

Unit 20.

# Closure

## Learning objectives

- ✓ ເຊົ້າໃຈ ແລະ ນຳໃຊ້ຂອບເຂດຕົວປ່ຽນໂດຍການເຂົ້າໃຈກົດຂອບເຂດຂອງຕົວປ່ຽນ.
- ✓ ເລີ່ມໃຊ້ ແລະ ປະຕິບັດໜຳທີ່ພາຍໃນຟັງຊັນໂດຍຜ່ານຟັງຊັນນອກ ແລະ ພັງຊັນຊ້ອນກັນ.
- ✓ ເຊົ້າໃຈແນວຄວາມຄິດຂອງຕົວປ່ຽນພາຍໃນຟັງຊັນ ແລະ ຕົວປ່ຽນພາຍນອກຟັງຊັນ. ນອກຈາກນີ້, ກວດສອບຕົວປ່ຽນທີ່ໄກ້ຖູງທີ່ສຸດໂດຍໃຊ້ keyword ແຕ່ບໍ່ແມ່ຕົວປ່ຽນໃນຟັງຊັນ.
- ✓ ສິ່ງຄືນການອ້າງອີງຂອງຟັງຊັນພາຍໃນ ທີ່ມີຢູ່ພາຍໃນຟັງຊັນໂດຍການ closure
- ✓ ສ້າງ closure ໂດຍໃຊ້ຟັງຊັນ lambda.
- ✓ ແປງຕົວປ່ຽນພາຍໃນຂອງການ closure ໂດຍໃຊ້ nonlocal.

## Learning overview

- ✓ ກຽນຮູ້ຂອບເຂດທີ່ຖືກຕ້ອງຂອງຫົວປ່ຽນ Python.
- ✓ ສ້າງ nested-function ແລະ ເອັນຝັງຊັ້ນພາຍໃນຝັງຊັ້ນ.
- ✓ ກຽນຮູ້ວິທີການໃຊ້ nonlocal ເພື່ອຊອກຫາຫົວປ່ຽນພາຍໃນທີ່ໄກ້ທີ່ສຸດ.
- ✓ ເຊົ້າໃຈວ່າເປັນຫຍັງ closure ແມ່ນມີຄວາມຈໍາເປັນ ແລະ ວິທີການກຳນົດມັນ.
- ✓ ກຽນຮູ້ວິທີການກຳນົດ closure ໂດຍເຢົາ lambda.
- ✓ ກຽນຮູ້ວິທີປ່ຽນຫົວປ່ຽນພາຍໃນຂອງ closure.

## Concepts You Will Need to Know From Previous Units

- ✓ ການນຳໃຊ້ຝັງຊັ້ນ lambda.
- ✓ ການນຳໃຊ້ filter, map, reduce ແລະ ພັງຊັ້ນພາຍໃນ Python ເຊິ່ງຝັງຊັ້ນ lambda ສາມາດຖືກນຳໃຊ້ໄດ້ງ່າຍ.
- ✓ ຄວາມສາມາດໃນການຜ່ານມູນຄ່າທີ່ມີປະສິດທິພາບໂດຍໃຊ້ variable parameters, basic parameters ແລະ keyword parameters..

## Keywords

scoping rule

nested function

local variable and  
global variable

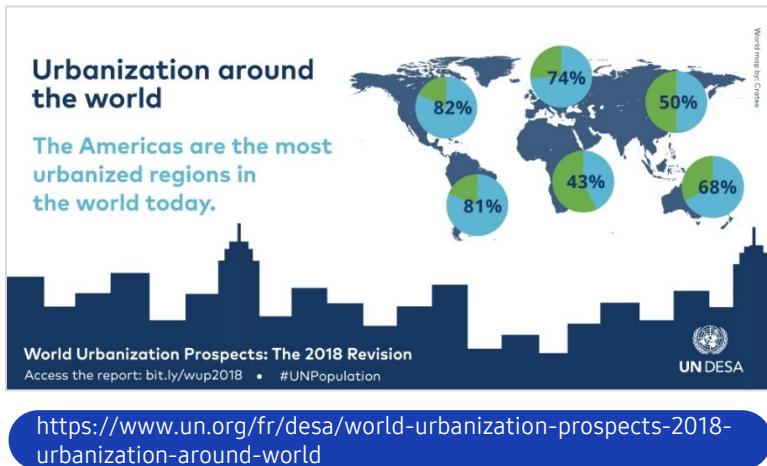
closure

nonlocal

# Mission

## 1. Real world problem

### 1.1. ບັນຫາດ້ານອຸດສາຫະກຳ: ຄວາມແຮອດໃນຕົວເມືອງ



- ▶ ໃນຫຼາຍປະເທດ, ຄວາມແຮອດໃນຕົວເມືອງ ເນື່ອງມາຈາກການພັດທະນາທາງດ້ານອຸດສະຫະກຳເພີ່ມຂຶ້ນຢ່າງວ່ອງໄວ.
- ▶ ໃນປີ 2018, ປະມານ 80% ຂອງປະຊາກອນໃນສະຫະລັດອາໄສຢູ່ໃນຕົວເມືອງ. ດາວວ່າຈະບັນລຸ 90% ໃນປີ 2050.
- ▶ ເນື່ອງຈາກຄວາມແຮອດຫຼາຍເກີນໄປຈະຮັດໃຫ້ເກີດອຳນະພາດໃນການເຮັດວຽກເຊັ່ນດຽວກັບ ກິດຈະກຳທາງດ້ານເສດຖະກິດ. ຫຼາຍປະເທດກຳລັງປະສິບບັນຫາທາງດ້ານມິນລະພິດທາງອາກາດ, ການຈໍລະຈອນ, ຫໍຢູ່ອາໄສ ແລະ ອື່ນງ.
- ▶ ໃນ mission ນີ້, ພວກເຮົາຈະວິຄາະຂໍ້ມູນຕົວເມືອງ ແລະ ປະຊາກອນເພື່ອຄົ້ນຄວາມແຮອດຂອງປະຊາກອນໃນຕົວເມືອງ.

## 1. Real world problem

### 1.1. ບັນຫາການຫັນເປັນອຸດສາຫະກຳ: ການຫັນເປັນໃຈກາງຂອງຕົວເມືອງ



- ▶ ການຫັນເປັນຕົວເມືອງໝາຍເຖິງຄວາມໝາເໜີນຂອງປະຊາກອນໃນຕົວເມືອງ ແລະ ຜົນໄດ້ຮັບຂອງການປ່ຽນແປງຂອງທ້ອງຖິ່ນ ແລະ ສັງຄົມ. ມັນຢັງໝາຍເຖິງການເພີ່ມຂຶ້ນຂອງປະຊາກອນໃນເມືອງ.
- ▶ ການເຄື່ອນຍ້າຍປະຊາກອນໃນຊົນນະບົດໄປສູ່ຕົວເມືອງເປັນລັກສະນະການເຄື່ອນຍ້າຍຕົວເມືອງ.
- ▶ ຖ້າພິຈາລະນາອັດຕາສ່ວນຂອງປະຊາກອນຂອງຕົວເມືອງທີ່ມີຫຼາຍກ່ວາ 100,000 ກັບປະຊາກອນໂລກເປັນດັດຊະນີຂອງຕົວເມືອງ, ມັນແມ່ນ 1.7% ໃນ ປີ 1800 ແລະ ເພີ່ມຂຶ້ນເປັນ 13% ໃນ ປີ 1950.
- ▶ ໃນໄລຍະ 150 ປີຜ່ານມາ, ປະຊາກອນໃນຕົວເມືອງເພີ່ມຂຶ້ນ 20 ເທົ່າ ໃນຂະນະທີ່ປະຊາກອນໂລກເພີ່ມຂຶ້ນພຽງແຕ່ 2.5 ເທົ່າ. ໃນປີ 1960, ອັດຕາສ່ວນໄດ້ກາຍເປັນ 20.1%.
- ▶ ດັ່ງນັ້ນ, ຄວາມເຂັ້ມຂຸ້ນຂອງປະຊາກອນໃນຕົວເມືອງແມ່ນເຫັນໄດ້ຊັດເຈນບໍ່ພຽງແຕ່ໃນປະເທດທີ່ພັດທະນາເລື່ອ, ແຕ່ຍັງຢູ່ໃນພາກພື້ນອາຊີຕາເວັນອອກສຽງໃຕ້, ອາເມລີກາລາຕິນ.

## 2. Solution

### 2.1. ເພີ່ມຕີມກ່ຽວກັບການຫັນເປັນຕົວເມືອງ

| ອ້າງເຖິງ <https://en.wikipedia.org/wiki/Urbanization>.



### 3. Mission

#### 3.1 ການວິຄາະຕົວເມືອງ



- ▶ ພວກເຮົາກໍາລັງວິຄາະປະເທດໃນອາຊີ. ພວກເຮົາຈະສ້າງຂໍ້ມູນປະຊາກອນໃນສື່ເມືອງໃຫຍ່ຂອງປະເທດດັ່ງກ່າວ ແລະ ວິຄາະຂໍ້ມູນ.
  - ▶ ສ້າງວັດຈະນານຸກົມທີ່ມີຊື່ city\_pop ເຊິ່ງປະກອບດ້ວຍຊື່ ແລະ ຈຳນວນປະຊາກອນ ເຊັ່ນ: A, B, C, D. ເປັນ ຂໍ້ມູນແບບ Dictionary ເຊິ່ງຂໍ້ມູນເລົ່ານີ້ຖືກເກັບໄວ້ດັ່ງຕໍ່ໄປນີ້.
  - ▶ ສະຖິຕິປະຊາກອນ (ຫົວໜ່ວຍ: ຫຼາຍພັນຄົນ)
    - A = 9765
    - B = 3441
    - C=2954
    - D=1531
  - ▶ ສ້າງຟັງຊັນ urban ເພື່ອພິມຈຳນວນປະຊາກອນສູງສຸດ ແລະ ຕໍ່ສຸດ, ຄວາມແຕກຕ່າງຂອງປະຊາກອນສູງສຸດ ແລະ ຕໍ່ສຸດ, ພ້ອມທັງຄ່າສະເລ່ຍຂອງປະຊາກອນໃນສື່ເມືອງນີ້.

### 3. Mission

#### 3.2. ຜົນການຈັດຕັ້ງປະຕິບັດການວິຄາະຕົວເມືອງ

```
1 def urban(city):
2     global max_pop, min_pop, pop_sum
3     n = 0
4     for name, pop in city.items(): # obtains maximum and minimum value by iterating in the loop statement
5         if pop > max_pop:
6             max_pop = pop
7         if pop < min_pop:
8             min_pop = pop
9         pop_sum += pop
10        n += 1
11    print('maximum population: ', max_pop)
12    print('minimum population: ', min_pop)
13    print('difference between maximum population and minimum population:', max_pop - min_pop)
14    print('average population:', pop_sum / n)
15
16 max_pop = 0
17 min_pop = 1000000
18 pop_sum = 0
19 # stores population statistics (unit : thousands) data in a dictionary
20 city_pop = { 'A':9765, 'B':3441, 'C':2954, 'D':1531 }
21 urban(city_pop)
```

Maximum population : 9765

Minimum population : 1531

Difference between maximum population and minimum population : 8234

Average population : 4422.75

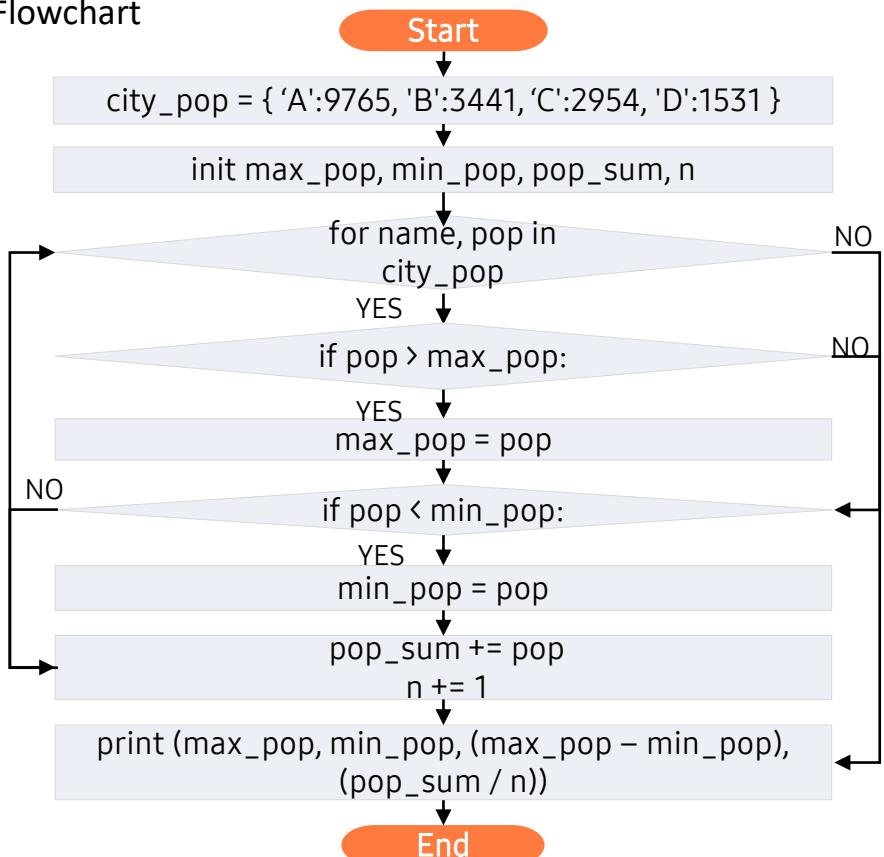
### 3. Mission

#### 3.3. Programming Plan

##### Pseudocode

```
[1] Start
[2] Store names and population of the cities in the dictionary
    city_pop
[3] Initialize max_pop, min_pop, pop_sum and the n value.
[4] for in the number of cities {
[5]     if the population of a city is higher than the population of
        another city {
[6]         The population of that city is saved as the highest
            population }
[7]     if the population of a city is lower than the population of
        another city
        { [8] The population of that city is saved as the lowest
            population.}
[9] Sum of all the accumulated population of the cities and the
    number of the cities are saved as pop_sum and n respectively.
[10] Print the maximum population, the minimum population, the
    difference of population and the average population.
[11] End
```

##### Flowchart



### 3. Mission

#### 3.4. Urbanization analysis final code

```
1 def urban(city):
2     global max_pop, min_pop, pop_sum
3     n = 0
4     for name, pop in city.items(): # obtains maximum and minimum value by iterating in the loop statement
5         if pop > max_pop:
6             max_pop = pop
7         if pop < min_pop:
8             min_pop = pop
9         pop_sum += pop
10        n += 1
11    print('maximum population: ', max_pop)
12    print('minimum population: ', min_pop)
13    print('difference between maximum population and minimum population:', max_pop - min_pop)
14    print('average population:', pop_sum / n)
15
16 max_pop = 0
17 min_pop = 1000000
18 pop_sum = 0
19 # stores population statistics (unit : thousands) data in a dictionary
20 city_pop = { 'A':9765, 'B':3441, 'C':2954, 'D':1531 }
21 urban(city_pop)
```

# | Key concept

## 1. ກົດໃນການກໍານົດຂອບເຂດຂອງ Python

### 1.1. ຂອບເຂດຂອງກົດ

- | Unit ມີແນະນຳຫຼາຍຝຶງຊັ້ນລະດັບສູງ ແລະ ແນວຄວາມຄິດນາມມະທຳ.
- | Unit ຈະເນັ້ນໜັກຂອບເຂດຂອງຕົວປ່ຽນ ຫຼື ພັງຊັ້ນ.
- | ພາສາງວຸນໂປຣແກຣມລວມທັງພາສາ Python ດີກໍານົດກົດການກໍານົດຂອບເຂດການເຂົ້າເຖິງ ແລະ ຄວາມຖືກຕ້ອງສໍາລັບຕົວປ່ຽນ ເຊິ່ງເອີ້ນວ່າ ກົດໃນການກໍານົດຂອບເຂດ (Scope Rule).
- | Python ຈໍາແນກຂອບເຂດຂອງຕົວປ່ຽນດັ່ງຕໍ່ໄປນີ້.
  - ▶ L(Local): ຕົວເປັນຕົວປ່ຽນພາຍໃນຝຶງຊັ້ນ
  - ▶ E(Enclosing Function Local): ຂອບເຂດຂອງຝຶງຊັ້ນທີ່ປະກອບດ້ວຍຝຶງຊັ້ນອື່ນ
    - ພາສາ Python ບໍ່ຄືກັບພາສາໂປຣແກຣມອື່ນ, ສາມາດກໍານົດຝຶງຊັ້ນພາຍໃນຝຶງຊັ້ນໄດ້.(inner function)
  - ▶ G(Global): ເປັນຕົວປ່ຽນພາຍນອກຝຶງຊັ້ນ
  - ▶ B(Built-in): ເປັນຕົວປ່ຽນ built-in namespace ທີ່ສ້າງມາ.

## 1. កិច្ចការកំណត់និតិវិធីនៃការបង្កើតខ្លួន Python

### 1.2. កិច្ចការកំណត់និតិវិធីបង្កើត

- | ក្នុងពីរក្រឹងការកំណត់និតិវិធីនៃការបង្កើតខ្លួន, មានភារយោងដែលមិនអាចបង្កើតបានទៀត។ ក្នុងពីរក្រឹងការកំណត់និតិវិធីនៃការបង្កើតខ្លួន, មានភារយោងដែលអាចបង្កើតបានទៀត។
- | ការបង្កើតបានទៀតគឺជាបញ្ជីនៃការបង្កើតខ្លួន។
- | B មែនជាបញ្ជីនៃការបង្កើតខ្លួន។
- | និងត្រូវបានបង្កើតខ្លួនដោយប្រើប្រាស់ការបង្កើតខ្លួន។

```
1 x = 10
2 y = 11          # corresponds to G
3 def foo():
4     x = 20      # foo function corresponds to L
5     def bar():
6         a = 30  # bar function corresponds to E
7         print(a, x, y) # each variable corresponds to L, E and G
8     bar()
9     x = 40
10    bar()
11
12 foo()
```

```
30 20 11
30 40 11
```

## 1. កិច្ចការកំណត់នូវបញ្ជីនៃ Python

### 1.2. កិច្ចការកំណត់នូវបញ្ជី

💡 Take note of TypeError that occurs in built-in functions

```
1 abs      # built-in function abs()  
<function abs(x, /)>
```

```
1 abs = 10 # global function abs()
```

```
1 abs(-5) # global function abs() blocks built-in function abs()
```

```
TypeError  
<ipython-input-13-cb24a21394fc> in <module>  
----> 1 abs(-5) # global function abs() blocks built-in function abs()
```

TypeError: 'int' object is not callable

```
1 del abs      # deletes global function  
2 print(abs(-10)) # built-in function abs appears  
  
10
```

- ព័ត៌មានលម្អិតនៃ built-in functions. ដើម្បីបង្កើត abs=10, តើ abs បែងចែងនៃ built-in function ផ្តល់អ្នកត្រូវបានបញ្ជី។
- ឱ្យដឹងថា abs ត្រូវបានបញ្ជីនៅក្នុង block មួយទៀត។ abs ត្រូវបានបញ្ជីនៅក្នុង global function ដែលបានបង្កើតឡើងដោយ del ឬ global function នៃ abs.

## 1. ກົດໃນການກໍານົດຂອບເຂດຂອງ Python

### 1.2. ກົດໃນການກໍານົດຂອບເຂດ

| ຕອນນີ້ສ້າງຕົວປ່ຽນນີ້ຊື່ counter ແລະ ກໍານົດ counter = 200 . ສັງເກດວ່າ 200 ຈະຖືກພິມອອກມາທັງພາຍໃນຝັງຊັ້ນ ແລະ ນອກຝັງຊັ້ນ ຫຼືບໍ່ ?

```
1 def print_counter():
2     counter = 200
3     print('counter =', counter) # counter value inside the function
4
5 counter = 100
6 print_counter()
7 print('counter =', counter)      # counter value outside the function
```

```
counter = 200
counter = 100
```

- | ເມື່ອກໍານົດຄ່າ counter = 200, ຈະສັງເກດວ່າ ຕົວປ່ຽນທີ່ປະກາດມາໃໝ່ຢູ່ນອກຝັງຊັ້ນ ຈະແຕກຕ່າງກັບຕົວປ່ຽນທີ່ປະກາດພາຍໃນຝັງຊັ້ນ.
- | ເຊັ່ນດຽວກັນ, ຕົວປ່ຽນທີ່ສ້າງຂຶ້ນພາຍໃນຝັງຊັ້ນເອີ້ນວ່າຕົວປ່ຽນພາຍໃນ. ຄ່າ 200 ອ້າງອີງໂດຍຕົວປ່ຽນພາຍໃນ, ສ່ວນຄ່າ 100 ຈະອ້າງອີງໂດຍຕົວປ່ຽນພາຍນອກ. ຄ່າຂອງຕົວປ່ຽນພາຍໃນຈະຫາຍໄປເມື່ອຝັງຊັ້ນເຮັດວຽກສໍາເລັດ.
- | ເພາະສະນັ້ນ, ຄ່າ 100 ຈຶ່ງມາແທນທີ່ຄ່າ 200 ແລະ ຖືກພິມອອກມາ.

## 1. កិច្ចការណ៍នឹងលទ្ធផលនៃ Python

### 1.3. ពិវប្បធយនី និង ពិវប្បធយនរាជធានី

| ក្រុមហ៊ុបពិវប្បធយនី ត្រូវបានគេប្រើប្រាស់ជាពិវប្បធយនី និង ពិវប្បធយនរាជធានី ដោយសារតម្លៃមីនី.

Local variable (L) ឬជាប្រើប្រាស់ជីវិតីការណ៍ និង ផ្តល់ឱ្យការងារនៃការងាររាជធានី។

```
1 def print_counter():
2     counter = 200
3     print('counter =', counter) # counter value inside the function
4
5 counter = 100
6 print_counter()
7 print('counter =', counter) # counter value outside the function
```

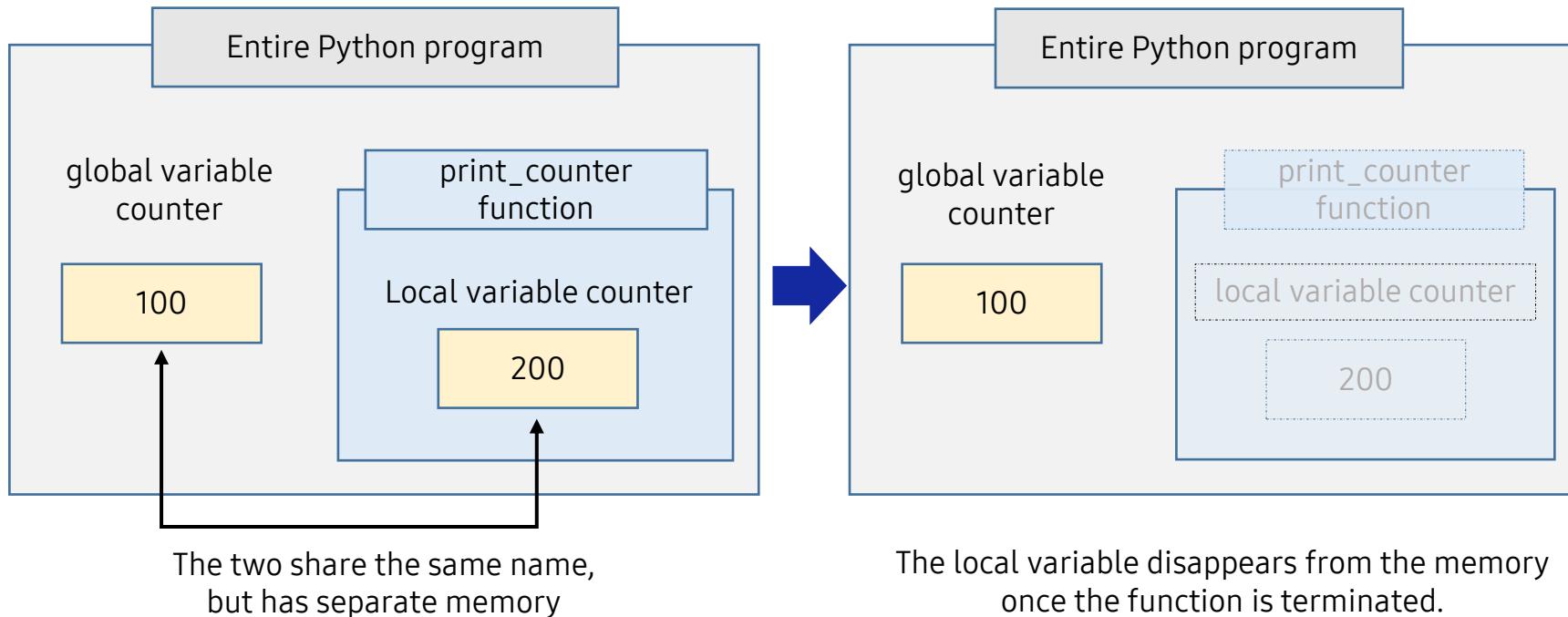
counter = 200  
counter = 100

Global variable(G) ឬជាប្រើប្រាស់ជីវិតីការណ៍ និង ផ្តល់ឱ្យការងារនៃការងាររាជធានី។

## 1. កិច្ចការណ៍ធម្មជាមួយនឹង Python

### 1.3. ពូប្បុនធយើន និង ពូប្បុនធយើនរាយការ

ឧប្បរដ្ឋ ពូប្បុនធយើន និង ពូប្បុនធយើនរាយការ នឹងប្រាកាសការងារ និងការបង្កើតផល។



## 1. ກົດໃນການກຳນົດຂອບເຂດຂອງ Python

### 1.3. ຕົວປ່ຽນພາຍໃນ ແລະ ຕົວປ່ຽນພາຍນອກຟັງຊັນ

 ສຸມໃສ່ global keyword ຫມາຍເຖິງການໃຊ້ຕົວປ່ຽນພາຍນອກຟັງຊັນ.

| ເຮົາຈະເອີ້ນຕົວປ່ຽນພາຍນອກທີ່ຊື່ counter ໂດຍປັດສະຈາກການສ້າງຕົວປ່ຽນໃໝ່ທີ່ຢູ່ນອກຟັງຊັນໄດ້ແນວໃດ? ນຳໃຊ້ຄຳສັບ global ໃສຕໍ່ໜ້າຕົວປ່ຽນ ເພື່ອໃຫ້ສາມາດນຳໃຊ້ຕົວປ່ຽນດັ່ງກ່າວພາຍນອກຟັງຊັນໄດ້. ດັ່ງສະແດງລຸ່ມນີ້

```
1 def print_counter():
2     global counter      # declaration to use the global variable counter outside the function
3     counter = 200
4     print('counter =', counter) # counter value inside the function
5
6 counter = 100
7 print_counter()
8 print('counter =', counter)      # counter value outside the function
```

```
counter = 200
counter = 200
```

| ດ້ວຍການປະກາດດັ່ງກ່າວ, ຕົວປ່ຽນ ພາຍນອກຟັງຊັນສາມາດປ່ຽນຄ່າເປັນ 200 ເນື້ອມີການກຳນົດ counter = 200 ພາຍໃນ ພັງຊັນ.

## 1. กິດໃນການກຳນົດຂອບເຂດຂອງ Python

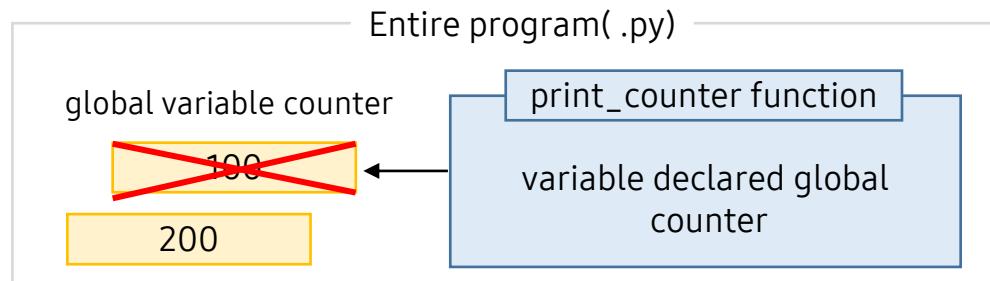
### 1.3. ຕົວປັບປຸງພາຍໃນ ແລະ ຕົວປັບປຸງພາຍນອກຟັງຊັ້ນ

| ກຳສັບ global ປັບປຸງຄ່າຢູ່ນອກຟັງຊັ້ນ.

ເນື້ອປະກາດເປັນ global, ຕົວປັບປຸງ counter ຈະກາຍເປັນຕົວປັບປຸງພາຍນອກ, ຈະບໍ່ແມ່ນຕົວປັບປຸງພາຍໃນ. ຖ້າມີການປັບປຸງຄ່າຕົວປັບປຸງພາຍໃນຟັງຊັ້ນ, ຄ່າໃນຕົວປັບປຸງດັ່ງກ່າວກໍຈະຖືກປັບປຸງຕາມ.

```
1 def print_counter():
2     global counter      # declaration to use the global variable counter outside the function
3     counter = 200
4     print('counter =', counter)  # counter value inside the function
5
6 counter = 100
7 print_counter()
8 print('counter =', counter)      # counter value outside the function
```

counter = 200  
counter = 200



The variable declared global references the global variable

# 1. ກິດໃນການກຳນົດຂອບເຂດຂອງ Python

## 1.4. First-class function

### | ສຶກສາ First-class function

- ▶ First-class function ສາມາດປະຕິບັດຄືກັບ Object ອື່ນໆ, ເຊັ່ນ ສ້າງ numbers, strings, classes ແລະ ເກັບໄວ້ໃນຕົວປັບປຸງ.
- ▶ ສາມາດສື່ງຜົງຊັນໄປທາ parameter, ແຊກຜົງຊັນໄປທາຕົວປັບປຸງອື່ນ ຫຼື ໃຊ້ມັນເພື່ອສື່ງຄ່າກັບຄືນ. ນອກຈາກນີ້ ຍັງສາມາດເກັບຜົງຊັນໄວ້ໃນປະເພດຂຶ້ມູນ ເຊັ່ນ: list ຫຼື dictionary. ຈາກ Code ດຳສັ່ງລຸ່ມນີ້ ເອັນໃຊ້ຜົງຊັນ callfunc ເຊິ່ງຮັບຜົງຊັນເປັນ parameter.

```
1 def callfunc(func):  
2     func()  
3  
4 def greet():  
5     print('Hello')  
6  
7 print('callfunc(greet) function call')  
8 callfunc(greet)
```

```
callfunc(greet) function call  
Hello
```

ແຊກຜົງຊັນ greet ເປັນ argument ສໍາລັບການເອັນໃຊ້ຜົງ  
ຊັນ callfunc.

# 1. កិច្ចការណ៍លទ្ធផលនៃការបង្កើតខ្លួន

## 1.4. First-class function

| ដូច្នេះ Python សាមសុគ្រែលដែលអាចរាយការជាអំពីការបង្កើតខ្លួន។

```
1 def plus(a, b):
2     return a + b
3 def minus(a, b):
4     return a - b
5 l_list = [plus, minus]
6 a = l_list[0](100, 200)
7 b = l_list[1](100, 200)
8 print('a =', a)
9 print('b =', b)
```

```
a = 300
b = -100
```

សាមសុគ្រែលដែលអាចរាយការជាអំពីការបង្កើតខ្លួន។

# 1. ກົດໃນການກຳນົດຂອບເຂດຂອງ Python

## 1.5. High-level functions ທີ່ຖືກສ້າງດ້ວຍ first-class functions

| ສາມາດຈັດການກັບວຽກທີ່ສັບຊ້ອນໂດຍການນຳໃຊ້ first-class functions

- ▶ ຜ່ານຟັງຊັນເປັນ argument ສໍາລັບຟັງຊັນອື່ນ. (ເກັບຟັງຊັນໄວ້ໃນຕົວປ່ຽນ)
- ▶ ຜ່ານຟັງຊັນເປັນຄ່າສົ່ງກັບຄືນ ຂອງຟັງຊັນອື່ນ. (ໃຊ້ຟັງຊັນເປັນຄ່າສົ່ງຄືນ)
- ▶ ເກັບຢັງຊັນໄວ້ໃນຕົວປ່ຽນ ຫຼື ໂຄງສ້າງຂຶ້ມູນ.

```
1 def add(a, b):  
2     return a + b  
3  
4 def f(g, a, b):  
5     return g(a,b)  
6 f(add, 3, 4)
```

7

| ເຮົາສາມາດນຳໃຊ້ຟັງຊັນເປັນ argument ຫຼື ສົ່ງຄ່າກັບຄືນ ດັ່ງສະແດງຂ້າງເທິງ, ເນື່ອງທັງໝົດຟັງຊັນໃນ Python ເປັນ first-class functions.

## 2. Outer Function ແລະ Nested Function

### 2.1. ຕົວຢ່າງ ການນຶຍາມ nested function ໃນ Python

ໃນ Python, ເຮົາສາມາດກຳນົດຝັງຊັນ italic, bold ເພື່ອຕີບແຕ່ງ ດັ່ງທີ່ສະແດງຢູ່ໃນ code ລຸ່ມນີ້ ແລະ ສາມາດຜ່ານຝັງຊັນ ດ້ວຍການສິ່ງຄ່າກັບຂອງຝັງຊັນ.

- ໃນຕົວຢ່າງລຸ່ມນີ້ ນຳໃຊ້ຄຸນສົມບັດຂອງ Python ດ້ວຍການນຶຍາມຝັງຊັນເຂົ້າໄປໃນຝັງຊັນ ແລະ ຜ່ານຝັງຊັນດ້ວຍການສິ່ງຄ່າຂອງຝັງຊັນກັບ.
- ເຮົາສາມາດສ້າງໂຄງສ້າງຕາມ code ລຸ່ມນີ້ໃນພາສາອື່ນໄດ້.
- ເຮົາສາມາດເຮັດໃຫ້ code ຄຳສັ່ງຢ່າຍຂຶ້ນໄດ້ ໂດຍຄຳສັ່ງຄວບຄຸມໜ້ອຍລົງ

```
1 def decorate(style = 'italic'):
2     def italic(s):
3         return '<i>' + s + '</i>'
4     def bold(s):
5         return '<b>' + s + '</b>'
6     if style == 'italic':
7         return italic
8     else:
9         return bold
10 dec = decorate()
11 print(dec('hello'))
12 dec2 = decorate('bold')
13 print(dec2('hello'))
```

ສາມາດເອີ້ນໃຊ້ຝັງຊັນພາຍໃນທີ່ຖືກປະກາດໄວ້ພາຍໃນຝັງຊັນ  
ດ້ວຍ nested-function.

```
<i>hello</i>
<b>hello</b>
```

## 2. Outer Function ແລະ Nested Function

### 2.2. ເຫດຜົນການນຳໃຊ້ Nested Function

- | ພັງຊັ້ນຊ່ອນກັນແມ່ນພັງຊັ້ນພາຍໃນພັງຊັ້ນ. ບໍ່ຄືກັບພັງຊັ້ນທີ່ຢູ່ດ້ານນອກ, ມັນສາມາດອ່ານຕົວປ່ຽນຂອງພັງຊັ້ນຫຼັກໄດ້ຢ່າງເອກະລາດ.
- | ເຮົາສາມາດຮັດໃຫ້ມັນຊັດເຈນໄດ້ດ້ວຍພັງຊັ້ນຊ່ອນກັນ.
- | ຢ່າງໃດກໍຕາມ, ເພື່ອຮັດໃຫ້ຈຸດປະສົງຊັດເຈນຂຶ້ນ, ສາມາດປະກາດພັງຊັ້ນຢູ່ນອກພັງຊັ້ນໄດ້ ດັ່ງສະແດງໃນ code ຄຳສັ່ງລຸ່ມນີ້.
- | Code ຄຳສັ່ງລຸ່ມນີ້ເຮັດວຽກຢ່າງຖືກຕ້ອງ.

```
1 def another_func():
2     print("hello")
3
4 def outer_func():
5     return another_func()
6
7 outer_func()
```

hello

## 2. Outer Function ແລະ Nested Function

### 2.2. ເຫດຜົນການນຳໃຊ້ Nested Function

- | ເຫດຜົນທີ່ສໍາຄັນກວ່າ ສໍາລັບການນຳໃຊ້ Nested Function ແມ່ນແນວຄົດທີ່ເອັນວ່າ closure.
- | ຫຼື ຫຼື ໃນຕູ້ອນໄຂທີ່ຈໍາເປັນໃນການດຳເນີນການ closure ແມ່ນການເອັນໃຊ້ Nested Function .
- | ແນວດລົດຂອງ Nested Function ຈະຮັດໃຫ້ເຂົ້າໃຈ closure.

### 3. ຂະບວນການຄົ້ນຫາຕົວປ່ຽນທີ່ບໍ່ແມ່ນຕົວປ່ຽນພາຍໃນ

#### 3.1. ຄໍາສັບ 'global'

| ຖ້າຕ້ອງການແກ້ໄຂຄ່າຕົວປ່ຽນ n1 ພາຍໃນຝັງຊັນ ຫຼັງຈາກປະກາດມັນເປັນຕົວປ່ຽນ global ເຊັ່ນ: global n1 ໃນນີ້ໜາຍຄວາມວ່າ “ ຈະໃຊ້ຕົວປ່ຽນພາຍນອກ n1, ບໍ່ແມ່ນ n1 ຢູ່ໃນຝັງຊັນ.”

```
1 n1 = 1 ←  
2 def func1():  
3     def func2():  
4         global n1  
5         n1 += 1  
6         print(n1) # print 2  
7     func2()  
8  
9 func1()
```

ຕົວປ່ຽນທີ່ປະກາດເປັນ global ຈະຄົ້ນຫາຕົວປ່ຽນທີ່ຢູ່  
ດ້ານນອກຝັງຊັນ ແລະ ນໍາໃຊ້ມັນ

2

### 3. ຂະບວນການຄົ້ນຫາຕົວປ່ຽນທີ່ບໍ່ແມ່ນຕົວປ່ຽນພາຍໃນ

#### 3.2. ຄວາມຈຳເປັນຂອງ nonlocal keyword

##### >NameError

```
1 def func1():
2     n2 = 1
3     def func2():
4         global n2
5         n2 += 1
6         print(n2) # error
7     func2()
8
9 func1()
```

```
NameError                                     Traceback (most recent call last)
<ipython-input-8-4a8342131d9b> in <module>
      7     func2()
      8
----> 9 func1()

NameError: name 'n2' is not defined
```

- ຖ້າຕ້ອງການອ້າງອີງຕົວປ່ຽນ global n2, ແຕ່ບໍ່ມີຕົວປ່ຽນ global n2 ທີ່ເປັນ ຕົວກາງ ຈະເກີດມີ NameError ເນື່ອງຈາກ n2 ບໍ່ໄດ້ປະກາດເປັນຕົວປ່ຽນ global.
- ກໍລະນີ້ໃຫ້ໃຊ້ nonlocal keyword ແທນ global keyword.

### 3. ຂະບວນການຄົ້ນຫາຕົວປ່ຽນທີ່ບໍ່ແມ່ນຕົວປ່ຽນພາຍໃນ

#### 3.2. ຄວາມຈຳເປັນຂອງ nonlocal keyword

```
1 def func1():
2     n3 = 1
3     def func2():
4         nonlocal n3
5         n3 += 1
6         print(n3)    # It's not error
7     func2()
8
9 func1()
```

nonlocal n3 variable is not a local variable.  
It is not a global variable either, so the nearest n3 variable is connected.

2

- | ນໍາໃຊ້ nonlocal keyword ເວລາປະກາດຕົວປ່ຽນ n3 ເຊິ່ງບໍ່ແມ່ນຕົວປ່ຽນພາຍໃນ ທັງບໍ່ແມ່ນຕົວປ່ຽນພາຍນອກໃນຂອບເຂດນີ້.
- | ຊ້າງກໍານົດ nonlocal n3 ດ້ວຍຕົວປ່ຽນທີ່ໃກ້ຄຽງ n3 ເຊິ່ງບໍ່ແມ່ນຕົວປ່ຽນພາຍໃນ, ບໍ່ມີບັນຫາ.
- | ການເຊື່ອມຕໍ່ຕົວປ່ຽນໄກ້ຄຽງ ເຮັດວ່າການຜູກມັດ.

### 3. ຂະບວນການຄົ້ນຫາຕົວປ່ຽນທີ່ບໍ່ແມ່ນຕົວປ່ຽນພາຍໃນ

#### 3.3. nonlocal keyword ແລະ binding

 ສຸມໃສ່ ເຮົາສາມາດຜູ້ກຳນົດກັບຕົວປ່ຽນທີ່ຕັ້ງຢູ່ນອກຟັງຊັ້ນໜຶ່ງລະດັບ ໂດຍໃຊ້ nonlocal.

| nonlocal : nonlocal keyword ທີ່ນີ້ປະກາດວ່າ x ບໍ່ແມ່ນຕົວປ່ຽນພາຍໃນ.

```

1 x = 20 # global variable
2 def f():
3     x = 40 ←
4     def g():
5         nonlocal x
6         x = 80
7     g()
8     print(x)
9
10 f()
11 print(x)

```

ຕົວປ່ຽນ nonlocal x ບໍ່ແມ່ນຕົວປ່ຽນພາຍໃນຟັງຊັ້ນ.  
ແຕ່ມັນກໍບໍ່ແມ່ນຕົວປ່ຽນພາຍນອກ, ດັ່ງນັ້ນທັງສອງຕົວປ່ຽນນີ້ຈຶ່ງຜູກຕິດກັນ.

80

20



Line 7, 8, 11

- ໃຊ້ nonlocal ເພື່ອດຳເນີນການກັບຟັງຊັ້ນ g.
- ພິມຄ່າ x ຈາກຟັງຊັ້ນ f. (ຕົວປ່ຽນຂອງຟັງຊັ້ນ g ບ່ຽນເປັນຄ່າ 80 ໃນ nonlocal.)
- ຫຼັງຈາກປະຕິບັດສໍາເລັດແລ້ວ, ພິມຄ່າໃນຕົວປ່ຽນ x. (ທີ່ຢູ່ນອກຟັງຊັ້ນທັງໝົດ ເທົ່າກັບ 20.)
- ເປັນການຜູກຕິດຕົວປ່ຽນ x ກັບຄ່າຈຶ່ງ.

### 3. ຂະບວນການຄົ້ນຫາຕົວປ່ຽນທີ່ບໍ່ແມ່ນຕົວປ່ຽນພາຍໃນ

#### 3.4. ຄວາມສໍາພັນລະຫວ່າງຕົວປ່ຽນ nonlocal ແລະ global

##### ❗ nonlocal ແລະ global variable

```
1 x = 70 # global variable
2 def f():
3     nonlocal x
4     x= 140
5
6 f()
7 print(x)
```

```
File "<ipython-input-33-5aa47a6c6d0e>", line 3
    nonlocal x
    ^
SyntaxError: no binding for nonlocal 'x' found
```

##### Line 3

- ກໍານົດພຽງແຕ່ຟັງຊັ້ນດຽວ ແລະ ປະກາດ nonlocal ພາຍໃນຟັງຊັ້ນ ເພື່ອຮັກສາຕົວປ່ຽນ global.
- ໃນກໍລະນີນີ້, ຈະເກີດຄວາມຜິດພາດຂຶ້ນ.

### 3. ຂະບວນການຄົ້ນຫາຕົວປ່ຽນທີ່ບໍ່ແມ່ນຕົວປ່ຽນພາຍໃນ

#### 3.5. ຂະບວນການຊອກຫາຕົວທີ່ບໍ່ແມ່ນຕົວປ່ຽນພາຍໃນ

- | ສົມມຸດວ່າ ພັງຊັນ g ຖືກກຳນົດໄວ້ໃນພັງຊັນ f ແລະ ພັງຊັນ h ຖືກກຳນົດໄວ້ໃນພັງຊັນ g.
- | ດ້ວຍ a ຖືກປ່ຽນແປງເນື່ອງຈາກວ່າ ຫຼັງຈາກພັງຊັນ h ຖືກດຳເນີນການແລ້ວ, ການອ້າງອີງ a ຂອງ g ແມ່ນ nonlocal.
- | a ບໍ່ມີການປ່ຽນແປງຫຼັງຈາກ g ພາຍໃນ f ຖືກປະຕິບັດ. ນີ້ແມ່ນຍອນວ່າມັນບໍ່ໄດ້ອ້າງເຖິງດ້າຂອງ f ເນື່ອງຈາກ nonlocal.
- | ນັ້ນແມ່ນ, nonlocal ຊອກຫາຕົວປ່ຽນຈາກ namespace ທີ່ໄກ້ທີ່ສຸດ.

```

1 def f():
2     a = 777
3     def g():
4         a = 100 ←
5         def h():
6             nonlocal a ←
7             a = 333
8             h()
9             print("[Level 2] a = {}".format(a))
10            g()
11            print("[Level 1] a = {}".format(a))
12
13 f()

```

Nonlocal variable find a variable from  
the nearest namespace.

```
[Level 2] a = 333
[Level 1] a = 777
```

## 4. ແນວຄວາມຄິດຂອງ Closure

### 4.1. ກ່ຽວກັບ Closure

- | ຮູບແບບພື້ນຖານຂອງຝັງຊັນ Closure ມີດັ່ງນີ້: ສ້າງຝັງຊັນຊ້ອນກັນ (Nested Function) ແລະ ສິ່ງຝັງຊັນດັ່ງກ່າວດ້ວຍຄ່າ.
- | ສາມາດເວົ້າໄດ້ວ່າຝັງຊັນຊ້ອນກັນ 'ຖືກປິດ'. ຖ້າຝັງຊັນຖືກປິດ, ຕົວປ່ຽນພາຍໃນຝັງຊັນຫຼັກ (ຝັງຊັນນອກ) ຈະບໍ່ຫາຍໄປຈາກທໍາມ່ວຍຄວາມຈຳ. ແລະ ຈະສາມາດເອັນໃຊ້ຝັງຊັນດັ່ງກ່າວໄດ້.
- | ສຶກສາແນວຄວາມຄິດຂອງ closure ດ້ວຍ code ຄຳສັ່ງດັ່ງລຸ່ມນີ້.
- | ຕົວປ່ຽນເອກະລາດແມ່ນຫຍັງ? ແມ່ນຕົວປ່ຽນທີ່ໃຊ້ສະເພາະໃນ block ເທົ່ານັ້ນ, ມັນບໍ່ແມ່ນຕົວປ່ຽນພາຍນອກ, ແລະ ມັນບໍ່ໄດ້ຖືກກຳນົດພາຍໃນບລັອກ. ມັນເປັນແນວຄວາມຄິດທີ່ສໍາພັນກັນ.
- | ໃນມູມມອງຂອງຝັງຊັນ, a ແມ່ນຕົວປ່ຽນເອກະລາດ. ຢ່າງໃດກໍຕາມ, ສໍາລັບຝັງຊັນທີ່ຂີ້ closure\_calc, a ເປັນພຽງແຕ່ຕົວປ່ຽນພາຍໃນຝັງຊັນເທົ່ານັ້ນ.

```

1 def close_calc():
2     a = 2
3     def mult(x):
4         return a * x
5     return mult
6
7 c = close_calc()
8 print(c(1), c(2), c(3))

```

2 4 6

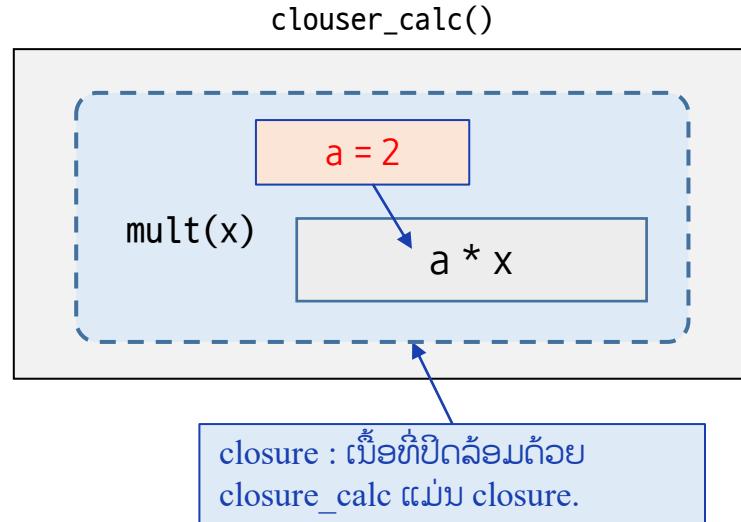
## 4. ແນວຄວາມຄິດຂອງ Closure

### 4.2. ການວິເຄາະ closure function

- | ໃນ code ລຸ່ມນີ້, ຄ່າທີ່ສື່ງຄືນຂອງຟັງຊັນ close\_calc ແມ່ນ ພັງຊັນ mult ຫາຍວ່າ ມັນສື່ງການອ້າງອີງຂອງຟັງຊັນພາຍໃນ.
- | ເນື່ອຈາກຄ່າທີ່ສື່ງຂອງຟັງຊັນ closure\_calc ແມ່ນຟັງຊັນ mult ແລະ ພັງຊັນຂອງ python ເປັນ first-class functions ມັນຈຶ່ງສາມາດເອັນຄ່າທັງໝົດໃນຟັງຊັນ mult ດ້ວຍ ຕົວປ່ຽນ c ໄດ້.
- | ຢ່າງໃດກໍຕາມ, ພັງຊັນ mult ໄດ້ຖືກປະມວນຜົນ ເຖິງແມ່ນວ່າຕົວປ່ຽນ a ທີ່ໃຊ້ໃນຕົວປ່ຽນ c ນັ້ນເປັນຕົວປ່ຽນຂອງຟັງຊັນ closure\_calc. ເຊິ່ງໝາຍຄວາມວ່າ ຕົວປ່ຽນ a ຂອງ ພັງຊັນ mult ບໍ່ໄດ້ຫາຍໄປຈາກໜ່ວຍຄວາມຈໍາຫຼັງຈາກການເຮັດວຽກຂອງຟັງຊັນ mult.

```
def closure_calc():
    a = 2
    def mult(x):
        return a * x
    return mult

c = closure_calc()
print(c(1), c(2), c(3))
```



## 4. ແນວດວາມຄິດຂອງ Closure

#### 4.3. ເປົ້າໝາຍຂອງ closure

- | ໃນການຂຽນໂປຣແກຣມແບບດັ່ງເຕີມ, ເຮົາຕ້ອງປະກາດຕົວປ່ຽນຮ່ວມນັ້ນໝາຍເຖິງຕົວປ່ຽນ global ເພື່ອໃຊ້ກັບໜ້າຍຟັງຊັ້ນ ໂດຍບໍ່ໄດ້ຄຳນິ່ງວ່າຟັງຊັ້ນໃດຟັງຊັ້ນໜຶ່ງຈະສັ່ນສຸດໜີ້ທີ່.
  - | ຢ່າງໃດກໍຕາມ, ການສ້າງຕົວປ່ຽນ global ທີ່ເປັນຕົວປ່ຽນຮ່ວມ ໂດຍບໍ່ມີການຄໍານິ່ງເຖິງການປິດຂອງຟັງຊັ້ນຈະເກີດຄວາມສ່ຽງໜ້າຍ.
  - | ການຂຽນໂປຣແກຣມ ຈະກາຍເປັນເລື່ອງຈ່າຍ ຖ້າໃຊ້ຕົວປ່ຽນ global ແຕ່ຖ້າໃຊ້ໜ້າຍເກີນໄປ ກໍຈະເຮັດໃຫ້ເກີດຜົນຂ້າງຄຽງ, ເຮັດໃຫ້ເກີດບັນຫາໃນການ debug.
  - | ເພື່ອສ້າງຟັງຊັ້ນ closure ໃຫ້ກໍານົດຕົວປ່ຽນສໍາລັບການປິດ (closure) ແລະ ພັງຊັ້ນ ແລ້ວປິດລ້ອມດ້ວຍຟັງຊັ້ນອື່ນ.
  - | ຖ້າມີຟັງຊັ້ນພາຍນອກປິດຟັງຊັ້ນ closure , ປະເພດການສຶ່ງຂອງຟັງຊັ້ນພາຍນອກກາຍເປັນ closure .
  - | ນີ້ສາມາດຫຼຸດຜ່ອນຄວາມຖືຂອງຕົວປ່ຽນ global ແລະ ເຊື່ອງການເຮັດພາຍໃນຟັງຊັ້ນຈາກພາຍນອກຟັງຊັ້ນ.

## 4. ແນວຄວາມຄິດຂອງ Closure

### 4.4. ການນຳໃຊ້ Closure

| ເປົ້າໝາຍຫຼັກຂອງການນຳໃຊ້ພັງຊັນ closure ມີສອງເປົ້າໝາຍດັ່ງລຸ່ມນີ້.

1. ຈຳກັດການນຳໃຊ້ຕົວປ່ຽນ global
2. ການເຊື່ອງຂໍ້ມູນ

- ▶ ໃນແຜ່ຂອງປະສິດທິພາບການເຮັດວຽກຂອງຫນ່ວຍຄວາມຈຳ, ໃຫ້ປິດຖ້າບໍ່ມີປະສິດທິພາບ. ຢ່າງໄດ້ກໍ່ຕາມ, ມັນອາດຈະມີປະສິດທິພາບຫຼາຍຂຶ້ນເພາະວ່າມັນຫຼຸດຕົວປ່ຽນ global
- ▶ ຖ້າຕ້ອງການເຊື່ອງຂໍ້ມູນ, ໃຫ້ປະກາດປິດເປັນຕົວປ່ຽນພາຍໃນຂອງພັງຊັນປິດລ້ອມ.

| ພັງຊັນ closure ໄດ້ແນະນຳໃຫ້ namespace ເອກະລາດສໍາລັບແຕ່ລະພັງຊັນ. ຈາກນັ້ນກໍສາມາດກຳນົດພັງຊັນຢ່າງເອກະລາດຄືກັບ instance object ໄດ້.

- ▶ ຂໍ້ມູນເພີ່ມຕື່ມກ່ຽວກັບ instances ແມ່ນຢູ່ໃນ Class chapters.

## 4. នេវកម្រោង Closure

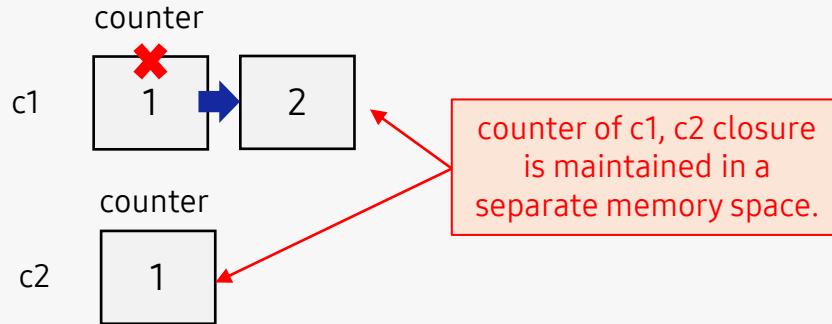
### 4.4. រាយការណ៍ Closure

| closure ແມ່ນແຍກអំពីកម្រោងទាំងផ្លូវ.

```

1 def makecounter():
2     count = 0
3     def counter():
4         nonlocal count
5         count += 1
6         return count
7     return counter
8 c1 = makecounter()
9 c2 = makecounter()
10 print('c1', c1())
11 print('c1', c1())
12 print('c2', c2())

```



c1 1  
c1 2  
c2 1

- ▶ តារាង code ខាងលើ ត្រូវបានដោះស្រាយ clousure ແຍកអំពីកម្រោងទាំងផ្លូវ.
- ▶ ពិគ្រោះ `count` សំឡែង `c1` និង `c2` មិនមែនអំពីកម្រោងទាំងនេះទេ.

## 4. ແນວຄວາມຄິດຂອງ Closure

### 4.5. ວິທີການສ້າງ closure

| ສ້າງ closure ດ້ວຍການຂຽນ nested-function ແລະ ການສົ່ງຄືນ.

```
1 def calc():
2     a = 3
3     b = 5
4     def mul_add(x):
5         return a * x + b
6     return mul_add
7
8 c = calc()
9 print(c(1), c(2), c(3), c(4), c(5))
```

8 11 14 17 20

#### ແຖວ 5, 6

- ຄິດໄລ່ໂດຍໃຊ້ຕົວປ່ຽນພາຍໃນ a, b ທີ່ຢູ່ນອກຟັງຊັ້ນ.
- ສົ່ງຄືນຟັງຊັ້ນ mul\_add.

## 5. ສ້າງ Closure ດ້ວຍພິຈາລະນາ Lambda

### 5.1. ຕົວຢ່າງການສ້າງ closure ດ້ວຍ lambda

- | ສ້າງ lambda expression ເຊັ່ນ: return lambda x: a\* x + b ແລະ return lambda expression ດ້ວຍຕົວຂອງມັນເອງ.
- | ດັ່ງນັ້ນ, ເຮົາສາມາດສ້າງ closure ດ້ວຍວິທີທີ່ຢ່າຍກວ່າໂດຍໃຊ້ lambda.
- | ປຶກກະຕິແລ້ວ closure ແມ່ນໃຊ້ຮ່ວມກັນກັບ lambda expression ແລະ ມັນຈ່າຍທີ່ຈະເຮັດໃຫ້ຫັງສອງສັບສົນກັນ.
- | lambda ເປັນພິຈາລະນາທີ່ບໍ່ໄດ້ກຳນົດຊື່, ສ່ວນ closure ຫມາຍເຖິງພິຈາລະນາທີ່ນຳມາໃຊ້ຊ້າຫຼັງຈາກການເຮັດວຽກຂອງມັນສັ່ນສຸດ

```
1 def clouser_calc():
2     a = 2
3     b = 3
4     return lambda x : a * x + b    # making closure with lambda
5
6 c = clouser_calc()
7 print(c(1), c(2), c(3), c(4), c(5)) # prints 5, 7, 9, 11, 13
```

5 7 9 11 13

## 5. ສ້າງ Closure ດ້ວຍພິຈາລະນາ Lambda

### 5.1. ຕົວຢ່າງການສ້າງ closure ດ້ວຍ lambda

| ປຽບທຽບ code ຄໍາສັ່ງ ກ່ອນ ແລະ ຫຼັງການໃຊ້ lambda

```

1 def calc():
2     a = 3
3     b = 5
4     def mul_add(x):
5         return a * x + b
6     return mul_add
7
8 c = calc()
9 print(c(1), c(2), c(3), c(4), c(5))

```

8 11 14 17 20

ກ່ອນໃຊ້ lambda



```

1 def clouser_calc():
2     a = 3
3     b = 5
4     return lambda x : a * x + b    # making closure with lambda
5
6 c = clouser_calc()
7 print(c(1), c(2), c(3), c(4), c(5)) # prints 5, 7, 9, 11, 13

```

8 11 14 17 20

ຫຼັງໃຊ້ lambda: Simpler expression

## 6. ປ່ຽນຕົວປ່ຽນພາຍໃນຂອງ Closure

### 6.1. ຕົວຢ່າງ code ຄໍາສັ່ງປ່ຽນຕົວປ່ຽນພາຍໃນຂອງ closure

- | ໃຊ້ nonlocal ເພື່ອປ່ຽນຕົວປ່ຽນພາຍໃນຂອງ closure
- | ອີງຕາມຜົນໄດ້ຮັບຂອງ  $a * x + b$  ຢູ່ໃນຕົວປ່ຽນພາຍໃນທີ່ຊື່ total ຂອງຝ່າງຊັ້ນ calc.

```
1 def calc():
2     a = 2
3     b = 3
4     total = 0 ← nonlocal ຄົ້ນຫາຕົວປ່ຽນຈາກ namespace ທີ່ໄກ້ທີ່ສຸດ.
5     def mult_add(x):           ໃນນີ້ມັນເປັນຕົວປ່ຽນ local ຂອງ closure.
6         nonlocal total
7         total = total + a * x + b
8         return total
9     return mult_add
10
11 c = calc()
12 print(c(1), c(2), c(3))
```

5 12 21

# Paper coding

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

**Q1.**

จงสร้าง nested-function ด้วยภาษาส้างฟังชันที่ชื่อ greetings และ ส้างฟังชันที่ชื่อ say\_hi ด้านบนเข้าไปในฟังชัน greetings. 乍กนั้นให้ เอ็นใช้ฟังชัน say\_hi พาไปฟังชัน greetings และ เอ็นใช้ฟังชัน greetings และ พิม ‘hello’ ออกมานะ. สำลับฟังชัน say\_hi ได้ สะແດງຕັ້ງລຸ່ມນີ້.

```
def say_hi():
    print('hello')
```

Condition for Execution

hello

Time

5 Minutes



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

**Q2.**

ສ້າງຝັງຊັນທີ່ຊື່ calc ໂດຍມີລາຍລະອຽດດັ່ງສະແດງລຸ່ມນີ້ ແລະ ກຳນົດຕົວປ່ຽນໃຫ້ກັບຝັງຊັນດັ່ງກ່າວ ຊື່ num. ຈາກນັ້ນໃຫ້ກຳນົດ num(3) ເພື່ອປະຕິບັດການໃຫ້ໄດ້ຜົນເທົ່າວັບ 14 ດັ່ງສະແດງລຸ່ມນີ້

```
def calc():
    a = 3
    b = 5
    def mul_add(x):
        return a * x + b
    return mul_add
```

Condition for Execution	14
Time	5 Minutes



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

**Q3.**

ຈາກຝັງຊັນ mul\_add ເຊິ່ງເປັນຝັງຊັນພາຍໃນຂອງຝັງຊັນ calc ໃນ Q2 ໃຫ້ນໍາໃຊ້ lambda expressions ເພື່ອພິມຜົນໄດ້ຮັບດັ່ງສະແດງລຸ່ມນີ້.

Condition for Execution	14
Time	5 Minutes



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

| Let's code

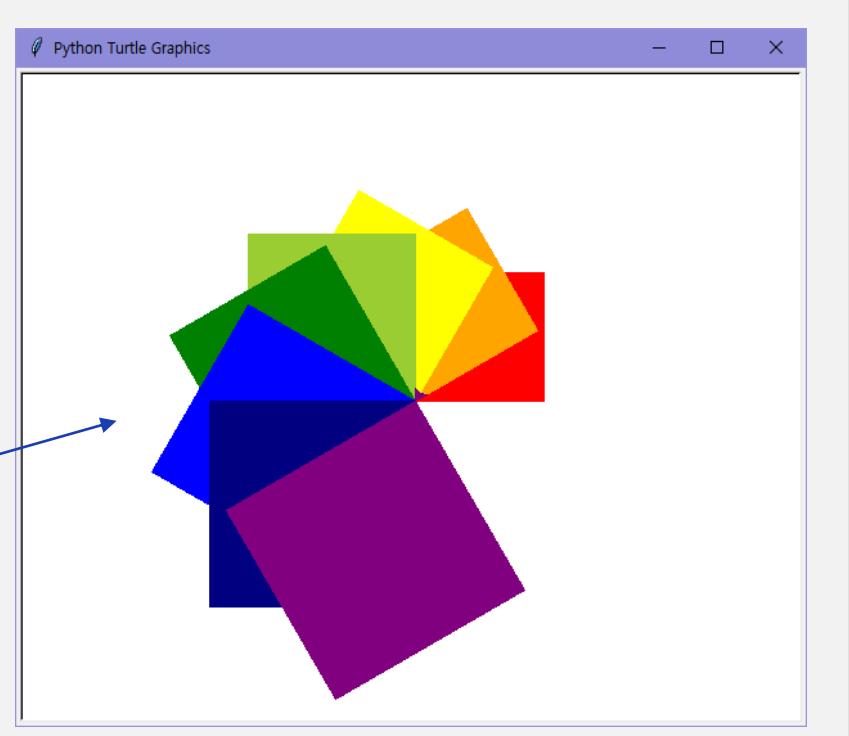
## 1. Turtle Graphics

### 1.1. Turtle graphics : ເຄື່ອງມີໃຊ້ແຕ່ມໃນ Python

| ໃນນີ້ ຈະອະທິບາຍກ່ຽວກັບ turtle graphics ແລະ ເຄື່ອງມີໃຊ້ແຕ່ມໃນ Python.

- ▶ ຮອງຮັບການແຕ່ມໃນ Python.
- ▶ Built-in ເປັນໄມດຸນພື້ນຖານຂອງ Python.
- ▶ ປະກອບດ້ວຍ methods ຕ່າງໆ.
- ▶ ມີ features ອຸດົມສົມບູນ.

ສາມາດເປັ່ນທີ່ສ່ວຍງາມ  
ຮູບພາບໃນຫນ້າຈຳຄອມພິວເຕີ

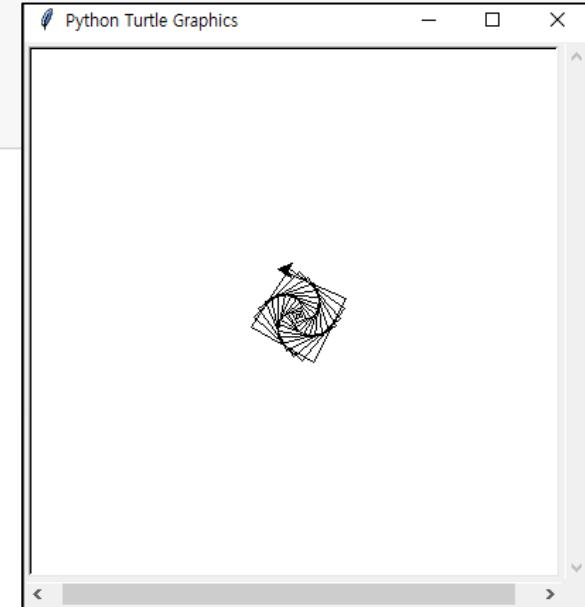


## 1. Turtle Graphics

### 1.2. งานแต้มรูป turtle

- | ถ้า run code คำสั่งลุ่มนี้ จะมีหน้าต่างປະກິດອອກມາໃຫ້ສາມາດแต้มรูบໄດ້.
- | จาก code คำสั่งลุ่มนี้ t ແມ່ນຊື່ຫຸ້ນຂອງ turtle module.

```
1 import turtle as t
2
3 t.setup(width = 400, height = 400)
4 for i in range(200):
5     t.forward(i)
6     t.left(93)
7 t.done()
```



# 1. Turtle Graphics

## 1.3. Turtle initialization และ modifying shapes

- | Window ໃໝ່ຈະປະກິດຂຶ້ນ ຖ້າມີການປ້ອນ code ດຳສັງລຸ່ມນີ້ໃນ Python chat Window.
- | t.setup function ຈະກຳນົດຂະໜາດ (size), ຫົວຂໍ (title) ແລະ ຄຸນສົມບັດອື່ນໆຂອງ chat window

```
1 import turtle as t
2
3 t.setup(width = 400, height = 400)
4 for i in range(200):
5     t.forward(i)
6     t.left(93)
7 t.done()
```

- ▶ t.forward(i) ຍ້າຍ cursor ໄປທາງຊ້າຍສໍາລັບ i pixels ໃນຄະນະທີ່ແຕ່ມເສັ້ນ.
- ▶ t.left(93) ໝູນ cursor ເປັນ 98 ອົງສາຊ້າຍ.

# 1. Turtle graphics

## 1.4. Turtle commands

| ມາຮູ້ຈັກກັບຄໍາສັ່ງຢາງຂອງ turtle graphics ເຊັ່ນ: forward(), left(), right () etc. Turtle graphics ມີຄຸນສົມບັດເພີ່ມເຕີມຢ່າງຫຼວງຫຼາຍ. ຕາຕະລາງ ລຸ່ມນີ້ໄດ້ສະແດງຄໍາສັ່ງຂອງ turtle.

command	task
begin_fill() ... end_fill()	begin_fill() ແລະ end_fill () ເປັນຄໍາສັ່ງທີ່ໃສ່ສືຂອງພື້ນຮູບ. ເຮົາສາມາດບັນທຶກພິກັດຂອງ turtle ຫຼື shape.
color(c)	ປ່ຽນສີຂອງ turtle. ໃນນີ້ສາມາດເລືອກສີໄດ້ຕາມໃຈເຊັ່ນ: ‘red’, ‘green’, ‘blue’, ‘black’, ‘gray’, ‘pink’ ພາຍໃນ value c.
shape(s)	ປ່ຽນ shape ຂອງ turtle. ດ້ວຍອ່ານວ່າ s ສາມາດເປັນ ‘arrow’, ‘turtle’, ‘circle’, ‘square’, ‘triangle’, ‘classic’ etc.
shapesize(s), shapesize(w, h)	ປ່ຽນແປງຂະໜາດຂອງ turtle.
pos(), position()	ສັ່ງຕຳແໜ່ງປະຈຸບັນຂອງ turtle ກັບ
xcor()	ສັ່ງຈຸດພິກັດ x ປະຈຸບັນຂອງ turtle ກັບ.
ycor()	ສັ່ງຈຸດພິກັດ y ປະຈຸບັນຂອງ turtle ກັບ.
heading()	ສັ່ງຫົວຂໍປະຈຸບັນກັບ.
distance(x, y)	ຄຳນວນໄລຍະຫ່າງລະຫວ່າງຕຳແໜ່ງປະຈຸບັນຂອງ turtle ແລະ ຕຳແໜ່ງທີ່ກຳນົດ.
penup(), pu(), up()	ຈັບເອົາບົກ (ປິດໃຊ້ວຽກການແຕ່ມ).

## 1. Turtle graphics

### 1.4. Turtle commands

#### | Initializing turtle และ modifying shapes

pendown() , pd(), down()	ເອົາບິກເພື່ອແຕ່ມ (ເປີດການແຕ່ມ).
pensize(w), width(w)	ບໍງານຄວາມອາມໝາຂອງບິກ
circle(r)	ແຕ່ມລົງມິນດ້ວຍຂະໜາດຂອງລັດສະໜີ r ທີ່ຕໍ່າແໜ່ງປະຈຸບັນ.
goto(x, y), setpos(x,y), setposition(x,y)	ວາງ cursor ໄປທີ່ຈຸດພິກັດ. ແຕ່ມເສັ້ນ ຖ້າ if penup(), ຖ້າບໍ່ pendown(),
stamp()	ສະແດງຂະໜາດ, ສີ ແລະ shape ຂອງ turtle ໃນຕໍ່າແໜ່ງປະຈຸບັນຂອງ turtle.
home()	ກຳນົດຕໍ່າແໜ່ງເລີ່ມຕົ້ນ ແລະ ກົງໄປທີ່ turtle.
textinput()	ສະແດງ chat window ສໍາລັບການປ້ອນຂໍ້ຄວາມ ແລະ window receives stings. ( <b>ຄໍາເຕອນ :ຕ້ອງໃຊ້ກັບ turtle.textinput()</b> )

- ▶ There are other various commands that enable the user to draw any kind of drawing.

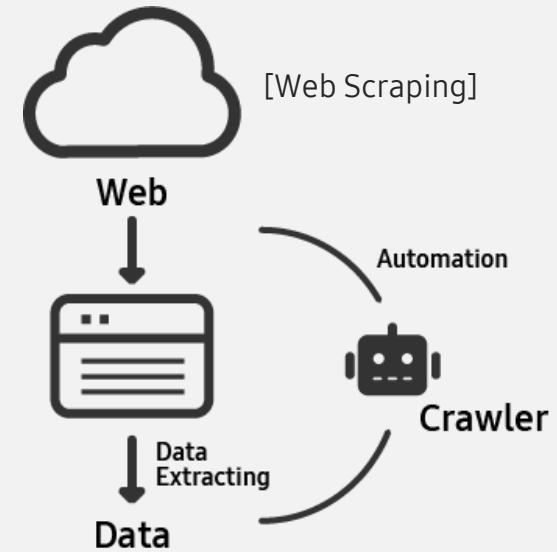
## 2. Regular Expressions และ Meta Characters

### 2.1. Regular expressions

- ▶ Regular expressions เป็นภาษาที่บันจุ patterns ทึກใช้โดย strings กับ certain rules.
- ▶ Regular expressions มีความสำคัญสำหรับการแยกอิเตาช์เวบ และ crawling.
- ▶ ระบุหัวข้อที่ต้องการ แล้วหาในเว็บไซต์.

Extracting web pages is called crawling.

<https://stackhoarder.com/2019/08/18/python%EB%B6%80%ED%84%B0-web-scraping-%EA%B9%8C%EC%A7%80-%EC%B5%9C%EB%8B%A8-%EC%8B%9C%EA%B0%84%EC%97%90-%EC%9D%B5%ED%98%80%EB%B3%B4%EC%9E%90/>



## 2. Regular Expressions และ Meta Characters

### 2.2. ການຄົ້ນຫາດ້ວຍ regular expressions

- | syntax ຂອງ regular expressions ແມ່ນຊັບຊ້ອນ. ໃນນີ້ຈະນໍາສະເໜີຝຶງຂັ້ນຄົ້ນຫາຂັ້ນພື້ນຖານເທົ່ານັ້ນ.
- | ນໍາໃຊ້ re module ໃນ regular expressions ຂອງ Python.
- | ສາມາດກວດສອບ ຖ້າ string ມີ specific string ດ້ວຍການນໍາໃຊ້ search function.
- | ກວດສອບ ຖ້າສອງ strings txt1 ແລະ txt2 ບັນຈຸ Life string.

```
1 import re
2
3 txt1 = "Life is too short, you need python."
4 txt2 = "The best moments of my life."
5 print(re.search('Life', txt1))    # checks if Life is in the sentence
```

```
<re.Match object; span=(0, 4), match='Life'>
```

```
1 print(re.search('Life', txt2))    # checks if Life is in the sentence
```

```
None
```

- ▶ ຜິນຂອງຄໍາສັ່ງ print ສໍາລັບ txt1 ແມ່ນ ‘match’ ເພະວ່າ txt1 ບັນຈຸ Life ທີ່ຂຶ້ນຕົ້ນດ້ວຍໂຕ L ໃຫຍ່ງ.
  - ຖ້າມີ matching string, ຂໍ້ຄວາມ ແຊ້ນວ່າ <re.Match object .. > ຈະຖືກພິມອອກມາ.
- ▶ ຜິນຂອງຄໍາສັ່ງ print ສໍາລັບ txt2 ແມ່ນ None ເພະວ່າ txt2 ບໍ່ບັນຈຸ Life ທີ່ຂຶ້ນຕົ້ນດ້ວຍໂຕ L ໃຫຍ່ງ.

## 2. Regular Expressions และ Meta Characters

### 2.2. ການຄົ້ນຫາດ້ວຍ regular expressions

- | ການເຊື່ອ group method ຂອງ match variable. ເນື່ອງຈາກ Life ສະແດງພຽງຄັ້ງດຽວ, group ຈຶ່ງພິມ Life.
- | group method ຄົ້ນຫາສ່ວນທີ່ກິງກັບ regular expression. ໃນນີ້ Life ແມ່ນຖືກພິມອອກມາ.

```
1 | match = re.search('Life', txt1)
2 | match.group()
```

'Life'

- ▶ ການຕິຄວາມຜົນໄດ້ຮັບນີ້, ກໍານົດຄ່າຜົນຮັບຂອງ re.search ທີ່ກິງກັບຕົວປ່ຽນ ແລະ ເຊື່ອ start, end, span method ໃຫ້ກັບຕົວປ່ຽນ.

```
1 | match.start()
```

0

```
1 | match.end()
```

4

```
1 | match.span()
```

(0, 4)

## 2. Regular Expressions และ Meta Characters

### 2.2. ການຄົ້ນຫາດ້ວຍ regular expressions

- | group method จะສິ່ງ string ທີ່ກົງກັນດ້ວຍ regular expression. ມັນສະແດງຕຳແໜ່ງເລີ່ມຕົ້ນ ແລະ ຕຳແໜ່ງສິ່ນສຸດຂອງ ການກົງກັນຂອງ string ດ້ວຍ ການນຳໃຊ້ start, end method.
- | ພ້ອມທັງ, span method จะສິ່ງ tuple ອອກມາ (start, end) ແມ່ນຕຳແໜ່ງຂອງ match

```
1 match.start()
```

```
0
```

```
1 match.end()
```

```
4
```

```
1 match.span()
```

```
(0, 4)
```

```
1 txt1[0:4]
```

```
'Life'
```

## 2. Regular Expressions และ Meta Characters

### 2.2. งานค้นหาด้วย regular expressions

- | ในกำลังนี้ text string ที่จะหาลุ่มมี, compile function ของ re module ค้นหา regular expression patterns และ compiles มันไว้ใน regular expression objects.
- | จากนั้น, regular expression object และ regex ที่ก็สามารถใช้ object ค้นหา text ด้วย given pattern.
- | \d{3} ค้นหา patterns ของตัวเลข 3 ตัว. \d{4} ค้นหา patterns ของตัวเลข 4 ตัว. ในอิงเลับ patterns ใน group 1 และ group 2.
  - ▶ That is, \d{3} finds three-digit integer patterns like 010.

```
1 import re
2
3 text = 'please call 010-2345-1234'
4 regex = re.compile('(\d{3})-(\d{4}-\d{4})')
5 match_obj = regex.search(text)
6 print(match_obj.group())
```

010-2345-1234

group(1)

group(1)

```
1 print(match_obj.group(1))
```

010

```
1 print(match_obj.group(2))
```

2345-1234

## 2. Regular Expressions และ Meta Characters

### 2.2. ການຄົ້ນຫາດ້ວຍ regular expressions

- ຕົວຢ່າງຜົນຂອງການຄົ້ນຫາໂດຍໃຊ້ search function ຂອງ regular expression.
- ມັນຄົ້ນຫາ string, ແຕ່ບໍ່ໄດ້ຄົ້ນຫາ life ດ້ວຍຕົວອກສອນນ້ອຍ 1 in txt2. ທີດລອງປະຕິບັດດັ່ງລຸ່ມນີ້.

```
1 txt2 = "The best moments of my life."  
2 print(re.search('Life', txt2))    # checks if Life is in the sentence
```

None

```
1 print(re.search('Life|life', txt2))    # checks if Life is in the sentence  
<re.Match object; span=(23, 27), match='life'>
```

- ກໍລະນີນີ້, ເນື່ອງຈາກ ‘life’ ແມ່ນ ບັນຈຸຢູ່ໃນ txt2, ມັນສະແດງຕໍາແໜ່ງຂອງ ‘life’ ຜ່ານຜົນຮັບ span (23, 27).
- ສັນຍາລັກ | ບໍ່ໝາຍຄວາມວ່າຄົ້ນຫາ ‘Life|life’. ແຕ່ມັນມີຄວາມໝາຍວ່າຄົ້ນຫາຂັ້ນຄວາມ Life ຫຼື life .

## 2. Regular Expressions ແລະ Meta Characters

## 2.3. Meta characters

- | เมื่อจาก Life ที่ life แม่นเป็นขั้นความดูวักัน ต่างกันแต่ติวอักษรอน L และ 1 สะนั้น มันจึงส่องผินรักของมาถีกัน.

```
1 print(re.search('[lL]ife', txt2)) # checks if Life or life is in the sentence
```

```
<re.Match object; span=(23, 27), match='life'>
```

- ຄືກັບຕົວຢ່າງທີ່ຜ່ານມາ, [Ll]ife ມີຄວາມໝາຍ Life ຫຼື life. ເຊື້ອງໝາຍໃຈ [] ກໍານີດຫວ່າງຂອງ characters.

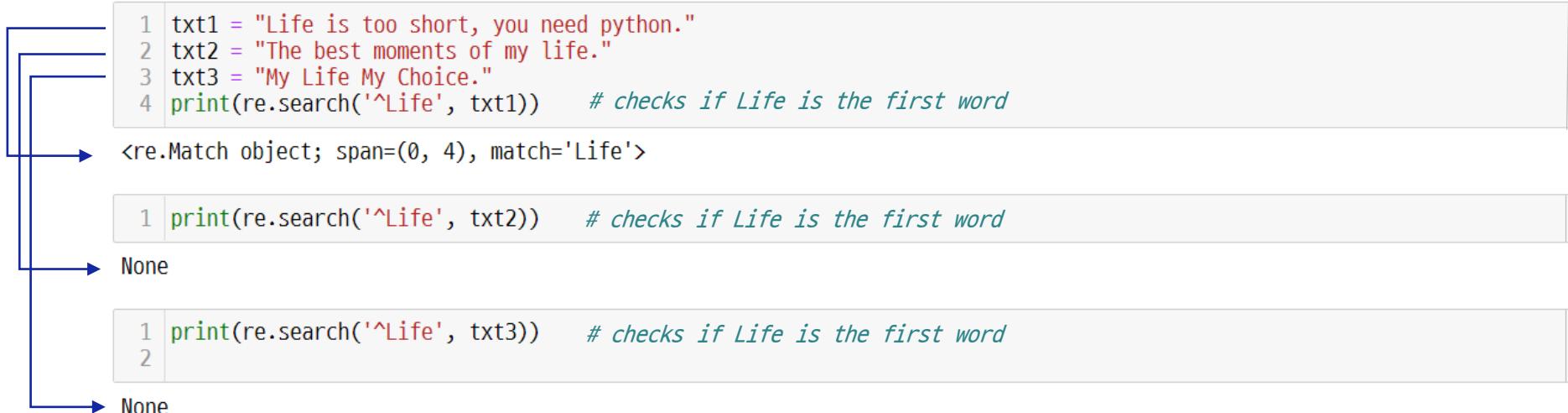
ເຊັ່ນ [0-9] ມີຄວາມໝາຍວ່າ ທັງໝົດ number characters ແມ່ນຈາກ 0 ເຖິງ 9.

- | ด้วยนั้น, เวิ่งสามารถถือเป็น characters โดยมีให้ characters ด้วย อักษรสองพิเศษเด่น [], -, |. ดังนั้น อักษรสองพิเศษนี้เป็นให้เพื่อเป็น meta characters.

## 2. Regular Expressions และ Meta Characters

### 2.3. Meta characters

| เคื่องหมาย ^ ແມ່ນມີຄວາມໝາຍວ່າເລີ່ມຕົ້ນຂອງ string ໃນregular expressions.



- ▶ ຜົນໄດ້ຮັບຈາກຂ້າງເທິງ, ຖ້າຄື້ນຫາ txt3 ໂດຍໃຊ້ ^Life ຜົນໄດ້ຮັບເປັນ None. ເນື່ອງຈາກ meta character ^ ກິງກັບປະໂຫຍກທີ່ມີ Life ຄື ດຳສັບທຳອິດ.

## 2. Regular Expressions และ Meta Characters

### 2.3. Meta characters

| ນໍາໃຊ້ regular expressions ດັ່ງສະແດງລຸ່ມນີ້.

```
1 txt1 = 'Who are you to judge the life I live'  
2 txt2 = 'The best moments of my life'  
3 print(re.search('life$', txt1)) # checks if Life or life is contained
```

None

```
1 print(re.search('life$', txt2)) # checks if Life or life is contained
```

<re.Match object; span=(23, 27), match='life'>

▶ ສາມາດຄົ້ນຫາຂໍ້ຄວາມ life ທີ່ຢູ່ທ້າຍປະໂຫຍກ ໂດຍໃຊ້ ອັກສອນພິເສດ \$ .

## 2. Regular Expressions และ Meta Characters

### 2.4. Essential meta characters

| มาเข่ยฟังขั้น regular expressions. meta characters สำลับ regular expressions ดังจะเดาๆ ลุ่มมี.

Regular expression	Function	Description
^	start	กິງກັບຂໍ້ຄວາມເລີ່ມຕົ້ນຂອງ string
\$	end	ກິງກັບຂໍ້ຄວາມສຸດທ້າຍຂອງ string
.	character	ກິງກັບໜຶ່ງຕົວອັກສອນ.
\d	number	ກິງກັບໜຶ່ງຕົວເລກ
\w	character or number	ກິງກັບໜຶ່ງຕົວອັກສອນ ຫຼື ຕົວເລກ.
\s	whitespace characters	ກິງກັບຊ່ອງຫວ່າງ, ຫຶ່ງ tab, a carriage return and a line feed
\S	except whitespace characters	ຫັງໝົດ characters ຍົກເວັ້ນ whitespace characters
*	repetition	0 ຫຼື ຫຼາຍ repetitions
+	repetition	1 ຫຼື ຫຼາຍ repetitions
[abc]	character range	[abc] ສະແດງ a ຫຼື b ຫຼື c
[^abc]	character range	[^abc] character ໄດກະໄດ້ ແຕ່ບໍ່ແມ່ນ a ຫຼື b ຫຼື c.

\* ສິ່ງທີ່ສໍາຄັນສໍາລັບ meta characters ແມ່ນ dot (.) ແລະ asterisk (\*). dot ມີຄວາມໝາຍວ່າ string ໄດກະໄດ້ ແລະ asterisk ມີຄວາມໝາຍວ່າ ເຮັດຊ້າຈໍານວນເຫົ່າໄດ້ຖືກກຳໄດ້.

## 2. Regular Expressions และ Meta Characters

### 2.4. Essential meta characters

- | dot meta character.
- | នីមួយៗ meta character មិត្តភាពមាយវា ស្ថុម character.
- | ពីរក្នុង strings ABA, ABBA, ABBBA, ម៉ែនកើតឡើងក្នុងក្រោមនេះ A..A ឬមែន ABBA, ដើម្បីមែនខ្លួចបែប Swedish ABBA.

```
1 re.search('A..A', 'ABA')      # does not fit the condition
```

```
1 re.search('A..A', 'ABBA')      # fits the condition
```

```
<re.Match object; span=(0, 4), match='ABBA'>
```

```
1 re.search('A..A', 'ABBBA')      # does not fit the condition
```

## 2. Regular Expressions ແລະ Meta Characters

### 2.4. Essential meta characters

- | asterisk (x) ດັ່ງນຳໃຊ້ຫຼາຍໃນ meta character.
- | ມັນກິຈກັບ strings ນັ້ນ repeat 0 times ຫຼື ສຸມ pattern ຂອງ string ກ່ອນ asterisk.

```
1 re.search('AB*', 'A')    # fits the condition
```

```
<re.Match object; span=(0, 1), match='A'>
```

```
1 re.search('AB*', 'AA')    # fits the condition
```

```
<re.Match object; span=(0, 1), match='A'>
```

```
1 re.search('AB*', 'J-HOP')  # does not fit the condition
```

```
1 re.search('AB*', 'X-MAN')  # fits the condition
```

```
<re.Match object; span=(3, 4), match='A'>
```

```
1 re.search('AB*', 'CABBA')    # fits the condition
```

```
<re.Match object; span=(1, 4), match='ABB'>
```

```
1 re.search('AB*', 'CABBBBBA')    # fits the condition
```

```
<re.Match object; span=(1, 7), match='BBBBBB'>
```

## 2. Regular Expressions และ Meta Characters

### 2.4. Essential meta characters

| Character ลุ่มນี้ก็จะกับ patterns แต่ที่ repeat นี่ repeat once สูง character ก่อน character.

```
1 re.search('AB?', 'A')    # fits the condition  
<re.Match object; span=(0, 1), match='A'>
```

```
1 re.search('AB?', 'AA')   # fits the condition  
<re.Match object; span=(0, 1), match='A'>
```

```
1 re.search('AB?', 'J-HOP') # does not fit the condition
```

```
1 re.search('AB?', 'X-MAN') # fits the condition  
<re.Match object; span=(3, 4), match='A'>
```

```
1 re.search('AB?', 'CABBA')     # fits the condition  
<re.Match object; span=(1, 3), match='AB'>
```

```
1 re.search('AB?', 'CABBBBBA')  # fits the condition  
<re.Match object; span=(1, 3), match='AB'>
```

| ผิดได้รับคือกับ \*, แต่ต่างกันหลายที่ string ก็จะกัน แม่น AB สำลับ CABBBBBA และ CABBA.

## 2. Regular Expressions ແລະ Meta Characters

### 2.4. Essential meta characters

- | meta character + ກົງກັບ patterns ນັ້ນແມ່ນ repeat ຫົ່ງ ຫຼື ຫຼາຍ random pattern ໃນຕໍ່ຫຼຳ +.
- | AB+ ບໍ່ກົງກັບ A, ແຕ່ ກົງກັບ strings ຂອງ patterns ເຊັ່ນ: AB, ABB, ABBB, CABBA.

```
1 re.search('AB+', 'A')      # does not fit the condition
1 re.search('AB+', 'AA')     # does not fit the condition
1 re.search('AB+', 'J-HOP')   # does not fit the condition
1 re.search('AB+', 'X-MAN')   # does not fit the condition
1 re.search('AB+', 'CABBA')    # 'ABB' string fits the condition
<re.Match object; span=(1, 4), match='ABB'>
1 re.search('AB+', 'CABBBBB')  # 'ABBBBB' fits the condition
<re.Match object; span=(1, 7), match='ABBBBB'>
```

## 2. Regular Expressions ແລະ Meta Characters

### 2.4. Essential meta characters

- | ກ່ຽວກັບ.findall ເຊິ່ງແມ່ນ search command.
- | Findall ຂອງ regular expressions ແຍກ strings ຫັງໝົດທີ່ເພີ້າຈະກັບ regular expression.

```
1 txt3 = 'My life my life my life in the sunshine'  
2 re.findall('[Mm]y', txt3)  
['My', 'my', 'my']
```

# | Pair programming



# Pair Programming Practice

| ແນວທາງ, ກິນໄກ ແລະ ແຜນສຸກເສີນ

ການຈັບຄຸ້ຂຽນໂປຣແກຣມ ເປັນການຈັບຄຸ້ຂອງນັກຮຽນເພື່ອເຮັດວຽກມອບໜາຍ, ນັກຮຽນຄວນມີແຜນ ແລະ ສາມາດປ່ຽນແທນກັນໄດ້ ໃນກໍລະນີມີຜູ້ໃດໜຶ່ງບໍ່ສາມາດເຂົ້າຮ່ວມເຮັດວຽກມອບໜາຍໄດ້ບໍ່ວ່າໃນກໍລະນີໃດກໍຕາມ ເຊິ່ງບັນຫາເລົ່ານີ້ຕ້ອງເຮັດໃຫ້ຈະແຈ້ງ ແລະ ກຳບໍ່ແມ່ນຄວາມຜິດຂອງນັກຮຽນທີ່ຈັບຄຸ້ບໍ່ດີ.

| จับคู่ที่ถ่ายถือวัน, บ่จ้าเป็นเหตุทางวัฒน, ความสามาดเป็นคุ้ร่วมงาน

| ກະຕຸນນັກງານໂດຍການໃຫ້ສື່ງຈູ່ໃຈພິເສດ

ຂໍສະເໜີແຮງງົງໃຈທີ່ເຮັດໃຫ້ນັກຮຽນຈັບຄຸ, ໂດຍສະເພາະນັກຮຽນທີ່ມີຄວາມສາມາດສູງ. ບາງຄຸສອນໄດ້ພິບວ່າ ການຈັບຄຸເຮັດວຽກມອບໝາຍ ແມ່ນມີປະໂຫຍດ ສໍາລັບໜຶ່ງ ຫຼື ສອງວຽກມອບເທົ່ານັ້ນ



# Pair Programming Practice

| ចំណាំការងារប៉ែន្តៅទិន្នន័យនៃក្រសួង

ສັງຫ້ທາຍສໍາລັບຄຸມເມ່ນພື້ອຊອກຫາວິທີທີ່ຈະປະເມີນຜົນການຮຽນຂອງນັກຮຽນ, ຄຸຮູ້ບໍ່ວ່າ ນັກຮຽນ ດັດຕັ້ງໃຈຮຽນ ຫຼື ບໍ່ຕັ້ງໃຈຮຽນ. ຜູ້ສ່ວວຊານໄດ້ແນະນຳໃຫ້ທີບທວນການອອກແບບຫຼັກສຸດການຮຽນ ແລະ ຮູບແບບການປະເມີນ ພ້ອມທັງປີກສາຫາລືຢ່າງຈົງຈັງກັບນັກຮຽນ ກ່ຽວກັບພິດຕິກຳທີ່ຈະບໍ່ຕັ້ງໃຈຮຽນ ນອກຈາກນີ້ຢັ້ງໄດ້ແນະນຳມອບວຽກມອບໝາຍໃຫ້ນັກຮຽນ ພ້ອມທັງອະທິບາຍໃຫ້ເຂົ້າເຈົ້າຢ່າງຈະແຈ້ງ

| ສະພາບແວດລ້ອມຂອງການຮຽນຮູ້ຮ່ວມກັນ

ສະພາບແວດລ້ອມການຮຽນຮູ້ຮ່ວມກັນເກີດຂຶ້ນໄດ້ທຸກເວລາທີ່ຜູ້ສອນຮຽກຮ້ອງໃຫ້ນັກຮຽນເຮັດວຽກຮ່ວມກັນໃນກິດຈະກຳການຮຽນຮູ້ ເຊິ່ງອາດຈະເປັນກິດຈະກຳທີ່ເປັນທາງການ ແລະ ບໍ່ເປັນທາງການ ແລະ ອາດຈະບໍ່ລວມເຖິງການປະເມີນຜົນການຮຽນໂດຍກິງ. ເຊັ່ນຕົວຢ່າງ ໃຫ້ນັກຮຽນຈັບຄຸ້ງກັນເພື່ອເຮັດວຽກອບໝາຍ ໂດຍນັກຮຽນຈະຕ້ອງທີ່ບໍ່ທວນກ່ຽວກັບການສອນຂອງອາຈານທີ່ຜ່ານມາ ແລະ ລະດົມແນວຄົດພາຍໃນກຸ່ມ ພ້ອມທັງມືການແບ່ງວຽກໃຫ້ແຕ່ລະຄົນຮັບຜິດຊອບ ຈາກນັ້ນກໍໃຫ້ມີການແລກປ່ຽນຄວາມຄິດເຫັນເຊິ່ງກັນ ແລະ ກັນ ເພື່ອເຮັດວຽກມອບໝາຍໃຫ້ສໍາເລັດຕາມເປົ້າໝາຍທີ່ວ່າງໄວ້.

## Q1.

ແຍກຂໍ້ມູນຈາກ list ທີ່ຊື່ lst ຊຶ່ງມີຄ່າແຕ່ 1 ~ 100. list ທີ່ຊື່ result ຈະມີອີງປະກອບຂອງ list ທີ່ຊື່ lst ຫານດ້ວຍ 5 ຫຼື 7. ໃຫ້ປະກາດຟັງຊັ້ນ func1(a) ແລະ nest ພົງຊັ້ນທີ່ຊື່ func2 ແລະ func3. ຈາກນັ້ນ ໃຫ້ເອີ້ນສອງຟັງຊັ້ນ ຢູ່ໃນພົງຊັ້ນ func1 ແລະ ພິມຕົວເລກທີ່ຫານຂາດໃຫ້ 5 ຫຼື 7 ພ້ອມທັງຈັດຄ່າດັ່ງກ່າວໂດຍໃຊ້ພົງຊັ້ນ sorted () .

### Print example

```
def func2():
    result1 = []
    for i in a:
        if i % 5 == 0:
            result1.append(i)
    return result1

def func3():
    result2 = []
    for i in a:
        if i % 7 == 0:
            result2.append(i)
    return result2
```

```
lst = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31,
32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 6
2, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92,
93, 94, 95, 96, 97, 98, 99, 100]
result = [5, 7, 10, 14, 15, 20, 21, 25, 28, 30, 35, 35, 40, 42, 45, 49, 50, 55, 56, 60, 63, 65, 70, 70, 75, 77, 80, 84, 8
5, 90, 91, 95, 98, 100]
```