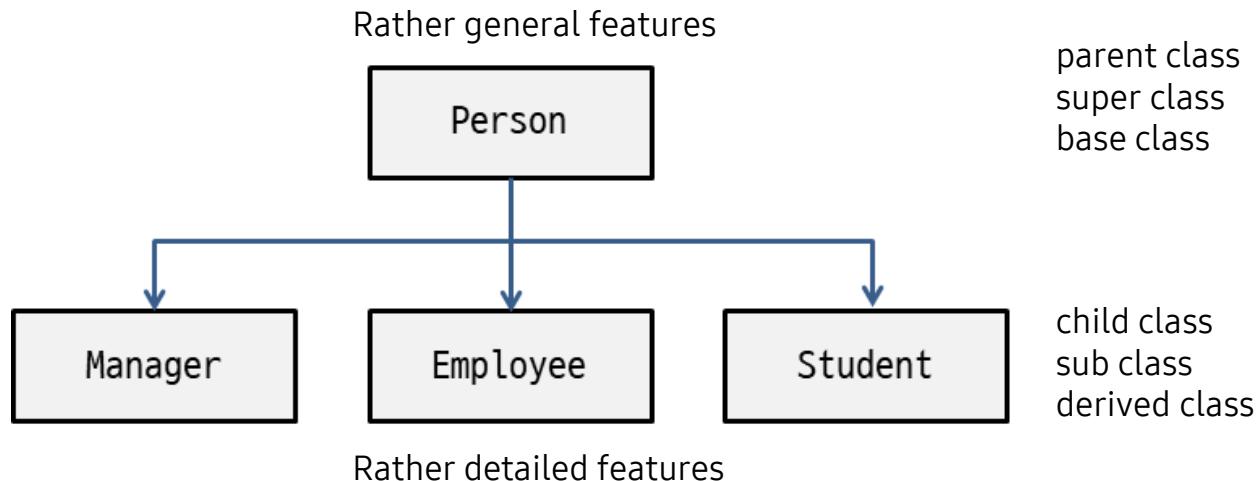
7.1. ແນວຄວາມຄິດຂອງ Class ແລະ ການສືບທອດ(Inheritance)

- | ການສືບທອດ(Inheritance) ແມ່ນເຕັກນິກທີ່ອະນຸຍາດໃຫ້ຜູ້ໃຊ້ຂຽນ class ລູກ ທີ່ສືບທອດວິທີການ ແລະ ຄຸນລັກສະນະຂອງ class ທີ່ສ້າງຂຶ້ນກ່ອນທັນນີ້.
- | ເພີ້ມສໍາລັບການສືບທອດຈາກ class ພໍ່ແມ່ ແມ່ນຂະບວນການພັດທະນາທີ່ມີປະສິດທິພາບ.
- | ການສືບທອດ ແມ່ນໜຶ່ງໃນຄຸນສົມບັດຕົ້ນຕຳຂອງການຂຽນໂປຣແກຣມແບບວັດຖ..
- | ລຸ່ມນີ້ແມ່ນຈຸດສະຫຼຸບຂອງຂໍ້ກຳນົດກ່ອນທີ່ຈະໄດ້ຮັບການເພີ່ມເຂົ້າໄປໃນ ການສືບທອດ(Inheritance).
 - ▶ class ສູງກວ່າທີ່ມີການຖ່າຍທອດ ວິທີການ ແລະ ຄຸນລັກສະນະຕ່າງໆເອັນວ່າ parent class, super class ຫຼື base class
 - ▶ class ທີ່ສືບທອດແມ່ນເອັນວ່າ child class, sub class ຫຼື derived class

7. Class Inheritance

7.1. ແນວຄວາມຄິດຂອງ Class ແລະ ການສືບທອດ(Inheritance)

- ຝາກຄວາມຄິດການສືບທອດແມ່ນສະແດງຢູ່ໃນຮູບລຸ່ມນີ້. Person ຂຶ້ນເປັນ parent class ມີລັກສະນະທົ່ວໄປ.
- Class Person ຈະມີລັກສະນະທົ່ວໄປເຊັ່ນ: age, gender, name ແລະ ອື່ນໆ.
- ຢ່າງໃດກໍຕາມ, ບັນດາ child class ເຊັ່ນ: Manager, Employee ແລະ Student ທີ່ສືບທອດຄູນລັກສະນະຈາກ Person ຈະມີຄຸນລັກສະນະຄືກັບ Person



7. Class Inheritance

7.2. Syntax ຂອງ Class ແລະ ການສືບທອດ (Inheritance)

| class inheritance ສາມາດສະແດງອອກໃນ syntax ຂອງ Python ໄດ້ແນວໃດ? syntax ແມ່ນສະແດງດັ່ງລຸ່ມນີ້.

```
1 class A:      # parent class A
2     statements
3
4 class B(A):  # child class B with A as the parent
5     statements
```

- ▶ ດັ່ງທີ່ສະແດງຢູ່ຂ້າງເທິງ, class ລູກ ຈະສືບທອດໂດຍການໃສ່ວົງເລັບໃຫ້ກັບຊື່ຂອງ class ພໍ່ແມ່.
- ▶ ຫຼັງຈາກນັ້ນ class ລູກ ສາມາດສືບທອດວິທີການ ແລະ ຄຸນລັກສະນະຂອງ class ພໍ່ແມ່: A ພ້ອມທັງ ກຳນົດຄຸນລັກສະນະ ແລະ ວິທີການໃຫມ່ຂອງຕົນ ເອງເຊັ່ນກັນ.

7. Class Inheritance

7.2. Syntax ຂອງ Class ແລະ ການສືບທອດ (Inheritance)

 Focus Class ລູກ ມີການເຂົ້າເຖິງ methods ຫຼື variables ຂອງ Class ພໍ່ແມ່ໄດ້.

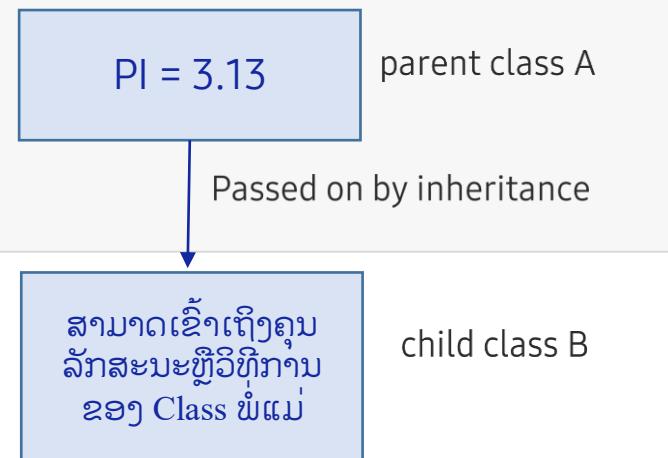
- | ໃນລຳດັບລຸ່ມນີ້, A ແມ່ນ Class ພໍ່ແມ່, B ແມ່ນ Class ລູກ.
- | ຕົວປ່ຽນ a ຂອງ class A ສາມາດເຂົ້າເຖິງ Pi ໄດ້ໂດຍກິງ, ເຊິ່ງແມ່ນ class variable ຂອງ A.
- | ຢ່າງໃດກໍຕາມ, class B ບໍ່ມີ PI ທີ່ເປັນຕົວປ່ຽນຂອງຕົວເອງ. ເນື່ອງຈາກວ່າ B ແມ່ນ Class ລູກຂອງ, ຕົວຢ່າງ: ຕົວປ່ຽນ b ຂອງ class B ສາມາດເຂົ້າເຖິງຕົວປ່ຽນ ຂອງ Class ພໍ່ແມ່ໄດ້.

```

1 class A :
2     PI = 3.14
3
4 class B(A):
5     pass
6
7 a = A()      # Generates instance a of class A  생성
8 b = B()      # Generates instance a of class B  , 생성
9 print('a.PI =', a.PI)
10 print('b.PI =', b.PI)   # b can access the attributes of the parent class

```

a.PI = 3.14
b.PI = 3.14



Paper coding

- ຕ້ອງເຂົ້າໃຈແນວຄິດພື້ນຖານຂອງຫຼັກສູດນີ້ໃຫ້ຄົບຖ້ວນກ່ອນຈະໄປສູ່ຂັ້ນຕອນຕໍ່ໄປ.
- ການທີ່ບໍ່ເຂົ້າໃຈແນວຄິດພື້ນຖານ ຈະເພີ່ມພາລະໃນການຮຽນຮູ້ຂອງຫຼັກສູດນີ້ ແລະ ຈະເຮັດໃຫ້ບໍ່ປະສົບຜົນສໍາເລັດ.
- ມັນອາດຈະເປັນເລື່ອງທີ່ຍາກຕອນນີ້, ແຕ່ຖ້າຢາກປະສົບຜົນສໍາເລັດໄດ້ນັ້ນ ພວກເຮົາຂໍແນະນຳໃຫ້ເຂົ້າໃຈແນວຄິດພື້ນຖານ ນັ້ງຢ່າງເລິກເຊິ່ງ ແລະ ກ້າວໄປສູ່ຂັ້ນຕອນຕໍ່ໄປ.

Q1. ສ້າງ class Dog ແລະ objects ຂອງມັນດ້ວຍຟັງຊັນທີ່ອະທິບາຍຂ້າງລຸ່ມນີ້.

- a) ມີ method ຫຶ່ງທີ່ຊື່ວ່າ def bark(self): Method ນີ້ສະແດງສຽງເຫົ່າ.
- b) ສ້າງ instance ທີ່ມີຊື່ວ່າ Dog ແລະ ອ້າງອີງ my_dog ດ້ວຍຄໍາສັ່ງທີ່ມີຊື່ວ່າ my_dog = Dog.
- c) ສະແດງສຽງເຫົ່າດ້ວຍ method ທີ່ມີຊື່ວ່າ my_dog.bark()
“woof woof”

ເງື່ອນໄຂສໍາລັບ ການປະຕິບັດ	woof woof
ເວລາ	5 ນາທີ



Write the entire code and the expected output results in the note.

Q2.

ກຳນົດ class Dog ດ້ວຍຝັງຊັນທີ່ອະທິບາຍຂ້າງລຸ່ມນີ້ ແລະ ເອັນໃຊ້ instances ແລະ methods.

- a) class Dog ນີ້ມີ attribute ທີ່ຊື່ວ່າ name.
 - b) ມີ method ເລີ່ມຕົ້ນທີ່ຊື່ວ່າ def __init__(self, name): method ຈະເລີ່ມຕົ້ນໂດຍ Dog's name.
 - c) ມີ method ທີ່ຊື່ວ່າ def bark(self). Method ນີ້ສະແດງສຽງເຫັ້ນ.
 - d) ສ້າງ my_dog instance ທີ່ມີຊື່ ‘Bingo’ ດ້ວຍຄໍາສັ່ງ my_dog = Dog(‘Bingo’)
 - e) ສະແດງສຽງເຫັ້ນດ້ວຍ method ທີ່ມີຊື່ວ່າ my_dog.bark()
- “Bingo : woof woof”

ເງື່ອນໄຂສໍາລັບ ການປະຕິບັດ	Bingo : woof woof
ເວລາ	5 ນາທີ



Write the entire code and the expected output results in the note.

| Let's code

1. ការអនកແບບ Class և ការប៉ុម (Class Design and Encapsulation)

1.1. ទំនាក់ទំនងខ្លួនការប៉ុម (Importance of encapsulation)

- | រាយការណីត attribute ឱ្យ age សំឡូល Cat class
 - | សំគាល់ nabi និង រាយការណីត nabi.age = -5 ដើម្បីបង្ហាញពីការប៉ុមជាមួយការបង្ហាញពីរបាយការណ៍ ព័ត៌មានផ្ទាល់នូវការប៉ុម ឬមិនប៉ុម។
 - | តើតុច្សេចបានតុច្សេចរាយការណីតណាមតុច្សេចណា?
 - | សំគាល់កើតឡើងនូវការប៉ុមជាមួយការបង្ហាញពីរបាយការណ៍ class
-
- | ការប៉ុម
 - ▶ វិញ្ញុណីតជាគម្រោងទូទៅរបស់ការបង្ហាញពីរបាយការណ៍ class attribute ឬមិនប៉ុម។
 - ▶ វិញ្ញុរាយការណីតជាគម្រោងទូទៅរបស់ការបង្ហាញពីរបាយការណ៍ attribute ឬមិនប៉ុម។

1. ការអភ់របៀប Class និងការព័ត៌មាន (Class Design and Encapsulation)

1.1. គោលសំគាល់ខ្លួនការព័ត៌មាន (Importance of encapsulation)

- | ចាប់ផ្តើមជាមុនការរាយការណ៍របៀប Class.
- | ពិនិត្យការព័ត៌មានដែលត្រូវបានរាយការណ៍.
- | បានពិនិត្យការព័ត៌មានដែលត្រូវបានរាយការណ៍.
- | កំឡុងពេលនឹងការព័ត៌មានដែលត្រូវបានរាយការណ៍.

```
1 class Cat:  
2     def __init__(self, name, age):  
3         self.name = name  
4         self.age = age  
5  
6     # string expression format of Cat objects  
7     def __str__(self):  
8         return 'Cat(name=' + self.name + ', age=' + str(self.age) + ')'  
9  
10 nabi = Cat('nabi', 3) # generates instance nabi  
11 print(nabi)  
12 nabi.age = 4  
13 nabi.age = -5 ← An abnormal case of age being negative  
14 print(nabi) ← Not a syntax error but a logical problem that occurred because values were not protected.
```

1. ການອອກແບບ Class ແລະ ການຫົ່ວໜຸ່ມ (Class Design and Encapsulation)

1.2. ແນວດວາມຄິດຂອງການຫໍ່ຫຼຸ້ມ (Concept of encapsulation)

- | ແຜນທີ່ແນວຄວາມຄົດຂອງການຫຳຫຼຸມ(encapsulation)
 - | ແມ່ນຝັງຊັ້ນການເຮັດວຽກທີ່ລວມເອົາ methods ແລະ variables ຂອງ class ເພື່ອຈໍາກັດການປ່ຽນແປງຈາກພາຍນອກ



1. ການອອກແບບ Class ແລະ ການຫົ່ງຫຼຸມ (Class Design and Encapsulation)

1.3. ຕົວຢ່າງ ໂຄດການຫຳຫຼຸມ: (Code example of encapsulation)

| ໂຄນນີ້ໄດ້ຮັບການແກ້ໄຂໃຫ້ມີ `set_age` ເນື່ອງຈາກຄໍາສັ່ງແບບມີຕູ້ອນໄຂ `if age > 0, self.__age = age` ຈະບໍ່ເຮັດວຽກຖ້າ `age` ມີຄໍາເປັນລົບ

```
1 class Cat:
2     def __init__(self, name, age):
3         self.__name = name
4         self.__age = age
5
6     # string expression format of Cat objects
7     def __str__(self):
8         return 'Cat(name=' + self.__name + ', age=' + str(self.__age) + ')'
9
10    # limits random external access on self.__age and prevents age from being negative
11    def set_age(self, age):
12        if age > 0:
13            self.__age = age
14
15    def get_age(self):
16        return self.__age
17
18 nabi = Cat('nabi', 3) # generates instance nabi
19 print(nabi)
20 nabi.set_age(4)      # approaches age via set_age() method
21 nabi.set_age(-5)    # prevents age from being negative via set_age() method
22 print(nabi)
```

```
Cat(name=nabi, age=3)
Cat(name=nabi, age=4)
```

ວິທີປ້ອງກັນສະຖານະການທີ່ວ່າອາຍຸຈະມີຄໍາເປັນລົບ

1. ការអនករបៀប Class និង ការពិន្ទុ (Class Design and Encapsulation)

1.4. ការពិន្ទុ និង វាំងិតតា (Encapsulation and setter)

- | ឱ្យតាំង if របៀបមិច្ឆេទនៃខ្លួនដើម្បីបង្កាត់ការពិន្ទុរបស់ខ្លួន។ ម៉ោងតាមរយៈ (age) បែនកាត់លិប
 - | គឺជាដំឡើងក្នុងការណែនាំការពិន្ទុនៅក្នុងការបង្កាត់ខ្លួន។ ការពិន្ទុនេះត្រូវបានធ្វើឡើងតាមការបង្កាត់ខ្លួន។
 - | និងការបង្កាត់ខ្លួនត្រូវបានធ្វើឡើងតាមការបង្កាត់ខ្លួន។ ការបង្កាត់ខ្លួនត្រូវបានធ្វើឡើងតាមការបង្កាត់ខ្លួន។
- ▶ ក្នុងការបង្កាត់ខ្លួនត្រូវបានធ្វើឡើងតាមការបង្កាត់ខ្លួន។ ការបង្កាត់ខ្លួនត្រូវបានធ្វើឡើងតាមការបង្កាត់ខ្លួន។ ការបង្កាត់ខ្លួនត្រូវបានធ្វើឡើងតាមការបង្កាត់ខ្លួន។
- ▶ ក្នុងការបង្កាត់ខ្លួនត្រូវបានធ្វើឡើងតាមការបង្កាត់ខ្លួន។ ក្នុងការបង្កាត់ខ្លួនត្រូវបានធ្វើឡើងតាមការបង្កាត់ខ្លួន។ ក្នុងការបង្កាត់ខ្លួនត្រូវបានធ្វើឡើងតាមការបង្កាត់ខ្លួន។

age ឬជាអត្ថបន្ឌិតនៃការបង្កាត់ខ្លួន។ ក្នុងការបង្កាត់ខ្លួនត្រូវបានធ្វើឡើងតាមការបង្កាត់ខ្លួន។

```
class Cat:
    def __init__(self, name, age):
        self._name = name
        self._age = age
    ...
    def set_age(self, age):
        if age > 0:
            self._age = age
```

setter : ផ្តល់ឱ្យការពិន្ទុរបស់ខ្លួន។ ការបង្កាត់ខ្លួនត្រូវបានធ្វើឡើងតាមការបង្កាត់ខ្លួន។

2. ពិវត្តាំនើងការអេក្រាសក្នុងវត្ថុ (Object's identity operator : is, is not)

2.1. តើយឱ្យវិគារបញ្ជីការណ៍ដោយលម្អិតខ្លួន (Referring identical object via object's id)

| ពិវត្តា/មិនការឡកខស់វត្ថុ: is, is not

- ▶ ຕົວດໍາເນີນການ ສິ່ງຄືນຄ່າເປັນ True ຖ້າສອງ instance ຄ້າຍຄືກັນ (ນັ້ນຄືຖ້າຕົວທີ່ກຳເນີນການສອງຕົວອ້າງອີງວັດຖຸດຽວກັນ) ແລະ ສິ່ງຄືນຄ່າ False ຖ້າຫຼັງສອງບໍ່ຄ້າຍກັນ

```
1 list_a = [10, 20, 30]                      # list_a which refers a list object
2 list_b = [10, 20, 30]                      # list_b which refers a list object
3 if list_a is list_b:                         # Verifies if the two list objects are identical
4     print('list_a is list_b')
5 else:
6     print('list_a is not list_b')
```

2. ពិវត្តាំងនៃការអេក្រាសក្នុងវត្ថុ (Object's identity operator : is, is not)

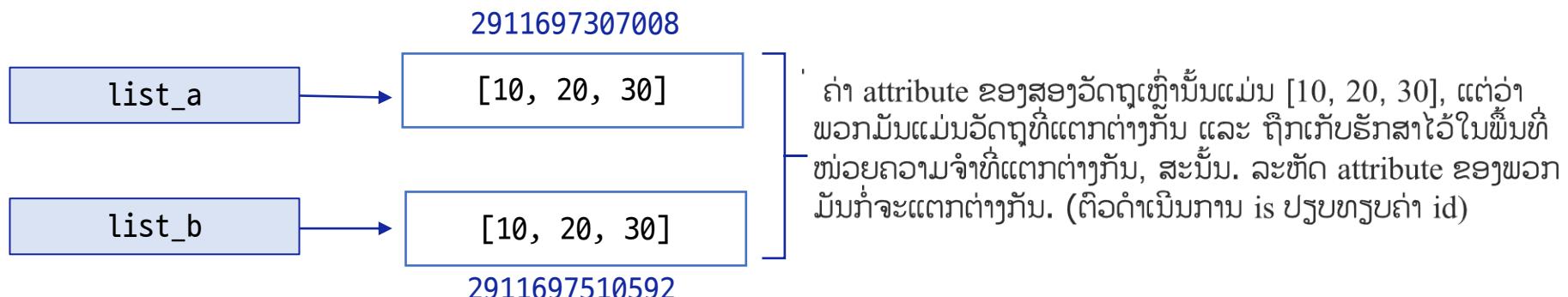
2.1. ອ້າງອີງວັດຖຸທີ່ຄ້າຍກັນຜ່ານລະຫັດຂອງວັດຖຸ

- | ກວດສອບຜົນໄດ້ຮັບດ້ວຍການພິມ (list_a) ແລະ id(list_b)

```
1 print('id(list_a) = ',id(list_a),', id(list_b) = ',id(list_b))
```

`id(list_a) = 2911697307008 , id(list_b) = 2911697510592`

- | ภาระจะผันได้รับ list_a และ list_b มีความแตกต่างกัน ทางด้านค่า id



list a is list b = False

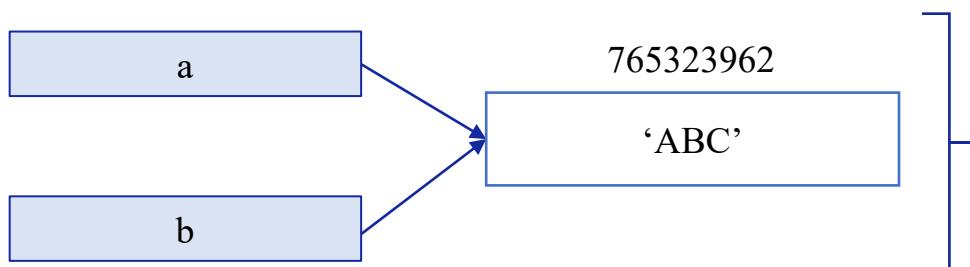
2. ពិវត្តាំនើងការអេក្រាសក្នុងវត្ថុ (Object's identity operator : is, is not)

2.1. ອ້າງອີງວັດຖຸທີ່ຄ້າຍກັນຜ່ານລະຫັດຂອງວັດຖຸ

a และ b ในโຄດลຸ່ມນີ້ໝາຍເຖິງ string ‘ABC’ ໃນນີ້ a ແລະ b ອ້າງອີງ object ດຽວກັນ, ບໍລິກັບ lists ຍ້ອນຫຍັງ? ເນື່ອງຈາກ string ເປັນປະເພດຂໍມູນທີ່ບໍ່ປ່ຽນແປງ (immutable) ຈຶ່ງອ້າງອີງໄປທີ່ object ດຽວກັນ.

```
1 a = 'ABC'                                # Variable a that refers a string object
2 b = 'ABC'                                # Variable b that refers a string object
3 if a is b:                                # Compares if the string object a and b are identical
4     print('a is b')                         # The string object a and b refer the same object
5 else:
6     print('a is not b')
```

a is b



object str ຂອງ Python ເກັບຮັກສາ object string ໃນຕາຕະລາງການເກັບຮັກສາ. ຫຼັງຈາກນີ້ນ, ຖ້າ ມັນກຳນົດ string ດຽວກັນກັບ a ແລະ b, ມັນຈະ ອ້າງອີງຕໍາແໜ່ງ ສະຖານທີ່ເກັບດຽວເພື່ອຊ່ວຍປະຢັດ ທີ່ໄດ້ຫຼາໄວ່ຄວບມາຈຶ່ງ

3. Class และ special method

3.1. ຄວາມຈໍາເປັນຂອງ special method

ສ້າງ class ຊື່ Vector 2D ທີ່ສະແດງເວັກຕີ ສອງມືຕີ ແລະ ສັງເກດຄວາມຈໍາເປັນຂອງ special method ໃນ class ນີ້. ຍ້ອນຫຍັງ $v1 + v2$ ຈຶ່ງສ້າງຂໍຟິດພາດໃນໂຄດຂ້າງລຸ່ມນີ້

```
1 class Vector2D:  
2     def __init__(self, x, y):  
3         self.x = x  
4         self.y = y  
5  
6 v1 = Vector2D(30, 40)          # + operator has not been defined in Vector2D : prints error  
7 v2 = Vector2D(10, 20)          # Vector2D의 + 연산이 정의되지 않았다: 오류 출력  
8 v3 = v1 + v2  
9 print('v1 + v2 = ', v3)
```

```
TypeError  
<ipython-input-37-e9d6e99fc219> in <module>  
      6 v1 = Vector2D(30, 40)          # + operator has not been defined in Vector2D : prints error  
      7 v2 = Vector2D(10, 20)          # Vector2D의 + 연산이 정의되지 않았다: 오류 출력  
----> 8 v3 = v1 + v2  
      9 print('v1 + v2 = ', v3)
```

TypeError: unsupported operand type(s) for +: 'Vector2D' and 'Vector2D'

3. Class และ special method

3.1. ຄວາມຈໍາເປັນຂອງ special method

ສະເໜດຂອງຂຶ້ນພາດຄື ເນື່ອງຈາກໂຄດບໍ່ໄດ້ລະບຸວ່າ class ຈະດຳເນີນການເພີ່ມຕື່ມພາຍໃນແນວໃດ
ຖ້າກໍານົດ method ທີ່ຊື່ add ແລະ ເພີ່ມ ອົງປະກອບ x, y ຂອງເວັກຕີທັງສອງ ຈາກນັ້ນສິ່ງຄືນຄ່າ Vector2D ທີ່ມີຄ່າຂອງສ່ວນປະກອບເຫຼື່ອນີ້
ເປັນຄ່າເລີ່ມຕົ້ນ

```
1 class Vector2D:
2     def __init__(self, x, y):
3         self.x = x
4         self.y = y
5     def __str__(self):
6         return "({}, {})".format(self.x, self.y)
7     def add(self, other):
8         return Vector2D(self.x + other.x, self.y + other.y)
9
10 v1 = Vector2D(30, 40)
11 v2 = Vector2D(10, 20)
12 v3 = v1.add(v2)
13 print('v1.add(v2) = ', v3)
```

v1.add(v2) = (40, 60)

Line 12

ພິມຜົນບວກຂອງສອງເວັກຕີໂດຍໃຊ້ເມຕອດ add v1.add(v2)
ການໃຊ້ຕົວດໍາເນີນການ ເຊັ່ນວ່າ +, - ຈະສະດວກກ່ວາ ເມຕອດ add

3. Class และ special method

3.2. กำหนด special method

- | `__add__` คือผู้ดูแลของวิธี加และการลบ, และ `__sub__` คือผู้ดูแลของการตัดและการลบ.
- | `v1 + v2` เรียกว่าการกับกันกับการอื่น ใช้ `v1.__add__(v2)`. เมื่อจาก `v1 + v2` ใช้ง่ายกว่า, พวกเขากำหนด `__add__` ให้ตัวดำเนินการ+ แทน เมตอด `__add__`.
- | ในทำนองนี้ Python จะนุยด้วยการใช้ฟังก์ชันใหม่เข้ากับฟังก์ชันเดิม ของ + ได้.

```
1 class Vector2D:  
2     def __init__(self, x, y):  
3         self.x = x  
4         self.y = y  
5     def __add__(self, other):  
6         return Vector2D(self.x + other.x, self.y + other.y)  
7     def __sub__(self, other):  
8         return Vector2D(self.x - other.x, self.y - other.y)  
9     def __str__(self):  
10        return "({}, {})".format(self.x, self.y)  
11  
12 v1 = Vector2D(30, 40)  
13 v2 = Vector2D(10, 20)  
14 v3 = v1 + v2  
15 print('v1 + v2 =', v3)  
16 v4 = v1 - v2  
17 print('v1 - v2 =', v4)
```

`v1 + v2 = (40, 60)`

`v1 - v2 = (20, 20)`

3. Class และ special method

3.2. กำหนด special method

```
1 class Vector2D:  
2     def __init__(self, x, y):  
3         self.x = x  
4         self.y = y  
5     def __add__(self, other):  
6         return Vector2D(self.x + other.x, self.y + other.y)  
7     def __sub__(self, other):  
8         return Vector2D(self.x - other.x, self.y - other.y)  
9     def __str__(self):  
10        return "({}, {})".format(self.x, self.y)  
11  
12 v1 = Vector2D(30, 40)  
13 v2 = Vector2D(10, 20)  
14 v3 = v1 + v2  
15 print('v1 + v2 =', v3)  
16 v4 = v1 - v2  
17 print('v1 - v2 =', v4)
```

v1 + v2 = (40, 60)

v1 - v2 = (20, 20)



Line 5-6

- กำหนด special method `__add__` เพื่อสั่งผินบวกของสองเวกเตอร์

3. Class และ special method

3.2. กำหนด special method

```
1 class Vector2D:  
2     def __init__(self, x, y):  
3         self.x = x  
4         self.y = y  
5     def __add__(self, other):  
6         return Vector2D(self.x + other.x, self.y + other.y)  
7     def __sub__(self, other):  
8         return Vector2D(self.x - other.x, self.y - other.y)  
9     def __str__(self):  
10        return "({}, {})".format(self.x, self.y)  
11  
12 v1 = Vector2D(30, 40)  
13 v2 = Vector2D(10, 20)  
14 v3 = v1 + v2  
15 print('v1 + v2 =', v3)  
16 v4 = v1 - v2  
17 print('v1 - v2 =', v4)
```

v1 + v2 = (40, 60)

v1 - v2 = (20, 20)

Line 14-15

- ต้องมีการกำหนดให้ $v1 + v2$ เป็นเรื่อง `__add__`
- การเข้าถึง $v1 + v2$ คือกับการเรียกใช้ method `v1.__add__(v2)`

3. Class และ special method

3.3. ឧបណិតពារូម្មរោង special method

| ពិនិត្យការសម្រេចរបស់ខ្លួន ដើម្បីលើកទិន្នន័យការប្រើប្រាស់ការសម្រេចរបស់ខ្លួន។

operator	Special method	functionality
<code>x + y</code>	<code>__add__(self, other)</code>	កំណត់រាយ ដើម្បីបន្ទាប់លើកទិន្នន័យ x និង y
<code>x - y</code>	<code>__sub__(self, other)</code>	កំណត់រាយ ដើម្បីបន្ទាប់លើកទិន្នន័យ x និង y
<code>x * y</code>	<code>__mul__(self, other)</code>	កំណត់រាយ ដើម្បីបន្ទាប់លើកទិន្នន័យ x និង y
<code>x ** y</code>	<code>__pow__(self, other)</code>	កំណត់រាយ ដើម្បីបន្ទាប់លើកទិន្នន័យ x ដោយ y.
<code>x / y</code>	<code>__truediv__(self, other)</code>	កំណត់រាយ ដើម្បីបន្ទាប់លើកទិន្នន័យ x ហានិថិយៈ y.
<code>x // y</code>	<code>__floordiv__(self, other)</code>	កំណត់រាយ ដើម្បីបន្ទាប់លើកទិន្នន័យ x ហានិថិយៈ y.
<code>x % y</code>	<code>__mod__(self, other)</code>	កំណត់រាយ ហាតិវសេដ ខ្លួន x ហានិថិយៈ y.
<code>+x</code>	<code>__pos__(self)</code>	x ត្រូវបានកំណត់ជាការបន្ទាប់លើកទិន្នន័យ
<code>-x</code>	<code>__neg__(self)</code>	x ត្រូវបានកំណត់ជាការបន្ទាប់លើកទិន្នន័យ

Note : Python 2 ទិន្នន័យ `_div_` និង `_truediv_`

3. Class และ special method

3.3. ឧបនិតតាំងក្រឡូវយោង special method

| ពិនិត្យលទ្ធផលរបស់ការប្រើប្រាស់ Python និង special method ដើម្បីសម្រេចការប្រើប្រាស់ការពិនិត្យក្នុងការបង្កើតរបៀបណាត់ខ្លួន។

operator	Special method	functionality
<code>x < y</code>	<code>__lt__(self, other)</code>	x ម៉ោយក់វា y បំ?
<code>x <= y</code>	<code>__le__(self, other)</code>	x ម៉ោយក់វា ឬ ឬផ្ទឹងៗក្នុង y បំ?
<code>x >= y</code>	<code>__ge__(self, other)</code>	x ឲ្យយក់វា ឬ ឬផ្ទឹងៗក្នុង y បំ?
<code>x > y</code>	<code>__gt__(self, other)</code>	x ឲ្យយក់វា y បំ?
<code>x == y</code>	<code>__eq__(self, other)</code>	x និង y មិត្តភក្តុះបំ?
<code>x != y</code>	<code>__ne__(self, other)</code>	x និង y មិត្តភក្តាហំ?

3. Class และ special methods

3.4. ຄວາມສໍາພັນລະຫວ່າງ built-in functions และ special method

- | Python ຍັງມີຟັງຊັນຈຳນວນຫລາຍເຊັ່ນ: len, float, int, str, abs, hash, iter ແລະ ອື່ນໆ.
- | ພັງຊັນ built_in ເຫຼື້ນີ້ແມ່ນເອີ້ນວ່າ special method ເຊັ່ນ: __len__, __float__, __int__, __str__, __abs__, __hash__, __iter__ ແລະ ອື່ນໆ.
- | ໃນນີ້, len built-in function ຖືກປະຕິບັດໂດຍການເອັນ __len__ special method

4. ຄວາມໝາຍຂອງ object, reference ແລະ assignment operator

4.1. ຄວາມໝາຍຂອງ object, reference ແລະ assignment operator

- | ສຶກສາປິດບາດຂອງ object, reference ແລະ assignment operator. ຕົວປ່ຽນເປັນພຽງແຕ່ຊື່ທີ່ມອບໃຫ້ກັບວັດຖຸໃນໜ່ວຍຄວາມຈໍາໃນຄໍາສັ່ງລຸ່ມນີ້, ມັນແມ່ນວິທີການໃຫ້ຄ່າຕົວເລກ 100 ຜ່ານຕົວປ່ຽນ `n` ໂດຍໃຊ້ `n = 100`.
- | ມັນສາມາດໄດ້ຮັບການກວດສອບໂດຍການເຮັດວຽກຂອງຟັງ `id`, ເຊິ່ງການກວດສອບ `id(100)` ແມ່ນຄືກັນກັບ `id(n)`.

```
1 n = 100  
2 id(100)
```

140719510270864

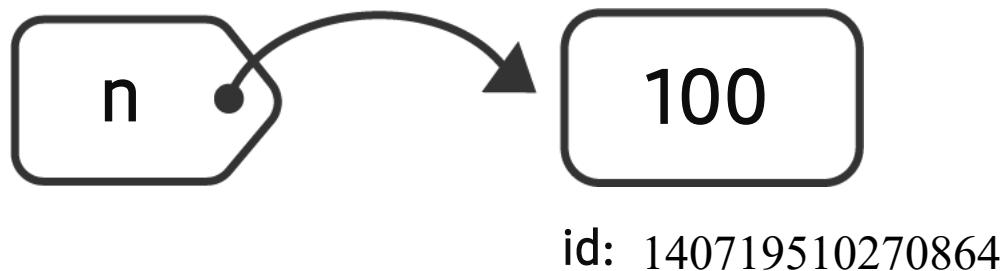
```
1 id(n)
```

140719510270864

4. ความหมายของ object, reference และ assignment operator

4.1. ความหมายของ object, reference และ assignment operator

- | ถ้ามีวัตถุที่มีค่า 100, ติ่งปีรุนอ้างอิง n สำลับวัตถุจะอ้างอิงเท็จวัตถุ 100 ที่ถูกกัน.
- | สามารถเข้าถึงวัตถุ 100 ผ่าน ติ่งปีรุน n.
- | ติ่งปีรุน m อีกตัวหนึ่งสามารถเข้าถึงวัตถุของติ่งปีรุน 100 โดยใช้ตัวดำเนินการกำหนด



4. ຄວາມໝາຍຂອງ object, reference ແລະ assignment operator

4.1. ຄວາມໝາຍຂອງ object, reference ແລະ assignment operator

- | Assignment operator = refers ແລະ re-refers object.
- | ທັງສອງຕົວປ່ຽນໝາຍເຖິງວັດຖຸດຽວກັນ..

```
1 n = 100  
2 m = n  
3 id(n)
```

140719510270864

```
1 id(m)
```

140719510270864

4. ຄວາມໝາຍຂອງ object, reference ແລະ assignment operator

4.1. ຄວາມໝາຍຂອງ object, reference ແລະ assignment operator

- | อ້າງອີງວັດຖຸດ້ວຍ attributes ທີ່ບໍ່ປ່ຽນຮູບໄດ້ແນວໃດ?
- | integer, real, string, Booleans ແລະ tuples ແມ່ນວັດຖຸທີ່ບໍ່ປ່ຽນແປງໄດ້. ໃນກໍລະນີຂ້າງລຸ່ມນີ້, m ແລະ n ຫມາຍເຖິງວັດຖຸດຽວກັນ 100.
 - ▶ n = 100
 - ▶ m = 100 # ທັງສອງແມ່ນຄືກັນ.
- | ກິນໄກດັ່ງກ່າວອະນຸຍາດໃຫ້ໃຊ້ໜ່ວຍຄວາມຈໍາຢ່າງມີປະສິດທິພາບ.

```
1 n = 100
2 m = 100
3 print(id(n))
4 print(id(m))
```

140736051753872
140736051753872

ຄວາມໝາຍຂອງ object, reference ແລະ assignment operator

- ເນື່ອງຈາກ m ແລະ n ທັງສອງອ້າງອີງຕົວເລກ 100, ສະນັ້ນ ids ຂອງທັງສອງ objects ແມ່ນເໜີອນກັນ.

4. ຄວາມໝາຍຂອງ object, reference และ assignment operator

4.2. assignment operator ຫມາຍເຖິງການອ້າງອີງເຖິງວັດຖຸອື່ນໆ

- | ຕົວປຽນ n ແລະ m ທີ່ອ້າງອີງເຖິງວັດຖຸສາມາດອ້າງອີງວັດຖຸໃໝ່ໄດ້.
- | ສະຖານະເບື້ອງຕົ້ນຂອງ n ແມ່ນ 100, ແລະ m ຜ່ານ $m = n$.
- | ຈາກນັ້ນ, m ແລະ n ອ້າງອີງຕົວປຽນອື່ນຜ່ານ $n = 200$.

```

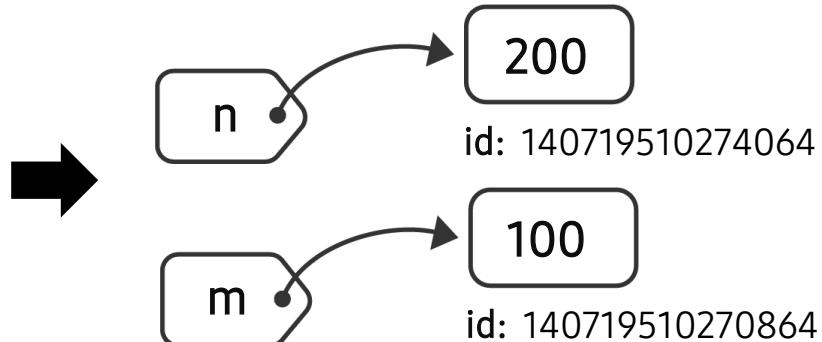
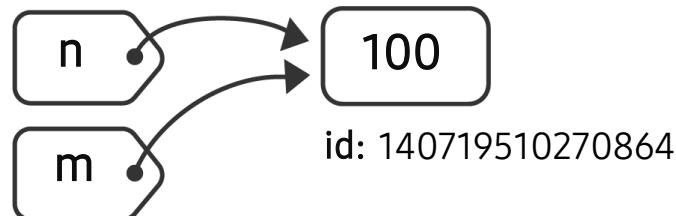
1 n = 100
2 m = n
3 n = 200
4 id(n)

```

140719510274064

```
1 id(m)
```

140719510270864



4. ຄວາມໝາຍຂອງ object, reference และ assignment operator

4.2. assignment operator ຫມາຍເຖິງການອ້າງອີງເຖິງວັດຖຸອື່ນໆ

- | Object assignment method และ ການຄໍານວນ $n = n + 1$
- | ໂຄດລຸ່ມນີ້ສະແດງໃຫ້ເຫັນວ່າ n ບໍ່ກຳນົດຄ່າໃໝ່ $n = n + 1$ ແລະ n ຄ່າເດີມ ແມ່ນ object ບໍ່ແຕກຕ່າງກັນ.

```
1 n = 100  
2 id(n)
```

140719510270864

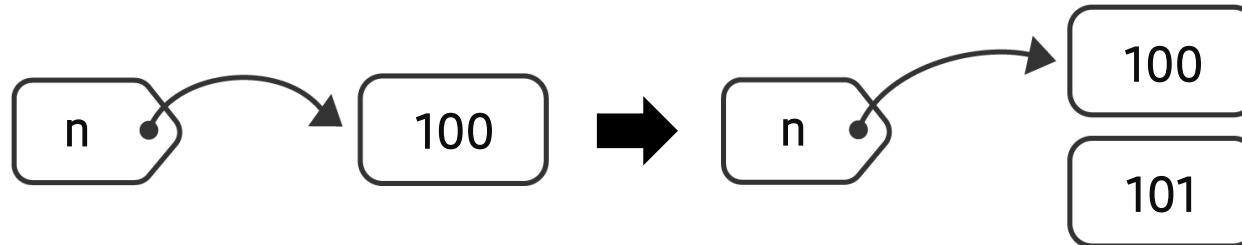
```
1 n = n + 1  
2 id(n)
```

140719510270896

4. ຄວາມໝາຍຂອງ object, reference และ assignment operator

4.2. assignment operator ຫມາຍເຖິງການອ້າງອີງເຖິງວັດຖຸອື່ນໆ

- | integer object ອ້າງອີງ ໂດຍ n ແມ່ນກຳນົດຕົວປູງ immutable



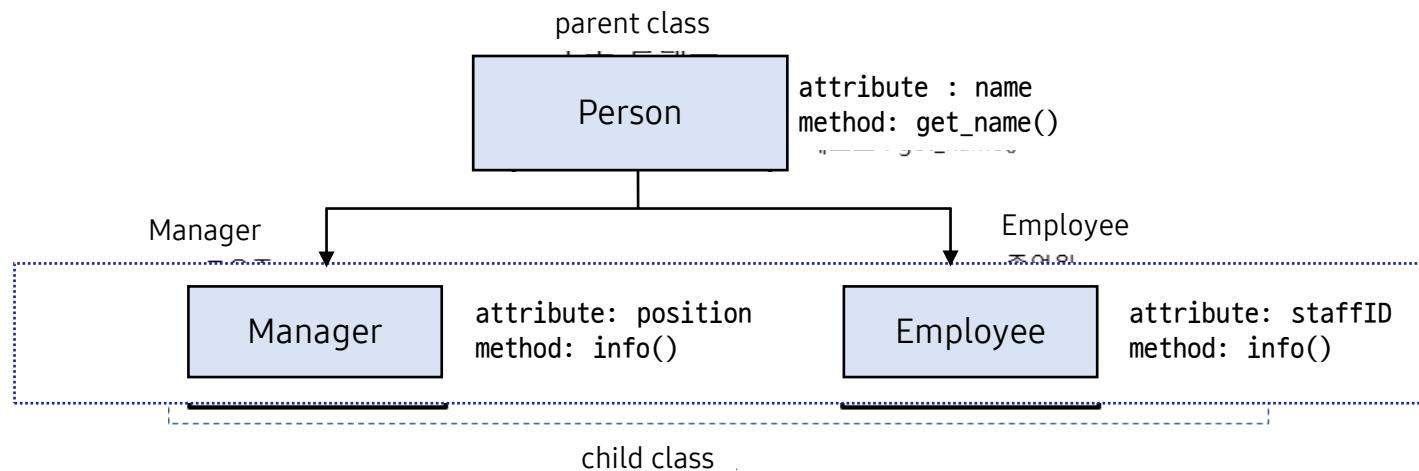
The integer object 100 is a garbage object that is not used.

- | ອັນນີ້ເຮັດໃຫ້ເສຍພື້ນທີ່ຄວາມຈໍາເພາະບໍ່ມີຕົວປຽນທີ່ອ້າງອີງເຖິງວັດຖຸນີ້ອີກຕ່ຳໄປ.
- | ວັດຖຸທີ່ບໍ່ສາມາດອ້າງອີງໄດ້ຢ້ອນການອ້າງອີງຫາຍໄປເຊັ່ນວ່າຂີ້ເຫຍື້ອ.
- | ຂັ້ນຕອນການຈັດການໜ່ວຍຄວາມຈໍາແມ່ນການລຶບຂີ້ເຫຍື້ອອອກຈາກໜ່ວຍຄວາມຈໍາ ເຊິ່ງເຊັ່ນວ່າການຮວບຮວມຂີ້ເຫຍື້ອ.

5. Class และ ภาระสืบทอด (Class and Inheritance)

5.1. Class ຂອງໂຄງສ້າງລຳດັບຊັ້ນ

- | ກວດເປົ່າ class ທີ່ມີໂຄງສ້າງຕາມລຳດັບຊັ້ນລຸ່ມນີ້.
- | ສາມາດສ້າງ class Manager ແລະ class Employee ໂດຍການສืທ່າງໂຄງສ້າງ class Person ເຊິ່ງເປັນ parent classs.
- | ເນື່ອງ Manager ແລະ Employee ລ້ວນແຕ່ເປັນມະນຸດ, ສອງ class ລູກຄວນມີຄຸນສົມບັດທີ່ແຕກຕ່າງໆກັນ.
- | ຖ້າ class Manager ແລະ class Employee ສືບຫອດຈາກ class Person, ສອງ class ຈະມີຕົວປ່ຽນ ແລະ function ເປັນຂອງຕົວເອງ.



5. Class และ ภาระสืบทอด (Class and Inheritance)

5.2. ภาระส้าง Class ຂອງໂຄງສ້າງລຳດັບຊັ້ນ

- | ກຳນົດໂຄງສ້າງທີ່ໄປຂອງ parent class ດັ່ງລຸ່ມນີ້
- | Class ນີ້ມີ attribute ທີ່ຊື່ name ແລະ method ທີ່ຊື່ get_name .

```
class Person:  
    def __init__(self, name):  
        self.name = name  
    def get_name(self):  
        return self.name
```

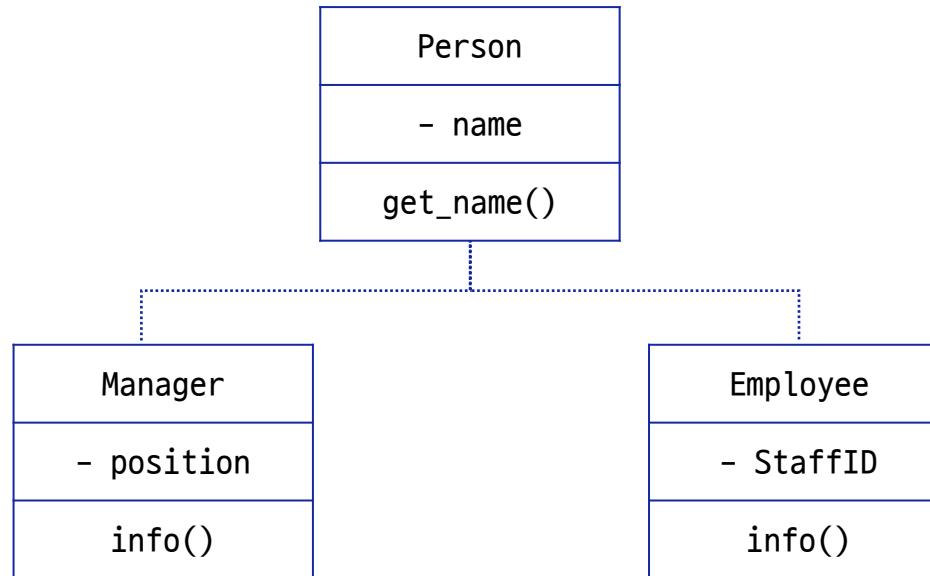
- | Class Employee ແລະ class Manager ສືບທອດຈາກ Person class ສາມາດກຳນົດດັ່ງສະແດງລຸ່ມນີ້.

```
class Manager(Person):  
    ...  
class Employee(Person):  
    ...
```

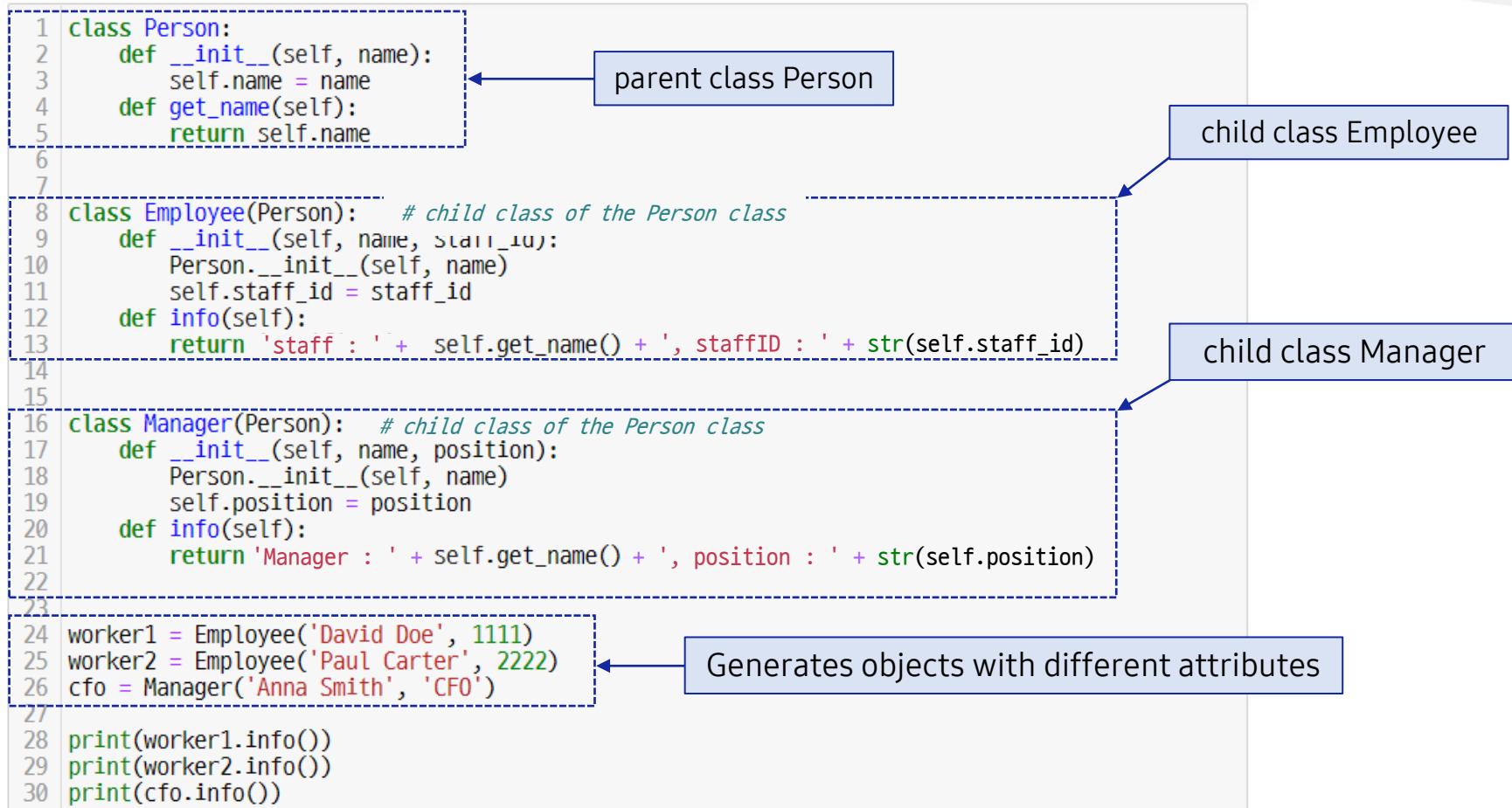
5. Class และ ภาระสืบทอด (Class and Inheritance)

5.2. ภาระสืบทอด Class ของโครงสร้างลำดับขั้น

| Slide ที่ดูไปจะแสดงโดยที่ใช้โครงสร้างลำดับขั้นของ class นี้



สามารถสืบทอด属性 name ที่มี name และ
method ที่มี get_name().



```
Employee : David Doe, staffID : 1111
Employee : Paul Carter, staffID : 2222
Manager : Anna Smith, position : CFO
```

| Pair programming



ການຝຶກປະຕິບັດການຂຽນໂປ້ແກ້ມເປັນຄູ່



ແນວທາງ, ກົມໄກ ແລະ ແຜນສູກເສີນ(Guideline, mechanisms & contingency plan)

ການກຽມໂປຣແກຣມຄຸ້ງກັບການກຳນົດແນວທາງແລະ ກົມໄກເພື່ອຊ່ວຍໃຫ້ນັກຮຽນຈັບຄຸ່ງຢ່າງເຫັນຈະສົມແລະເພື່ອໃຫ້ເຂົ້າຈັບຄຸ້ກັນຕົວຢ່າງເຊັ້ນ ນັກຮຽນຄວນພັດປ່ຽນກັນ ”ຄວບຄຸ່ມເມີ້າ” ການກຽມການທີ່ມີປະສິດທິພາບຈໍາເປັນຕ້ອງມີແຜນສູກເສີນໃນກໍລະນີທີ່ເພື່ອນຮ່ວມງານຄົນໜຶ່ງບໍ່ຢູ່ຫຼັດສິນໃຈທີ່ຈະບໍ່ເຂົ້າຮ່ວມດ້ວຍເຫດຜົນໄດ້ກໍຕາມ ໃນກໍລະນີເຫຼົ່ານີ້ສິ່ງສໍາຄັນຄືຕ້ອງເຮັດໃຫ້ຊັດເຈນວ່ານັກຮຽນທີ່ກະຕືລືລົ້ມຈະບໍ່ຖືກວິໄນເນື້ອຈາກການຈັບຄຸ່ບໍ່ດີພໍ

ก า น จ บ ค ล ท ท ถ า ย ถ ิ ก ว ั น , บ ร ่ จ า บ ี น ต ื อ ฯ ท ร ุ ว ั น , ค ว า မ ສ မ า ด บ ี น ค ล ร ่ ว ီ งาน

ການຂຽນໂປ່ງແນວຄຸ້ມສາມາດມີປະສິດທິພາບເມື່ອນັກຮຽນທີ່ຄ້າຢົກກັນ, ເຖິງວ່າບໍ່ຈໍາເປັນຕ້ອງມີຄວາມເຫຼົ່າຫຽມກັນ, ແຕ່ຄວາມສາມາດຖືກຈັບຄຸ້ມເປັນຄຸ້ຮ່ວມງານ. ການຈັບຄຸ້ມກຮຽນທີ່ບໍ່ກົງກັນມີຈະເຮັດໃຫ້ການມີສ່ວນຮ່ວມທີ່ບໍ່ສືມດຸນກັນ. ຄູສອນຕ້ອງເນັ້ນທັນກວ່າການຂຽນໂປ່ງແນວລມຄຸ້ມບໍ່ແມ່ນຍຸດທະສາດ "ແບ່ງປັນແລະເອົາຊະນະ", ແຕ່ເປັນຄວາມພະຍາຍາມຮ່ວມມືທີ່ເທົ່າລົງໃນທຸກໆຄວາມພະຍາຍາມສໍາລັບໂຄງການທັງໝົດ. ຄູຄວນຫຼົງກາເວັ້ນການຈັບຄຸ້ມກຮຽນທີ່ອ່ອນເອຫາຍກັບນັກຮຽນທີ່ເຂັ້ມແຂງຫາຍ.

ກະຕຸນນັກຮຽນໂດຍການໃຫ້ສິ່ງຈູ່ໃຈພິເສດ (Motivate students by offering extra incentives)

ການສະເໜີແຮງຈຸງໃຈພື້ນດັບສາມາດຊ່ວຍກະຕຸນນັກຮຽນໃຫ້ຈັບຄຸ້, ໂດຍສະເພາະກັບນັກຮຽນທີ່ກ້າວທຳນໍາ. ຄຸບາງຄົນພິບວ່າມັນເປັນປະໂຫຍດທີ່ຈະຮຽກຮ້ອງໃຫ້ນັກຮຽນຈັບຄຸ້ພຽງແຕ່ທີ່ນີ້ໜີ້ສອງວຽກ.



ການຝຶກປະຕິບັດການຂຽນໂປ້ແກ້ມເປັນຄຸ້



ប៉ុងក្រាសការងារកែវការងារកំរិះ (Prevent collaboration cheating)

ສິ່ງທ້າທາຍສໍາລັບຄຸນເມື່ອຊອກຫາວິທີທີ່ຈະປະເມີນຜົນໄດ້ຮັບຂອງບຸກຄົນ, ໃນຂະນະທີ່ນຳໃຊ້ຜົນປະໂຫຍດຂອງການຮ່ວມມື. ເຈົ້າຮູ້ໄດ້ແນວໃດວ່າ ນັກຮຽນຮຽນ ຫຼື ຖື່ກຕົວ? ຜູ້ຊ່ວງຊານແນະນຳໃຫ້ທີ່ບໍ່ທວນຄືນການອອກແບບຫຼັກສູດແລະການປະເມີນ, ພ້ອມທັງປີກສາຫາລືຢ່າງຈະເຈັ້ງແລະຊັດເຈນ ກັບນັກຮຽນກ່າວ່າວັກພິດຕິກຳທີ່ຈະຖືກຕົວມາວ່າຂຶ້ຕົວ. ຜູ້ຊ່ວງຊານຊູກຍູ້ໃຫ້ຄຸດເຮັດການມອບທາມຍໃຫ້ມີຄວາມທາມຍສໍາລັບນັກຮຽນແລະອະທິບາຍ ຄົນຄ່າຂອງສິ່ງທີ່ນັກຮຽນຈະຮຽນຮ້ອຍການເຮັດສໍາເລັດ.

ສະພາບແວດລ້ອມການຮຽນຮູ້ການຮ່ວມມື (Collaborative learning environment)

ສະພາບແວດລ້ອມການຮຽນຮູ້ຮ່ວມກັນເກີດຂຶ້ນໄດ້ທຸກເວລາທີ່ຜູ້ສອນຮຽກຮ້ອງໃຫ້ນັກຮຽນເຮັດວຽກຮ່ວມກັນໃນກົດຈະກຳການຮຽນຮູ້. ສະພາບແວດລ້ອມການຮຽນຮູ້ຮ່ວມກັນສາມາດມີສ່ວນຮ່ວມທັງກົດຈະກຳທີ່ເປັນທາງການ ແລະ ບໍ່ເປັນທາງການ ແລະ ອາດຈະບໍລວມເຖິງການປະເມີນໂດຍກິງ. ຕົວຢ່າງ, ນັກສຶກສາຄຸ້ງເຮັດວຽກກ່ຽວກັບການມອບທາມຍາການຂຽນໂປ່ງແງ້ມ; ນັກສຶກສາຕ່ຳມົງນ້ອຍໆສືນທະນາຄໍາຕາຍບໍ່ທີ່ເປັນໄປໄດ້ກ່ຽບຕໍ່ຖາມຂອງອາຈານໃນລະຫວ່າງການບັນຍາຍ; ແລະ ນັກຮຽນເຮັດວຽກຮ່ວມກັນນອກທ້ອງຮຽນເພື່ອຮຽນຮູ້ແນວຄວາມຄິດໃໝ່. ການຮຽນຮູ້ການຮ່ວມມືແມ່ນແຕກຕ່າງຈາກໂຄງການທີ່ນັກຮຽນ "ແບ່ງປັນແລະ ເອົາຊະນະ." ເມື່ອນັກຮຽນແບ່ງວຽກ, ແຕ່ລະຄົນຮັບຜິດຊອບພຽງແຕ່ສ່ວນໜຶ່ງຂອງການແກ້ໄຂບັນຫາແລະມີໂຄງກາດຈຳກັດຫຼາຍສໍາລັບການເຮັດວຽກຜ່ານບັນຫາກັບຄືນອື່ນ. ໃນສະພາບແວດລ້ອມຮ່ວມມື, ນັກຮຽນມີສ່ວນຮ່ວມໃນການສືນທະນາທາງປັນຍາກັບກັນແລະກັນ.

Q1.

ສ້າງ class ຫຼື student ທີ່ມີຟັງຊັ້ນການຮັດວຽກ ດັ່ງຕໍ່ໄປນີ້

ໃນ class ນີ້ແມ່ນໃຊ້ຮັດ Quiz ສາມວິຊາ ຄື: ພາສາອັງກິດ, ຄນິດສາດ ແລະ ວິທະຍາສາດ ໂດຍໃຫ້ປ້ອນຄະແນນຂອງ Quiz ທາງເປັນພິມ. ໃຫ້ສ້າງ instance ໂດຍນຳໃຊ້ class ນີ້ ເຊິ່ງມີ attributes ແລະ actions (methods) ດັ່ງຕໍ່ໄປນີ້

attributes

name : student name stored in string type.

student_id : student ID such as s2020001 stored in 8-digit integers.

eng_quiz : student's English quiz core stored in list format.

math_quiz : student's math quiz score stored in list format.

science_quiz : student's science quiz score stored in list format.

actions(methods)

`__init__` : initializes with the student's name and studentID.

`__str__` : returns the student's name, studentID and quiz scores as strings

`set_eng_quiz` : sets the student's English quiz.

`set_math_quiz` : sets the student's mathematics quiz.

`set_science_quiz` : sets the student's science quiz.

`get_name` : returns the student's name

`get_student_id` : returns the studentID

`get_eng_quiz` : returns the student's English quiz

`get_math_quiz` : returns the student's math quiz

`get_science_quiz` : returns the student's science quiz

`get_total_score` : returns the student's total quiz score

`get_avg_score` : returns the student's average quiz score

Q1.

Output example

```
Enter the student's name : David Doe
Enter the student's ID : 20213093
Enter the student's English quiz score : 90
Enter the student's mathematics quiz score : 95
Enter the student's science quiz score : 100
Name : David Doe, ID : 20213093
English quiz score : 90, Mathematics quiz score : 95
Science quiz score : 100,
Total : 285, Average : 95.0
```

A photograph of a person working at a desk. They are wearing an orange long-sleeved shirt and are holding a brown paper coffee cup with a black lid in their right hand. Their left hand is resting on a black computer keyboard. In the background, there are two computer monitors displaying code or text. On the desk, there is also a stack of papers and a small notepad with a pen. The scene is lit with warm, golden light.

End of
Document



Together for Tomorrow! Enabling People

Education for Future Generations

©2021 SAMSUNG. All rights reserved.

Samsung Electronics Corporate Citizenship Office holds the copyright of book.

This book is a literary property protected by copyright law so reprint and reproduction without permission are prohibited.

To use this book other than the curriculum of Samsung innovation Campus or to use the entire or part of this book, you must receive written consent from copyright holder.