

Unit 19.

# Lambda

## ឧបសម្ព័ន្យការងាររុណវី

- ✓ ផ្តល់ការពន្លានមុនពីការបង្កើតការងារដោយប្រើប្រាស់ការអនុវត្តន៍ការងារដែលបានរាយការណ៍ឡើង។
  - ✓ ផ្តល់ការពន្លានមុនពីការបង្កើតការងារដោយប្រើប្រាស់ការអនុវត្តន៍ការងារដែលបានរាយការណ៍ឡើង។
  - ✓ ផ្តល់ការពន្លានមុនពីការបង្កើតការងារដោយប្រើប្រាស់ការអនុវត្តន៍ការងារដែលបានរាយការណ៍ឡើង។
  - ✓ ផ្តល់ការពន្លានមុនពីការបង្កើតការងារដោយប្រើប្រាស់ការអនុវត្តន៍ការងារដែលបានរាយការណ៍ឡើង។

### ພາບລວມຂອງປິດຮຽນ

- ✓ ກຽນຮູ້ສິ່ງຕ່າງໆຂອງຟັງຊັນ lambda ພ້ອມທັງ ການກຳນົດ ແລະ ການເອີ້ນໃຊ້ ຜັງຊັນດັ່ງກ່າວ.
- ✓ ກຽນຮູ້ການແຍກອີງປະກອບທີ່ກິຈກັບຕູ່ອນໄຂບາງຢ່າງອອກຈາກ list ໂດຍໃຊ້ຟັງຊັນ filter.
- ✓ ກຽນຮູ້ ວິທີແກ້ໄຂຄ່າຂອງ unit ໃນ list ໂດຍນຳໃຊ້ຟັງຊັນ map.
- ✓ ກຽນຮູ້ວິທີຄິດໄລ່ຜົນລວມສະສົມຈາກຂໍ້ມູນຫຼາຍລາຍການ ໂດຍນຳໃຊ້ຟັງຊັນ reduce.

### ສິ່ງຈໍາເປັນຕ້ອງຮູ້ຈາກ Units ຜ່ານມາ

- ✓ ເຂົ້າໃຈກ່ຽວກັບຟັງຊັນຕ່າງໆ ແລະ ສາມາດອະທິບາຍຟັງຊັນ built-in functions ພ້ອມທັງຟັງຊັນທີ່ຜູ້ໃຊ້ກຳນົດເອງ
- ✓ ນຳໃຊ້ parameter ເພື່ອຜ່ານຄ່າ ໄປຍັງຟັງຊັນ
- ✓ ສ້າງ list ແລະ ນຳໃຊ້ list-related methods

## Keywords

`filter`

`map`

`reduce`

`lambda`

# Mission

## 1. บัมทາโลกແຫ່ງຄວາມເປັນຈີງ

### **1.1. Digital content ແລະ copyright issues**



- ▶ ในสัดຕะวัดที่ 21 แม่นยุกадิจิตตอม และ ยุกอัดทะนะทำ.
  - ▶ เถียงป่าฯ ได้กำหนด, ในขณะที่ส้างเมืองหาดิจิตตอมต้องใช้เวลา และ ความพยายาม หลาย, แต่ก็มีความสุรุ่งที่จะมีความเข้มข้น แล้ว งานเปียกผู้.
  - ▶ ถั่งนั้น งานคุ้มครองอับสินทางปั้นยา เด่น ลิขะสิดเจ่งเป็นประเด็นสำคัญในขั้นตอนเปียกผู้ เมืองหาดิจิตตอม.
  - ▶ โดยที่ว่าไป ลิขะสิดของปีมจะถึงปี 50 บี 70 ปี คาดจะมีความแตกต่างกันบ่หลายในบาง ประเทศ.

## 1. បំណងជាលក្ខណៈទិន្នន័យ

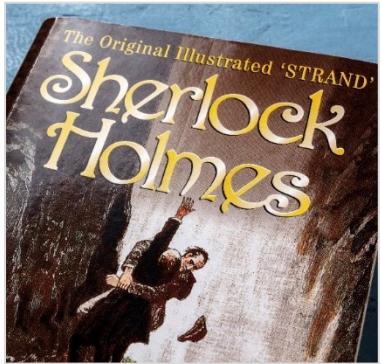
### 1.2. នគរបាយ Sherlock Holmes



- ▶ នគរបាយ Sherlock Holmes ខ្សោយ Arthur Conan Doyle ແມ່ນມີສິ່ງທີ່ໄດ້  
▶ នគរបាយນີ້ປະກອບມີ A Study in Scarlet (1887) – The Sign of the Four (1890) – The Adventures of Sherlock Holmes (1892) – The Memoirs of Sherlock Holmes (1894) – The Hound of the Baskervilles (1901) – The Return of Sherlock Holmes (1905) – The Valley of Fear (1915) – His Last Bow (1917) – The Case-Book of Sherlock Holmes (1927)  
▶ ຖ້າໄດ້ຮັບວັນທີ 1887.11.20 ແລະ ແຍກວິເຄາະວັນທີດັ່ງກ່າວ. ການແຍກວິເຄາະເປັນການວິເຄາະລາຍການ ຫຼື ໄວ  
ປະກອນຂໍ້ມູນ.  
▶ ການខ្សោយໂປຣແກຣມເພື່ອກວດສອບ ໂດຍອາໄສການແຍກວິເຄາະວ່າ ປີທີ່ບ້ອນເຂົ້າໄປເປັນປີທີ່ຕີພິມດ້ວຍ Conan Doyle.

## 1. បំណងជាន់ការណែនាំរបស់ខ្លួន

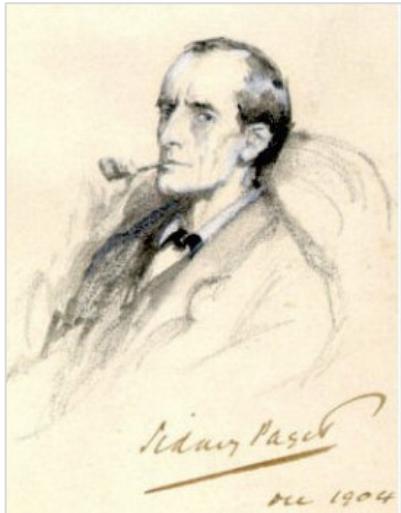
### 1.2. នគរណីយាយ Sherlock Holmes



- ▶ ប៉ែននគរណីយាយតិចសរុប ខ្សោយឡាយ Conan Doyle តើមិ Sherlock Holmes ប៉ែនពិវឌ្ឍន៍រៀង.
- ▶ នគរណីយាយនេះ ត្រូវបានរៀងដោយរៀងរាល់របស់ខ្លួន ដើម្បីបង្ហាញលើការងាររបស់ខ្លួន និងបង្ហាញលើការងាររបស់ខ្លួន។ នគរណីយាយនេះ ត្រូវបានរៀងដោយរៀងរាល់របស់ខ្លួន ដើម្បីបង្ហាញលើការងាររបស់ខ្លួន។
- ▶ ពិវឌ្ឍន៍រៀង Sherlock Holmes ប៉ែនពិវឌ្ឍន៍រៀងទាំងអស់ មិនត្រូវបានរៀងដោយរៀងរាល់របស់ខ្លួន ដើម្បីបង្ហាញលើការងាររបស់ខ្លួន។

## 1. บันทາໄລກແຫ່ງຄວາມເປັນຈິງ

### 1.2. ນະວະນິຍາຍ Sherlock Holmes



- ▶ ຫາຂໍ້ມູນເພີ່ມຕົມກ່ຽວກັບ Sherlock Holmes ໄດ້ທີ່ເວບໄຊ [https://en.wikipedia.org/wiki/Sherlock\\_Holmes](https://en.wikipedia.org/wiki/Sherlock_Holmes).

## 2. Mission

### 2.1. ແຍກວັນທີເພື່ອເພື່ອປຽບທຽບກັບປີທີ່ຕີພິມຂອງນະວະນິຍາຍ Sherlock Holmes



- ▶ ຮັບ ປີ, ເດືອນ ແລະ ວັນທີ ຈາກຜູ້ໃຊ້ ແລະ ກວດສອບວ່າ ປີນີ້ ເປັນປີທີ່ຕີພິມຂອງນະວະນິຍາຍ Sherlock Holmes ຫຼື ບໍ່.
- ▶ ປ້ອນຂໍ້ມູນຜູ້ໃຊ້ ຖ້າມັນຢູ່ໃນຮູບແບບ 1887.11.20
- ▶ ສ້າງຟັງຊັ້ນເພື່ອພິມຄໍາວ່າ “This is not a publication year of the Sherlock Holmes series” ຖ້າປີທີ່ຜູ້ໃຊ້ ປ້ອນບໍ່ແມ່ນປີທີ່ຕີພິມຂອງນະວະນິຍາຍ Sherlock Holmes.
- ▶ ສ້າງຟັງຊັ້ນ lambda ທີ່ຊື່ `is_sherlock_book_y` ເພື່ອກວດສອບວ່າ ປີທີ່ປ້ອນເປັນປີທີ່ຕີພິມຫຼືບໍ່.
- ▶ ນອກຈາກນີ້ ເຮົາຍັງກວດສອບວ່າ ເດືອນ ແລະ ວັນທີ ທີ່ຜູ້ໃຊ້ໄດ້ປ້ອນ ຖືກຕ້ອງຫຼືບໍ່.
- ▶ ຕົວຢ່າງ, 1915.5.3 ແມ່ນປ້ອນຖືກຕ້ອງ, ແຕ່ຖ້າປ້ອນ 1915.5.33 ແມ່ນບໍ່ຖືກຕ້ອງ. ຖ້າບໍ່ຖືກຕ້ອງ ໂປຣແກຣມ ຕ້ອງພິມຄໍາວ່າ “The date is incorrect.” ອອກມາຫາໜ້າຈໍາເພື່ອໃຫ້ຜູ້ໃຊ້ຮັບຮູ້

## 2. Mission

### 2.2. ສ້າງ Sherlock Holmes

```
1 def valid_date(date):
2     days = [31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
3     sherlock_series = [1887, 1890, 1892, 1894, 1901, 1905, 1915, 1917, 1927]
4     splitted = date.split('.')
5     year = int(splitted[0])
6     month = int(splitted[1])
7     day = int(splitted[2])
8     is_sherlock_book_yr = lambda x: x in sherlock_series
9
10    if not is_sherlock_book_yr(year):
11        print('This is not a publication year of the Sherlock Holmes series.')
12        return False
13    if month < 1 or month > 12 :
14        print('The month is incorrect.')
15        return False
16    if day < 1 or day > days[month-1]:
17        print('The date is incorrect.')
18        return False
19
20    return True
21
22 date = input('Distinguish year, month and date by .(e.g. 1917.6.10) : ')
23 if valid_date(date) == True:
24     print('This is the publication date of a Sherlock Holmes series!')
25 else:
26     print('This is not the publication date of a Sherlock Holmes series or the input has an error.')
```

ແຍກ ປີ, ເດືອນ ແລະ ວັນ ດັ່ງນີ້ (e.g. 1917.6.10) : 1927.9.10  
ນີ້ແມ່ນວັນທີຕີພິມຂອງນະວະນິຍາຍ Sherlock Holmes

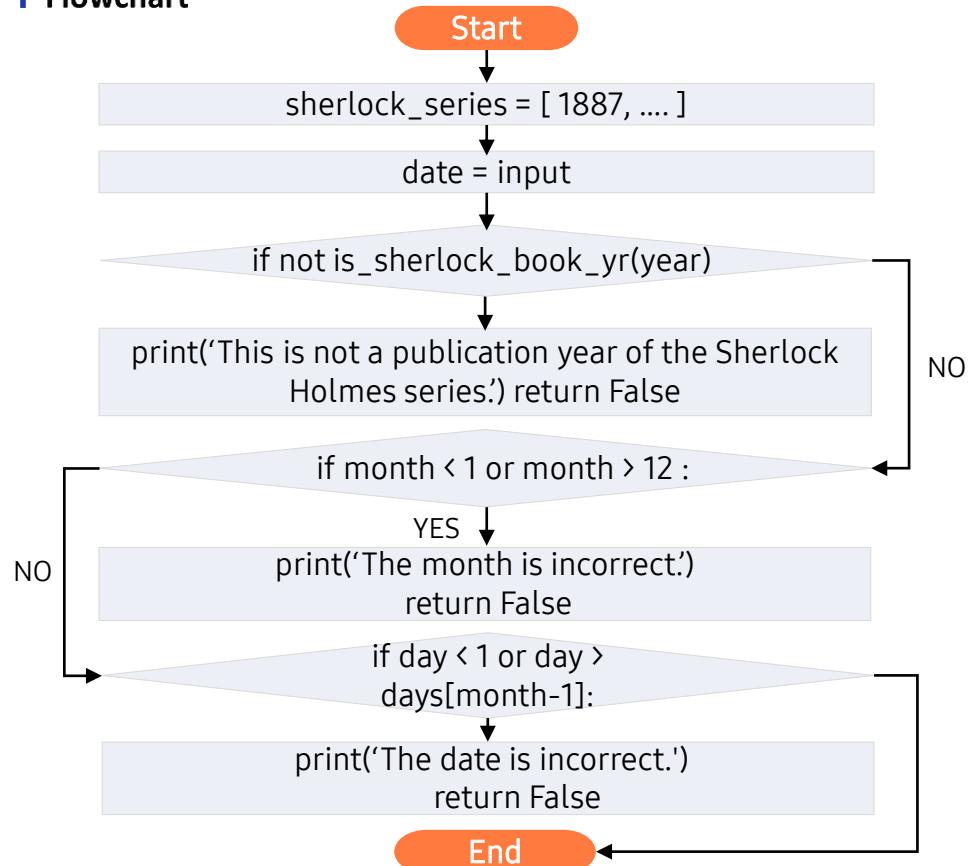
## 2. Mission

### 2.3. ချမှတ်သွေ့နည်း (valid\_date function)

#### Pseudocode

- [1] Start
- [2] Store publication years of the Sherlock Holmes series in a list
- [3] Receive dates as an input from the user
- [4] if the input date is not a publication year, then {  
    [5] return False with print statement and exit.}
- [6] if the month is incorrect then {  
    [7] return False with print statement and exit.}
- [8] if the date is incorrect then {  
    [9] return False with print statement and exit.}
- [10] End

#### Flowchart



## 2. Mission

### 2.4. ໂຄດຄໍາສັງສຸດທ້າຍຂອງໂປຣແກຣມ Sherlock Holmes

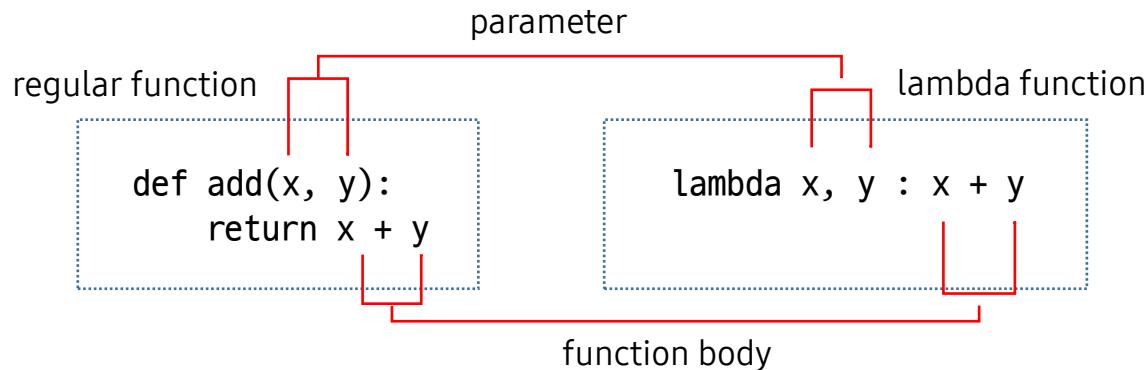
```
1 def valid_date(date):
2     days = [31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
3     sherlock_series = [1887, 1890, 1892, 1894, 1901, 1905, 1915, 1917, 1927]
4     splitted = date.split('.')
5     year = int(splitted[0])
6     month = int(splitted[1])
7     day = int(splitted[2])
8     is_sherlock_book_yr = lambda x: x in sherlock_series
9
10    if not is_sherlock_book_yr(year):
11        print('This is not a publication year of the Sherlock Holmes series.')
12        return False
13    if month < 1 or month > 12 :
14        print('The month is incorrect.')
15        return False
16    if day < 1 or day > days[month-1]:
17        print('The date is incorrect.')
18        return False
19
20    return True
21
22 date = input('Distinguish year, month and date by .(e.g. 1917.6.10) : ')
23 if valid_date(date) == True:
24     print('This is the publication date of a Sherlock Holmes series!')
25 else:
26     print('This is not the publication date of a Sherlock Holmes series or the input has an error.')
```

# | Key concept

# 1. Lambda Expression

## 1.1. Lambda expression ຫຼື lambda function syntax

- | ກຳນົດຄໍາສັ່ງ def ເພື່ອສິ່ງຄືນຜົນບວກຂອງປາຣາເມັດຕີ x, y ດ້ວຍການຂຽນຝັງຊັ້ນ add ດັ່ງສະແດງລຸ່ມນີ້.
- | ສ້າງຝັງຊັ້ນ def add: ຂຽນເນື້ອຫາ ແລະ ກຳນົດຄໍາສັ່ງກັບດ້ວຍຄໍາສັ່ງ return.
- | ໃນ Python, ເຮົາສາມາດສ້າງຝັງຊັ້ນສໍາລັບການໃຊ້ວຽກ ພຽງຄັ້ງດຽວ ໃຫ້ຄວບຄຸມຂັ້ນຕອນຮັດວຽກທັງໝົດໄດ້ ເຊິ່ງເອັນມັນວ່າ ຜັງຊັ້ນ lambda.
- | ຜັງຊັ້ນ lambda ເປັນຝັງຊັ້ນທີ່ບໍ່ມີຊື່ ຫຼື ເອັນອີກຢ່າງໜຶ່ງວ່າ ຄໍາສັ່ງ lambda. ເນື່ອງຈາກລັກສະນະດັ່ງກ່າວ ມັນຈຶ່ງຖືກເອັນວ່າ anonymous function.
- | ສົມທຽບຝັງຊັ້ນທົ່ວໄປ ແລະ ຜັງຊັ້ນ lambda : ຜັງຊັ້ນທົ່ວໄປຈະປະກອບມີ def keyword, ຊື່ຂອງຝັງຊັ້ນ, parameters, ສອງຈ້າ ແລະ body. ໃນຂະນະທີ່ ຜັງຊັ້ນ lambda ມີພຽງ parameter ແລະ function.



## 1. Lambda Expression

### 1.1. Lambda expression และ lambda function syntax

| ทำถาวมเข้าใจกับฟังชัน lambda และ สิ่งผ่าน argument. กำหนด lambda ด้วย keyword และ ป้อน arguments เข้าไปอีก.

```
lambda [parameter 1, parameter 2, ...] : [expression]
```



```
(lambda [parameter 1, parameter 2, ...] : [expression])(argument 1, argument 2, ...)
```

| syntax ของฟังชัน lambda เป็นลักษณะ เช่น: [parameters] and colons(:) ตั้งสหແດງລຸ່ມນີ້

```
lambda [parameters] : {expression}
```

## 1. Lambda Expression

## 1.2. Syntax ຂອງຟັຈຊັ້ນທີ່ວໄປ

**Ex** ຕົວຢ່າງໂຄດຄໍາສັ່ງທີ່ມີການສິ່ງເປີນບວກຂອງຈຳນວນໂດຍໃຊ້ຝ້າງຊັ້ນ add.

```
1 # addition using add() function
2 def add(x, y):
3     return x + y
4
5 print('sum of 100 and 200 :', add(100, 200))
```

sum of 100 and 200 : 300

| ເຖິງແມ່ນວ່າ ການດຳເນີນການຂອງຝຶ່ງຊັ້ນຈະເປັນໄປແບບ່ງ່າຍ ເຊິ່ງຈະຕ້ອງບວກຈໍານວນສອງຈໍານວນເຂົ້າກັນ, ແຕ່ໂຄດຄໍາສົ່ງຕ້ອງປະຕິບັດຕາມ syntax ຫັງໜີດທີ່ປະກອບມີ def keyword, ຊື່ຝຶ່ງຊັ້ນ, body ແລະ return.

## 1. Lambda Expression

### 1.3. Syntax ຂອງຝັງຊັນທົ່ວໄປ ແລະ ຝັງຊັນ lambda

- | ນຳໃຊ້ຝັງຊັນ lambda ເພື່ອດໍາເນີນການທີ່ຝັງຊັນ add ເຮັດໄດ້ ໂດຍບໍ່ຕ້ອງກຳນົດຝັງຊັນ.

```
1 # addition using the lambda function
2 add = lambda x, y: x + y
3 print('sum of 100 and 200 :', add(100, 200))
```

sum of 100 and 200 : 300

- | ກຳນົດຝັງຊັນ lambda ດ້ວຍຄໍາສໍາລັນ lambda. expression ຮັບຄ່າ x, y ດ້ວຍ parameters ແລະ returns x + y.
- | ຝັງຊັນໄດ້ບວກສອງ argument ເຂົ້າກັນ ໂດຍບໍ່ຈໍາເປັນຕ້ອງມີ def add(x, y):
- | ຕອນນີ້ lambda ໄດ້ເພີ່ມຕົວປ່ຽນ. ການເຮັດວຽກຂອງຕົວດໍາເນີນການ ດ້ວຍການປ້ອນສອງປາຣາມັດເຕີ ຄື 100, 200.
- | ນີ້ເປັນການຢືນຢັນວ່າ ຝັງຊັນ lambda ສາມາດທຳງານໄດ້ຕາມປຶກກະຕິ ຄືກັບຝັງຊັນທົ່ວໄປ.

## 1. Lambda Expression

### 1.3. Syntax ຂອງຝັງຊັນທົ່ວໄປ ແລະ ຝັງຊັນ lambda

| ຝັງຊັນ Lambda ເຮັດວຽກໄດ້ດີ ແມ່ນແຕ່ ຕົວປ່ຽນ add ກໍຍັງບໍ່ຈໍາເປັນຕ້ອງໃຊ້.

```
1 print('sum of 100 and 200 :', (lambda x, y: x + y)(100, 200))
```

sum of 100 and 200 : 300

#### Line 1

- ຝັງຊັນ Lambda ຈະດຳເນີນການ  $x + y$  ແລະ ສື່ງຄ່າກັບ.
- ສື່ງ arguments 100, 200 ໄປຢັ້ງ parameter x ແລະ y.

## 1. Lambda Expression

### 1.3. Syntax ຂອງຟັງຊັນທີ່ວໄປ ແລະ ພັງຊັນ lambda

| ພັງຊັນ lambda ເຮັດໃຫ້ຂຽນໂຄດໄດ້ຢ່າຍ.

```
1 a = [1, 2, 3, 4, 5, 6, 7]
2 square_a = list(map(lambda x: x**2, a))
3 print(square_a)
```

```
[1, 4, 9, 16, 25, 36, 49]
```

| ພັງຊັນ map ຈະໄດ້ຮູນໃນບົດຕໍ່ໄປ ມັນຄືກັບ ພັງຊັນ lambda.

| ໂຄດຄຳສັ່ງຂ້າງເທິງແມ່ນຄຳສັ່ງ lambda ແລະ ພັງຊັນ map ທີ່ມີການ return x ກໍາລັງສອງ.

## 2. filter, map, reduce

### 2.1. Syntax ຂອງຝັງຊັນ filter

- | Python ມີ ຝັງຊັນຫຼາກຫຼາຍຮູບແບບ ແລະ ພຶ້ງ filter ກຳປັນຫົ່ງໃນນັ້ນ.
- | ພຶ້ງຊັນ filter ຮັບອີງປະກອບທີ່ເຮັດຊໍ້ໄດ້ ແລະ ສິ່ງກັບສະເພາະອີງປະກອບທີ່ເປັນ True.
- | syntax ຂອງ ພຶ້ງຊັນ filter ສະແດງດັ່ງລຸ່ມນີ້.

```
filter((To be applied function, {iterable object})
```

## 2. filter, map, reduce

### 2.2. ពិវិះការងារខ្សោយខ្លួន filter

Ex មី list ទីរោបអប់ខ្លួនបានបញ្ជាយតិន. ខ្សោយតិនដែលមិនមេខ្លួន។

```
1 # return True for values over 19, and False for those that are not.
2 def adult_func(n):
3     if n >= 19:
4         return True
5     else:
6         return False
7
8 ages = [34, 39, 20, 18, 13, 54]
9 print('adults list :')
10 for a in filter(adult_func, ages): # filter ages by using filter () function
11     print(a, end = ' ')
```

```
Adults list :
34 39 20 54
```

#### Line 1 - 6

- ឯងខ្លួន adult\_func ចាប់តាំងពីតិនដែលមិនមេខ្លួន។ តិនដែលមិនមេខ្លួននឹងត្រូវត្រូវតិនដែលមិនមេខ្លួន។

## 2. filter, map, reduce

### 2.2. ពិវិះការងារខ្សោយតម្លៃ filter

Ex មី list ទីរោបអប់ខ្លួនបានបញ្ជាយតិន. ខ្សោយតម្លៃដែលធ្វើឡើងថា មីអាយុស្ថាប់ 19 ឆ្នាំ

```
1 # return True for values over 19, and False for those that are not.
2 def adult_func(n):
3     if n >= 19:
4         return True
5     else:
6         return False
7
8 ages = [34, 39, 20, 18, 13, 54]
9 print('adults list :')
10 for a in filter(adult_func, ages): # filter ages by using filter () function
11     print(a, end = ' ')
```

```
Adults list :
34 39 20 54
```

#### Line 8 - 11

- list ទីនេះ ages បានរាយចកចំដោយអាយុ. ផ្សោយតម្លៃ adult\_func ដ៏មួយតិវប៉ូន ages ទីនេះ arguments.
- ផ្សោយតម្លៃ filter ដោយអាយុ ដោយផ្សោយតម្លៃ adult\_func និង ត្រូវតាមតាមរយៈ True នៅក្នុង។
- សំណើនាយករាយ 18, 13 ប៉ុណ្ណោះមិននឹងត្រូវបានបញ្ជាយ។

## 2. filter, map, reduce

### 2.3. ລັກສະນະພິເສດຂອງຝ່າຍັນ filter ຊື່ adult\_func

| ລັກສະນະພິເສດຂອງຝ່າຍັນ adult\_func ດັ່ງສະແດງລຸ່ມນີ້.

1. ຂັ້ນຕອນວິທີຂອນຂ້າງ່າຍ
2. ບໍ່ຈໍາເປັນຕ້ອງມີຕົວປ່ຽນພາຍໃນເພີ່ມເຕີມ
3. ບໍ່ຈໍາເປັນຕ້ອງນຳມາໃຊ້ຊື້າ ຫຼັງຈາກຕອງຂໍ້ມູນແລ້ວ.

| ຖ້າຕ້ອງການໃຊ້ຝ່າຍັນຮັດວຽກ່າຍ ແລະ ບໍ່ຈໍາເປັນຕ້ອງໃຊ້ຊື້າ ກໍຕ້ອງໃຊ້ ພັກຊັນ lambda.

## 2. filter, map, reduce

### 2.3. ແປງຟ້າຂັ້ນ adult\_func filter ເປັນ ພັງຂັ້ນ lambda

- | ເຮົາສາມາດຫຼຸດຄວາມຊັບຊ້ອນຂອງພັງຂັ້ນ ໂດຍການແປງຟ້າຂັ້ນ adult\_func ເປັນພັງຂັ້ນ lambda.
- | ການນຳໃຊ້ ພັງຂັ້ນ lambda ສາມາດຊຽນໂຄດໄດ້ຈ່າຍ ແລະ ສັ້ນ ດັ່ງລຸ່ມນີ້.

```
1 ages = [34, 39, 20, 18, 13, 54]
2 print('adults list :')
3 for a in filter(lambda x: x >= 19, ages): # filter ages using filter () function
4     print(a, end = ' ')
```

```
adults list :
34 39 20 54
```

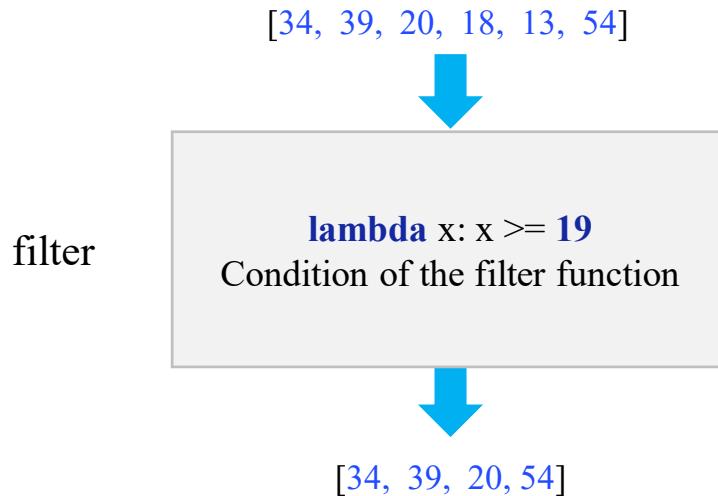
#### Line 3

- ສິ່ງຄ່າ True ສໍາລັບ x ຫຼາຍກວ່າຫຼືເທົ່າກັບ 19, ແລະ ພິມອາຍຸທີ່ໃຫຍ່ກວ່າເທົ່າກັບ 19 ອອກມາ.
- ຕອງຕາມເງື່ອນໄຂດັ່ງນີ້ lambda x: x>= 19.

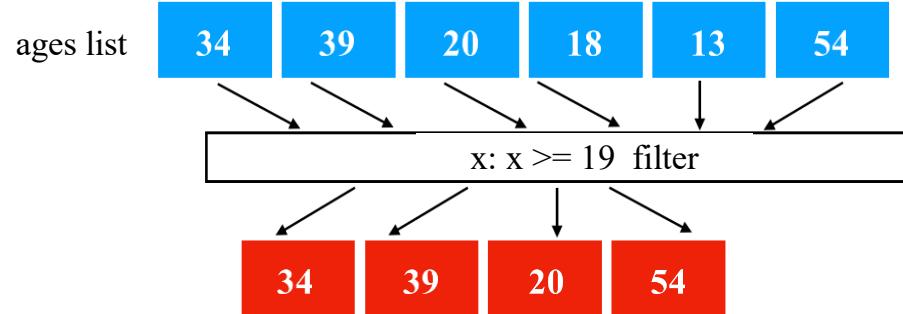
## 2. filter, map, reduce

### 2.4. ខ្សែនពីរការរោគខំដូចជា lambda តាមខ្លួន filter

- | ធនធាននេះ មែនជាបាយខ្សែនពីរការរោគ.
- | តាមតាមលេខ និងស៊ីវភៅ និង ស៊ីវភៅ នៃចំណែកខ្លួន lambda



The behavior of the filter function  
and the role of the lambda function



Iterable objects and returned values of the filter function

## 2. filter, map, reduce

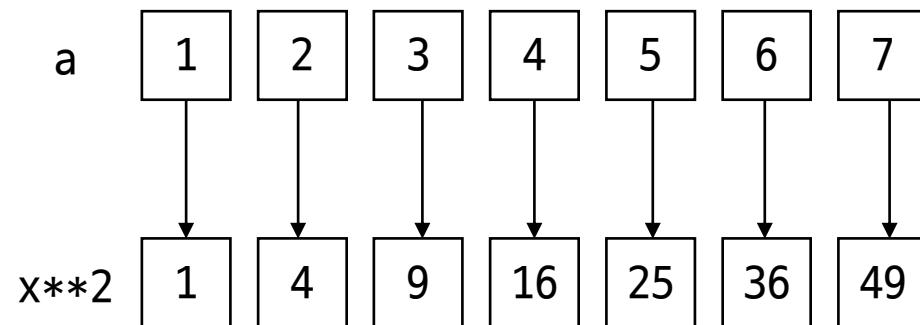
### 2.5. ການໃຊ້ຝັງຊັນ lambda

- | ສັງເກດພິດຕີກໍາຂອງຝັງຊັນ lambda ເນື່ອຮັດຕອງຂໍ້ມູນ.
- | ພັງຊັນ filter ເປັນຝັງຊັນຕອງຂໍ້ມູນຕາມຕື່ອນໄຂ ຈາກລາຍການຂໍ້ມູນ [34, 39, 20, 18, 13, 54]. ອາດຈະຕອງຂໍ້ມູນອາຍຸຕັ້ງແຕ່ 19 ປີ ຂຶ້ນໄປ, ແລະ ຕໍ່າກວ່າ 19 ປີ.
- | ພັງຊັນ lambda ຈະສື່ງຄ່າ True ຫຼື False ແມ່ນຂຶ້ນກັບຕື່ອນໄຂໃນການຕອງເຊັ່ນ:  $x \geq 19$  ຂອງ lambda  $x$ :
- | ພັງຊັນ filter ຮັບການຕອງຕາມຕື່ອນໄຂ ແລະ ເຮັດຊໍ້າຈິນກວ່າຈະໄດ້ຜົນຮັບ.

## 2. filter, map, reduce

## 2.6. ផែងតាម map

- | Python มีฟังก์ชันในตัวที่อึ้งว่า map. ฟังก์ชันนี้ จะสิ่งประพฤติข้อมูล ด้วยการ mapping function.
  - | ฟิจาระนากาสิ่งที่มีอยู่ใน list ที่มีอยู่ประกอบกัน [1, 4, 9, 16, 25, 36, 49] ด้วยการนำให้ตัวดำเนินการ กำลังสอง ของค่า [1, 2, 3, 4, 5, 6, 7] ด้วยส่วนต่อไปนี้



## 2. filter, map, reduce

## 2.7. ຄິດໄລ່ກຳລັງສອງຂອງຄ່າທັງໝົດໃນ list

**Ex** ຖ້າອີງປະກອບຂອງ list ທີ່ຊື່ a ດັ່ງສະແດງລຸ່ມນີ້, ໃຫ້ເພີ່ມຄ່າກໍາລັງສອງຂອງ list ດັ່ງກ່າວທັງໝົດທຸກຄ່າ.

- | เป็นภาษาແກ້ໄຂບັນຫາທີ່ເຮົາຮູ່ຢູ່ແລ້ວ.
  - | ຂຶ້ນກຳລັງສອງຂອງແຕ່ລະຄ່າໃນ list ທີ່ຂື່ a ຈາກນັ້ນ ໃຫ້ນໍາໃຊ້ ພິຈຊັນ append ເພື່ອເພີ່ມຄ່າດ້ວຍກ່າວລົງໄປໃນ square\_a.

```
1 a = [1, 2, 3, 4, 5, 6, 7]
2 square_a = []
3 for n in a:
4     square_a.append(n**2) # add n square to list square_a
5 print(square_a)
```

```
[1, 4, 9, 16, 25, 36, 49]
```

 Line 3, 4

- ກໍາລັງສອງຂອງເຕັລະຄ່າ ແລະ ເພີ່ມມັນລົງໄປໃນ square a

## 2. filter, map, reduce

### 2.8. ພິດຕິກຳຂອງຝັງຊັນ map

| ຈາກໂຄດຄໍາສັ່ງຊ້າງເທິງແມ່ນຂອນຂ້າງຍາວ ເຮົາສາມາດແກ້ໄຂບັນຫານີ້ ດ້ວຍໂດຍໃຊ້ຝັງຊັນ map ເພື່ອເຮັດໃຫ້ການຂຽນໂຄດສັນ ແລະ ຢ່າຍຂຶ້ນ ດັ່ງສະແດງລຸ່ມນີ້.

```
map(function_to be applied , iterable object, ...)
```

- | ພິງຊັນ map ຈະຮັບ argument ສອງຕົວຂຶ້ນໄປ. ທຳອິດເປັນ mapping function ແລະ ທີ່ສອງແມ່ນ object ທີ່ສາມາດເຮັດຊ້າ ເຊັ່ນ ລາຍການທີ່ຈະຖືກແຊກເຂົ້າໄປໃນ mapping function.
- | ດ້ວຍການນຳໃຊ້ຝັງຊັນ map, ຄ່າແຕ່ລະຄ່າໃນ object ຄືກັບ list ແມ່ນຖືກນຳໄປໃຊ້ໃນ mapping function ແລະ returns list [1, 4, 9, 16, 25, 36, 49] ເຊິ່ງມາຈາກກຳລັງສອງຂອງຄ່າ [1, 2, 3, 4, 5, 6, 7]

```
1 def square(x):  
2     return x ** 2  
3  
4 a = [1, 2, 3, 4, 5, 6, 7]  
5 # apply the return value of square function to each term of a  
6 square_a = list(map(square, a))  
7 print(square_a)
```

[1, 4, 9, 16, 25, 36, 49]

## 2. filter, map, reduce

### 2.9. ការລວມកំណើងមិនមែន map នៅលើ lambda

- | ក្នុងកំណាំស៉ាងទាំងមិនមែន map នៅលើ lambda
- | នឹងសាមាត្រូវការអប់អ៊ូនខ្សោយក្នុងកំណាំស៉ាងដោយបំឈាន់ថា មិនមែន map នៅលើ lambda. មិនមែន lambda ត្រួតវិភាគតុងក្បរវិប នៅលើមិនមែនមិនមែន lambda.

```
1 a = [1, 2, 3, 4, 5, 6, 7]
2 square_a = list(map(lambda x: x**2, a))
3 print(square_a)
```

```
[1, 4, 9, 16, 25, 36, 49]
```

#### Line 2

- នៃមិនមែន map មិនមែន lambda ផ្លូវ returns តាមការលើសរុប នៅលើការបង្ហាញរបស់រាយការណ៍ list.
- ពេលតាមការលើសរុបខ្សោយ a ត្រូវក្រោចតាមការលើសរុបខ្សោយ square\_a ដូចជាបញ្ជីក្នុងរបស់រាយការណ៍ list.

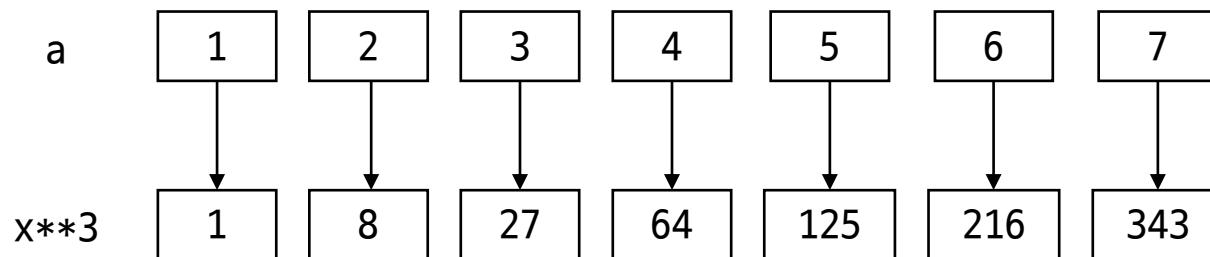
## 2. filter, map, reduce

### 2.9. ການລວມກັນຂອງຝັງຊັນ map ແລະ lambda

| ເຮົາສາມາດແປງໂຄດຄຳສັ່ງໃຫ້ສາມາດເປັນກຳລັງສາມໄດ້ດັ່ງລຸ່ມນີ້.

```
1 a = [1, 2, 3, 4, 5, 6, 7]
2 cubic_a = list(map(lambda x: x**3, a))
3 print(cubic_a)
```

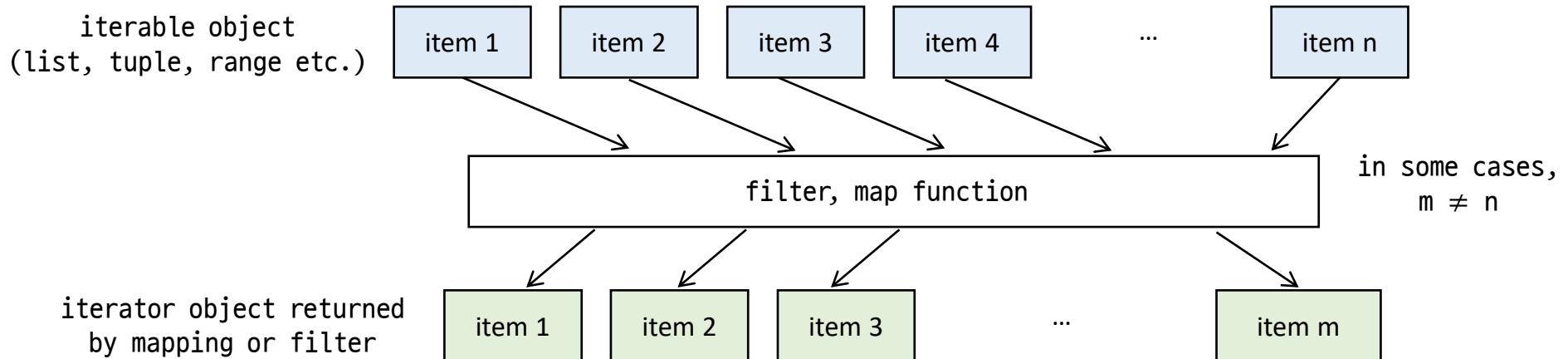
[1, 8, 27, 64, 125, 216, 343]



## 2. filter, map, reduce

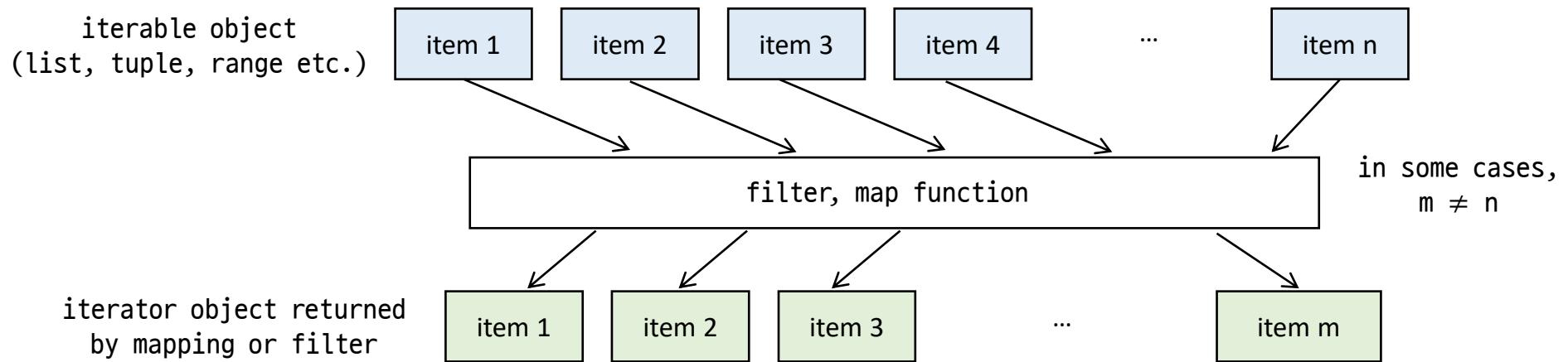
### 2.10. បច្ចេកទឹន iterable និង ផ្សេងៗ filter, ផ្សេងៗ map

- | នៃ Python មីផ្សេងៗនៃពិវិះអ៊ីនវ៉ា filter និង map. ដំឡើង mapping function រាប់ព័ត៌មានរបស់រាយការណ៍បច្ចេកទឹន iterable.
- | ផ្សេងៗនេះមិនត្រូវត្រួតពិនិត្យ iterable objects និង object ត្រូវត្រួតពិនិត្យប៉ុណ្ណោះប៉ុណ្ណោះដូចជា list ដែលការងារនេះមិនត្រូវត្រួតពិនិត្យ។



## 2. filter, map, reduce

### 2.10. ປະເພດຂໍ້ມູນ iterable ແລະ ພັງຊັນ filter, ພັງຊັນ map



- | ແຕ່ລະລາຍການຂອງ ປະເພດຂໍ້ມູນ iterable ເຊັ່ນ: list, tuple, range ອື່ນໆ ແມ່ນປັບປຸງໂດຍ mapping functions ເຊັ່ນວ່າ filter ຫຼື map ແລະ ມີການສິ່ງຄ່າກັບ. ແຕ່ລະຄ່າຂອງ iterator object ແລະ object ສາມາດ ປັບປຸງກັບໄປເປັນປະເພດຂໍ້ມູນ iterable ໂດຍ ນຳໃຊ້ປະເພດຂໍ້ມູນ.

## 2. filter, map, reduce

### 2.10. ປະເພດຂໍ້ມູນ iterable ແລະ ພັງຊັນ filter, ພັງຊັນ map

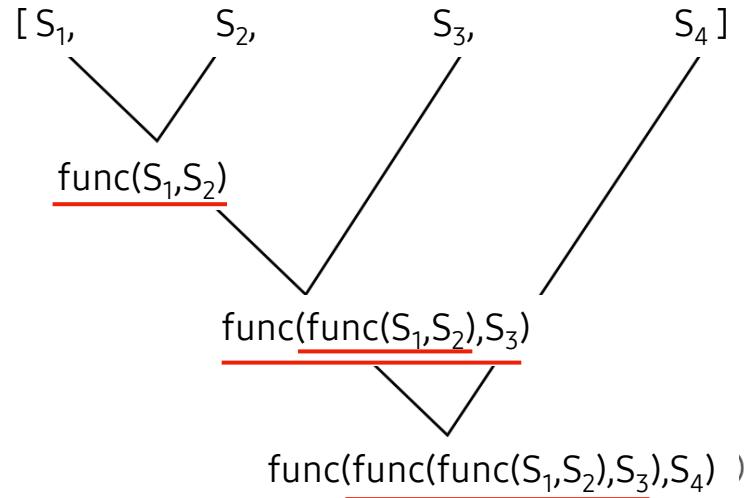
 Focus ມີໃຊ້ພັງຊັນ lambda ເທິງ filter ແລະ map ເພື່ອຫຼຸດພັງຊັນໃຫ້ໜ້ອຍລົງ.

- | ເປັນຫຍັງຈຶ່ງ filter ແລະ map ອີງປະກອບຂອງ ປະເພດຂໍ້ມູນ iterable ເຊັ່ນ: list, tuple, range ແລະ ອື່ນໆ. ໂດຍນຳໃຊ້ ພັງຊັນ filter ແລະ map?
- | ວຽກຄັ້ງກ່າວສາມາດຮັດໄດ້ໂດຍການເອັນໃຊ້ພັງຊັນ, ແຕ່ການເອັນໃຊ້ພັງຊັນຫຼາຍເກີນໄປ ກໍຈະຮັດໃຫ້ເກີດບັນຫາ ໃນລະຫວ່າງການເອັນໃຊ້ພັງຊັນ ມັນຈະ ບັນທຶກສະຖານະກ່ອນ ແລະ ຫຼັງການເອັນໃຊ້, ມັນຈະບັນທຶກສະຖານະກ່ອນໜັ້ນອີກຄັ້ງ ແລະ ດໍາເນີນການແບບນີ້ໄປເລື່ອຍ່າງ ເຊິ່ງມັນຈະກໍໃຫ້ເກີດບັນຫາ.
- | ມັນຈະຮັດໃຫ້ຄວາມໄວໃນການຮັດວຽກຂອງໂປຣແກຣມຊ້າລົງ.

## 2. filter, map, reduce

### 2.11. reduce function

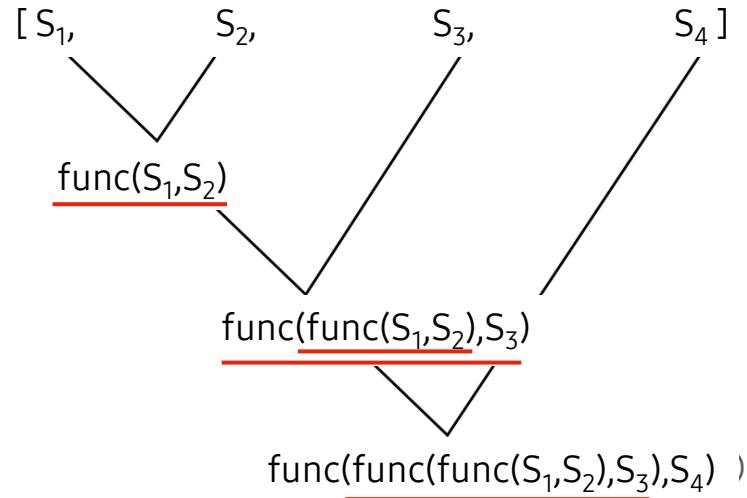
- | Reduce function ແມ່ນຢູ່ໃນ module ທີ່ເອີ້ນວ່າ `functool`. ມັນຈະສິ່ງຄ່າກັບເປັນຄ່າດຽວ ໂດຍດຳເນີນການ ດ້ວຍພັງຊັ້ນທີ່ກຳນົດ ກັບລາຍການຂອງ iterable object.
- | ພິຈາລະນາຕົວຢ່າງລຸ່ມນີ້. list ທີ່ຊື່ `seq = []`, ການດຳເນີນການຂອງ `reduce (func, seq)` function ດັ່ງລຸ່ມນີ້.



## 2. filter, map, reduce

### 2.11. reduce function

- | อิงປະກອບທຳອິດ ແລະ ອີງປະກອບທີ່ສອງ ຂອງ list, s1 ແລະ s2 ກໍານົດໃຫ້ຝຶງຊັ້ນ func, ເຊິ່ງການເຮັດວຽກຂອງ parameter ກໍຈະຫຼຸດ ຈາກນັ້ນ ຄ່າຈະຖືກ returned ໂປ ພູ້ໃຫ້ເປັນ input ສໍາລັບຝຶງຊັ້ນຕໍ່ໄປ.
- | ການກະທຳ  $\text{func}(\text{func}(, ), )$ . ແມ່ນການວິນຊ້າຈົນກະທັງລາຍການສຸດທ້າຍທີ່ໄດ້ຮັບ input
- | ຫີ້ສຸດມັນກໍຈະສົ່ງຄ່າດຽວກັບ



## 2. filter, map, reduce

## 2.12. ຕົວຢ່າງ reduce function

## Ex กามນាໃຊ້ reduce function

```
1 from functools import reduce  
2  
3 a = [1, 2, 3, 4]  
4 n = reduce(lambda x, y: x + y, a)  
5 print(n)
```

10

- | 1, 2, 3, 4 ແມ່ນອີງປະກອບຂອງ list ທີ່ຊື່ a.
  - | ເລີ່ມຕົ້ນ, 3 ຖືກສິ່ງຄືນຈາກ ພັງຊັນ lambda ເຊັ່ນ lambda x, y: x+y ເມື່ອມີການປ້ອນຄ່າ 1 ແລະ 2 .
  - | ຈາກນັ້ນ return ຄ່າ 3, ສໍາລັບອີງປະກອບທີ 3 ໂດຍມີຄ່າເທົ່າ 3, ແມ່ນຖືກປ້ອນໃຫ້ພັງຊັນ lambda. ແລ້ວມັນຈະສິ່ງຄ່າ 6 ກັບ ເຊິ່ງເປັນຜົນບວກຂອງສອງຄ່າ, ຈາກນັ້ນ 6 ແລະ ຄ່າສຸດທ້າຍ 4 ແມ່ນຖືກປ້ອນເຂົ້າໄປທີ່ພັງຊັນ lambda ແລ້ວສິ່ງຄ່າ 10 ອອກມາ ຖື່ວ່າຈີບການເຮັດວຽກຂອງພັງຊັນ.

`((1 + 2) + 3) + 4) => returns 10`

# Paper coding

- ຕ້ອງເຂົ້າໃຈແນວຄິດພື້ນຖານຂອງຫຼັກສູດນີ້ໃຫ້ຄົບຖ້ວນກ່ອນຈະໄປສູ່ຂັ້ນຕອນຕໍ່ໄປ.
- ການທີ່ບໍ່ເຂົ້າໃຈແນວຄິດພື້ນຖານ ຈະເພີ່ມພາລະໃນການຮຽນຮູ້ຂອງຫຼັກສູດນີ້ ແລະ ຈະຮັດໃຫ້ບໍ່ປະສົບຜົນສໍາເລັດ.
- ມັນອາດຈະເປັນເລື່ອງທີ່ຍາກຕອນນີ້, ແຕ່ຖ້າຢາກປະສົບຜົນສໍາເລັດໄດ້ນັ້ນ ພວກເຮົາຂໍແນະນຳໃຫ້ເຂົ້າໃຈແນວຄິດພື້ນຖານ ນີ້ຢ່າງເລີກເຊິ່ງ ແລະ ກ້າວໄປສູ່ຂັ້ນຕອນຕໍ່ໄປ.

**Q1.** ມີ list ທີ່ມີອີງປະກອບເປັນຈຳນວນຖ້ວນ ຊື່ n. list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]. ໃຫ້ສ່ຽງ even\_list ກັບ ສະເພາະອີງປະກອບທີ່ເປັນເລກຄຸ້ມ ຈາກ n\_list ໂດຍນໍາໃຊ້ຝຶ່ງຊັນ filter ແລະ ພຶ່ງຊັນ lambda.

The returned even\_list = [2, 4, 6, 8, 10]

Conditions for Execution	even_list = [2, 4, 6, 8, 10]
Time	5 Minutes

Create an empty list named even\_list and add even value items by the append method. Use the for statement and the filter function. Use lambda function inside the filter function.

**Q2.**

ມີ list ທີ່ມີອີງປະກອບເປັນຈຳນວນຖ້ວນ ຊື່ n\_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]. ໃຫ້ສິ່ງ even\_list ກັບ ສະເພາະອີງປະກອບທີ່ເປັນ  
ເລກຖ່ຽນ ຈາກ n\_list ໂດຍນໍາໃຊ້ຟັງຊັນ lambda ໃນນີ້ປໍ່ໃຫ້ໃຊ້ statement ແຕ່ໃຫ້ໃຊ້ຟັງຊັນ list ແກນ.

Returned even\_list = [2, 4, 6, 8, 10]

Conditions for Execution	even_list = [2, 4, 6, 8, 10]
Time	5 Minutes

Modify objects returned by the filter function to list objects by the list function, and assign them to even\_list.



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

**Q3.**

ຂຽນພື້ນຖານ map ເພື່ອປັບປຸງ a\_list ເຊິ່ງບັນຈຸຕົວອັກສອນນີ້ອຍ ['a', 'b', 'c', 'd'] ໃຫ້ເປັນຕົວອັກສອນໃຫຍ່ ແລະ ເກັບໄວ້ໃນຕົວປັງ  
upper\_a\_list = ['A', 'B', 'C', 'D'].

ທຳອິດ ໃຫ້ກໍານົດພື້ນຖານ ທີ່ຊື່ to\_upper ເພື່ອຮັບຄ່າເປັນຕົວອັກສອນນີ້ອຍເຊິ່ງເປັນ parameter ແລະ returns ຕົວອັກສອນໃຫຍ່ອອກມາ.

Conditions for Execution	upper_a_list = ['A', 'B', 'C', 'D']
Time	5 Minutes



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

**Q4.**

ถ้าให้ผู้เขียนบวกของจำนวนเต็ม จาก 1 ถึง 100 ด้วยภาษา Python ใช้ reduce function และ lambda expression. โดยป้อนค่า range (1, 101).

Condition for Execution

Sum of 1 to 100 : 5050

Time

5 Minutes

| Let's code

## 1. จัดการกับข้อผิดพลาด

### 1.1. สิ่งสำคัญสี่ลับ งานจัดการกับข้อผิดพลาด

| มีขั้นตอนที่ต้องได้รับโดยการตัดคำสั่งสี่ลับข้อผิดพลาด. ลุ่มๆ เมื่อทำงานของโดยการตัดคำสั่งที่บ่อดี เนื่องจากมีการทำงานจัดการกับข้อผิดพลาดในกำลังนี้ป้อนข้อมูล บ่ถูกต้อง. เพื่อแก้ไขขั้นหาด้วยว่าทุกอย่างในงานของโดยการตัดคำสั่งต้องใส่คำสั่ง try ~ except

```

1 try:
2     a, b = input('Enter two numbers : ').split()
3     result = int(a) / int(b)
4     print('{}/{} = {}'.format(a, b, result))
5 except :
6     print('Check if the numbers are correct.')

```

Enter two numbers : 10 2  
10/2 = 5.0

ป้อนค่าที่ถูกต้อง.

```

1 try:
2     a, b = input('Enter two numbers : ').split()
3     result = int(a) / int(b)
4     print('{}/{} = {}'.format(a, b, result))
5 except :
6     print('Check if the numbers are correct.')

```

Enter two numbers : 100 two  
Check if the numbers are correct.

except statement ทำอย่างไรเมื่อมองเห็นเชิงเดิม ให้เมื่อว่า จะมีการป้อน string 'two' ตาม.

Enter two numbers : 100 two  
Check if the numbers are correct.

ถ้าป้อน 100 และ 0, โดยคำสั่งจะพยายามหาร 100/0. except statement จะจัดการกับข้อผิดพลาด

## 1. ຈັດການກັບຂໍ້ຍິກເວັນ

### 1.2. ບັນດາຂໍ້ຍິກເວັນ

- | ມີຊະນິດຂໍ້ຍິກເວັນຫຍ່ງແດ່?
- | ໃນຕາຕະລາງດ້າຍຂວາແມ່ນບັນດາຂໍ້ຍິກເວັນຊະນິດຕ່າງໆ.
- | ເຮົາຈະສຶກສາຂໍ້ຍິກເວັນເລື່ອນີ້ໃນພາຍຫຼັງ.

BaseException	ProcessLookupError	UnboundLocalError
Exception	TimeoutErrorReferenceError	OSError
ArithmetError	RuntimeError	BlockingIOError
FloatingPointError	NotImplementedError	ChildProcessError
OverflowError	RecursionError	ConnectionError
ZeroDivisionError	StopIteration	BrokenPipeError
AssertionError	StopAsyncIteration	ConnectionAbortedError
AttributeError	SyntaxError	ConnectionRefusedError
BufferError	IndentationError	ConnectionResetError
EOFError	TabError	FileExistsError
ImportError	SystemError	FileNotFoundError
ModuleNotFoundError	TypeError	InterruptedError
LookupError	ValueError	IsADirectoryError
IndexError	UnicodeError	NotADirectoryError
KeyError	UnicodeDecodeError	PermissionError
MemoryError	UnicodeEncodeError	FutureWarning
NameError	UnicodeTranslateError	ImportWarning

## 1. จัดการกับข้อผิดพลาด

### 1.2. บันดาลข้อผิดพลาด

| เริ่มมาทดสอบข้อผิดพลาด ด้วยการคำสั่งลุ่มมี. print('error :', e) .

```
1 try:  
2     b = 2 / 0  
3     a = 1 + 'hundred'  
4 except Exception as e:  
5     print('error :', e)
```

object 2 contains the information of the exception.

error : division by zero

| ໂຄດคำสั่งนี้ແມ່ນການຫານດ້ວຍສູນ ຈຶ່ງເກີດ error ແລ້ວ.

## 💡 อิກຂັ້ນຕອນໜຶ່ງ

### 1. Exception Handling

- | ການຈັດການກັບຂໍຢີກເວັ້ນທີ່ຊັບຊັອນ ສາມາດາປະຕິບັດໄດ້ໂດຍການນຳໃຊ້ try - except – else – finally.
- | ໂດຍຄໍາສົ່ງລຸ່ມນີ້ ບັນຈຸຄໍາສົ່ງສະເພາະໃຫ້ສະແດງ ZeroDivisionError ມີຄໍາສົ່ງດໍາເນີນການ ໂດຍບໍ່ສິນໃຈຂໍຢີກເວັ້ນ.

```
1 def divide(x, y):
2     try:
3         result = x / y
4     except ZeroDivisionError:
5         print('error by zero division')
6     else:
7         print('result :', result)
8     finally:
9         print('execution complete')
10
11 print('divide(100, 2) function call :')
12 divide(100, 2)
13 print('divide(100, 0) function call :')
14 divide(100, 0)
```

```
divide (100, 2) function call :
result : 50.0
execution complete
divide (100, 0) function call :
error by zero division
execution complete
```

## 💡 ອີກຂັ້ນຕອນໜຶ່ງ

### 1. Exception Handling

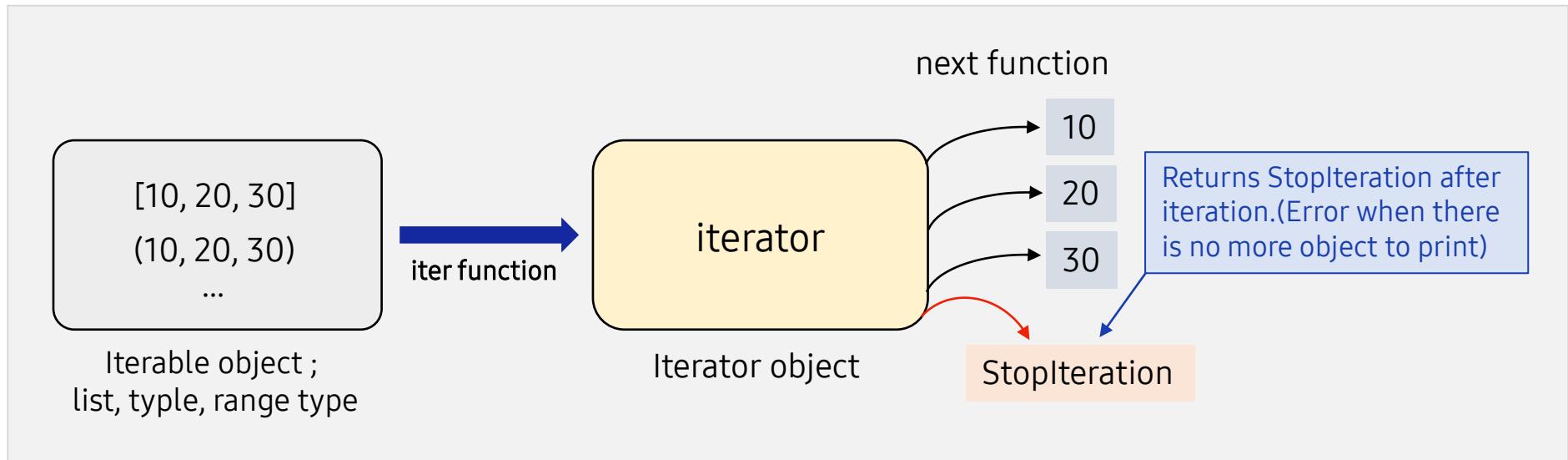
| ກວດສອບພື້ນຖານສະເພາະຂອງຄໍາສັ່ງ try – except –else –finally .

ຊື່	ລາຍລະອຽດ
try:	ເຊື້ອຍົກເວັນບໍ່ເກີດຂຶ້ນ ເນື່ອງຈາກຄໍາສັ່ງ ໄດ້ຮັບການປະມວນຜົນແລ້ວ, ມັນກໍຈະຂ້າມໄປທີ່ except. ແຕ່ຖ້າມີຂໍ້ອົກເວັນເກີດຂຶ້ນ ລະບົບຈະກວດສອບຂໍຜິດພາດ ແລະ ສິ່ງຜ່ານໄປທີ່ except.
except:	block ນີ້ ເຮັດວຽກກໍລະນີມີຂໍ້ອົກເວັນ.
else:	block ນີ້ເຮັດວຽກເມື່ອມີຂໍ້ອົກເວັນ.
finally:	block ເຮັດວຽກຕະຫຼອດໂດຍບໍ່ສິນໃຈຂໍ້ອົກເວັນ.

## 2. Iterator Object

### 2.1. Iterator เป็น object ที่เรียกวากฎเบื้องหลังคำสั่งวินช์ของ Python

- | Iterator เป็นห้องในฟังก์ชันลับดับสูงของ Python.
- | Iterator object เป็นวัตถุที่ใช้ข้อมูลติดต่อกันจากโถงส้างขั้มุนที่มีห้องอิงประกอบขึ้นໄປ.



## 💡 ອີກຂັ້ນຕອນໜຶ່ງ

### 2.1. Iterator Object

- | List ຂອງ Python ເຊັ່ນ: [10, 20, 30] ຖືກເອັ້ນຊ້າຫວາຍຄັ້ງ ດັ່ງນີ້ for i in [10, 20 ,30]: .
- | ຈະຮູ້ໄດ້ແນວໃດວ່າ ຄໍາສັ່ງ for ເຖິງອີງປະກອບສຸດທ້າຍໃນ list [10, 20, 30]?
- | ສິ່ງນີ້ແປນໄປໄດ້ເນື້ອງຈາກ object ທີ່ຊື່ StopIteration ແມ່ນຖືກເຊື່ອງ ແລະ ມີການສິ່ງຄືນອີງປະກອບສຸດທ້າຍ [10, 20, 30] ເຊັ່ນ [10, 20, 30, StopIteration]
- | StopIteration object ຈະແຈ້ງໃຫ້ຜູ້ໃຊ້ຮູ້ວ່າ ບໍ່ມີລາຍການຕໍ່ໄປທີ່ຈະສິ່ງຄືນໃນ iteration loop.
- | Object ນີ້ຖືກສ້າງໂດຍພັງຊັນໃນຕົວ ເຊັ່ນວ່າ method \_\_next\_\_ .
- | ຂໍ້ມູນເພີ່ມຕີມໃຫ້ໄປເບິ່ງເອກະສານ Python development.

## 2. Iterator Object

### 2.2. ฟังก์ชันในตัวสำหรับ iterable objects

- | ฟังก์ชันในตัวสำหรับทุกๆ ตัวใน Python ที่สามารถนำไปใช้ใน iterable object.
- | ดูน้ำผลักดัน min หรือ ฟังก์ชัน max รับ iterable objects เป็น input และ สิ่งที่ต้องการเป็นค่าม้อยสุด และ ค่าใหญ่สุด. นอกจากนี้ยังมีฟังก์ชันในตัวอื่นๆ เช่น: all, any, ascii, bool, filter, iter อีกด้วย.

## 2. Iterator Object

### 2.2. Iterator และ iter, next function

- | ข้อมูล列表 list ดังนี้ [10, 20, 30] ได้ถูกแปลงเป็น ประเภท list\_iterator ด้วยฟังก์ชัน iter .
- | ฟังก์ชัน iter รับข้อมูลอินพุต เป็นตัว input และ returns iterators.
- | โดยที่อินพุตให้ฟังก์ชัน iter คุ่นับฟังก์ชัน next.
- | ฟังก์ชัน next จะรับ iterator ที่ input เข้ามา และ ส่งอิงประจําตัวกับม้า.

```
1 iter_a = iter([10, 20, 30])
2 next(iter_a)
```

10

```
1 next(iter_a)
```

20

## 💡 ອີກຂັ້ນຕອນໜຶ່ງ

### 2.2. Iterator ແລະ iter, next function

- | ສໍາລັບ list a=[10, 20, 30], ຖ້າເອີນ next(a) ເພື່ອເບິ່ງອີງປະກອບຂອງ list, ຈະເກີດ error ຂຶ້ນ. ເນື່ອງຈາກ list object ບໍ່ສາມາດ return ອີງປະກອນຕິດຕໍ່ກັນໄດ້.
- | ພັງຊັນ iter ຂອງ Python ສ້າງ list ເຂົ້າໄປທີ່ iter\_a object.
- | iter\_a ຈະສົ່ງຄ່າທີ່ຊ້າງ ດ້ວຍ next(iter\_a)
- | ພັງຊັນ next ເອີນອີງປະກອບທີ່ຊ້າງໄດ້ຢ່າງຕໍ່ເນື່ອງ.

```
1 a = [10, 20, 30]
2 next(a)
```

```
TypeError
<ipython-input-4-e4a568985220> in <module>
      1 a = [10, 20, 30]
----> 2 next(a)

TypeError: 'list' object is not an iterator
```

```
1 iter_a = iter([10, 20, 30])
2 next(iter_a)
```

10

```
1 next(iter_a) # Returns the next time from the iterator
```

20

## 2. Iterator Object

### 2.3. ປະເພດຂໍ້ມູນ Non-iterable ແລະ ການນຳໃຊ້ຝັງຊັນ iter



```
1 n = 100 # Non-iterable data type int
2 n_iter = iter(n)
```

```
TypeError Traceback (most recent call last)
<ipython-input-18-59ae9a7950d8> in <module>
      1 n = 100 # Non-iterable data type int
----> 2 n_iter = iter(n)
```

**TypeError**: 'int' object is not iterable

- ▶ ถ้าสิ่งประพฤติเป็น int ที่มีค่าเท่า 100 เป็น argument ให้กับฟังก์ชัน iter, จะเกิด TypeError ขึ้น.
  - ▶ ประพฤติขึ้นมุ่งจามวนทั่ว int บล็อกมาดปรุ่มเป็น iterator object เพาะว่า มันบล็อกมุ่งประพฤติขึ้นมุ่ง iterable.

## 2. Iterator Object

### 2.4. ផ្សេងៗនៃ `next` និងការប្រើប្រាស់របៀបទូទាត់ជាការការពី `iterator object`

```
1 lst = [10, 20, 30]
2 l_iter = iter(lst) # Converts list type object to an iterator
3 type(l_iter)
```

list\_iterator

```
1 next(l_iter)
```

10

```
1 next(l_iter)
```

20

```
1 next(l_iter)
```

30

| ធានាធាមស្ថាប់ list object ទី 1.

| ទី 2. object មីនា រាយការណ៍ iter និង ពិនិត្យនូវ l\_iter. នឹងសារតារាងនៃលទ្ធផល បានបញ្ជាក់ថា បានបង្ហាញទី 1 និងទី 2 ដែលជាការការពី iterator object.

| ទី 3. បានបង្ហាញទី 3 ដែលជាការការពី iterator object.

| ចុចិត្តនៃការប្រើប្រាស់ list object គឺជាអ្នកបង្ហាញទី 1, 2, 3.

## 2. Iterator Object

### 2.5. StopIteration Exception

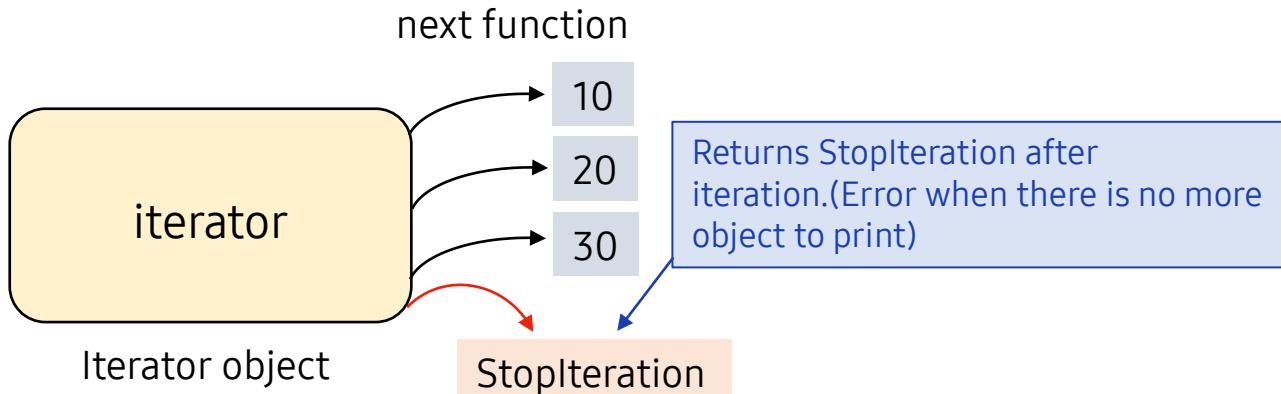
#### Return StopIteration exception

```
1 next(l_iter)
```

```
StopIteration
<ipython-input-25-43aece5a3964> in <module>
----> 1 next(l_iter)
```

```
StopIteration:
```

- ▶ ถ้าอีเม้นฟังชัน next สำลับ iterator object มันจะสิ่งอยู่ประกอบทั้งหมดกับ, ฟังชันจะส้างขึ้นยิกเว้น StopIteration.



## 2. Iterator Object

### 2.6. ນໍາໃຊ້ ວິທີການສະເພາະ `__next__`

- | method ໃນ Python ທີ່ເຮັດວຽກສະເພາະ ເຄື່ອນວ່າ special method.
- | ການຂຽນ special method ແມ່ນໃຊ້ເຄື່ອງໝາຍ `_` ໄວໜ້າ ແລະ ຫຼັງ ພັງຊັນ ດັ່ງນີ້ `__next__()`
- | ເຄື່ອງໝາຍຕັ້ງກ່າວໃຊ້ເພື່ອກຳນົດ operator ຫຼື ພັງຊັນໃໝ່, ເຊິ່ງຖືກກຳນົດໄວ້ໃນ Python.
- | ຄືທີ່ຜ່ານມາ, ມັນສ້າງຕົວປ່ຽນທີ່ຊື່ `__iter__` ແລະ ແປງ `list` ທີ່ຊື່ `lst` ເປັນ iterator object ທີ່ເຮັດໜ້າທີ່ເປັນ input.
- | ‘`object.__next__`’ ເຮັດໜ້າທີ່ຄືກັບພັງຊັນ `next`. ມັນຈະ returns ເມື່ອມີ object ສ່າງຄືນທີ່ນັ້ນ.

```
1 lst = [10, 20, 30]
```

```
1 l_iter = iter(lst) # converts list type object into an iterator
```

```
1 l_iter.__next__() ← Call special method __next__
```

10

```
1 l_iter.__next__()
```

20

```
1 l_iter.__next__()
```

30

## 💡 ອີກຂັ້ນຕອນໜຶ່ງ

### 2.6. ການນຳໃຊ້ special method `__next__`

- | ການດຳເນີນການ  $10 + 20$  ໄດ້ຄ່າ 30.
- | ນີ້ແມ່ນຕົວດຳເນີນການບວກ ໂດຍນຳໃຊ້ special method `__add__`.
- | ການບວກ  $10 + 20$  ແມ່ນໃຊ້ຝັງຂັ້ນສະເພາະຄື `(10).__add__(20)`.
- | Python ມີຫຼາຍ method ຫີ້ຂຽນ `__XXX__`. ແມ່ນ special methods.
- | ກ່ຽວກັບ class ແລະ method ເພີ່ມຕີມແມ່ນຈະອະທິບາຍຢູ່ Unit 21.

1	10	+	20
30			

1	(10)	.	__add__	(20)
30				

## 2. Iterator Object

### 2.7. ໂຄດຄໍາສັງເກືອດສອບວ່າ object ທີ່ສາມາດຮັດຊ້າໄດ້ຫຼືບໍ່

```
1 try:  
2     l = [10, 20, 30]  
3     iterator = iter(l)  
4 except TypeError:  
5     print('list is not an iterable object.')  
6 else:  
7     print('list is an iterable object.')
```

list is an iterable object.

#### Line 1 - 3

- ນໍາໃຊ້ iter (l), ເພື່ອກວດສອບວ່າ object l ສາມາດປ່ຽນເປັນ iterator ໃນ try :
- ຂໍຟິກເວັ້ນບໍ່ເກີດຂຶ້ນ.
- ສະນັ້ນ ຂໍຄວາມ 'list is an iterable object.' ປະກິດຂຶ້ນ.

## 2. Iterator Object

### 2.7. ໂຄດຄໍາສັງຫຼືກວດສອບວ່າ object ທີ່ສາມາດຮັດຊ້າໄດ້ຫຼືບໍ່

```
1 # verifies if the tuple is an iterable object
2 try:
3     t = ('Hong gil dong', 22, 79.7)
4     iterator = iter(t)
5 except TypeError:
6     print('tuple is not an iterable object.')
7 else:
8     print('tuple is an iterable object.)
```

tuple is an iterable object.

#### Line 1 - 3

- ນໍາໃຊ້ iter (l), ເພື່ອກວດສອບວ່າ object l ສາມາດປ່ຽນເປັນ iterator ໃນ try :
- ຂໍຢັກເວັ້ນບໍ່ເກີດຂຶ້ນ.
- ສະນັ້ນ ຂໍຄວາມ 'list is an iterable object.' ປະກິດຂຶ້ນ.

## 2. Iterator Object

### 2.7. ໂຄດຄໍາສັງຫຼືກວດສອບວ່າ object ທີ່ສາມາດຮັດຊ້າໄດ້ຫຼືບໍ່

```
1 # verifies if the tuple is an iterable object
2 try:
3     n = 100
4     iterator = iter(n)
5 except TypeError:
6     print('n is not an iterable object.')
7 else:
8     print('n is an iterable object.'
```

n is not an iterable object.

#### Line 2 - 4

- ຜ່ານ iter(n) ໃນ try : ເພື່ອກວດສອບ object n ທີ່ຖືກແປງເປັນ iterator
- ເກີດຂໍ້ຢັກເວັນ. except: ຜ່ານຂໍ້ຢັກເວັນ.
- ຂໍ້ຄວາມ ‘is not an iterable object’ ປະກິດຂຶ້ນ.

## 2. Iterator Object

### 2.8. ໂຄດຄໍາສັງເພື່ອເຂົ້າເຖິງອີງປະກອບຂອງປະເພດ object: range\_iterator

| ເຮົາສາມາດເຂົ້າເຖິງອີງປະກອບຂອງ range\_iterator ດ້ວຍການນຳໃຊ້ ຜັງຊັ້ນ next.

```
1 type(range(3))
```

```
range
```

```
1 r_iter = iter(range(3)) # converts range type object into an iterator
2 type(r_iter)
```

```
range_iterator
```

```
1 next(r_iter) # you can use next() function because it is an iterator
```

```
0
```

```
1 next(r_iter)
```

```
1
```

```
1 next(r_iter)
```

```
2
```

## 2. Iterator Object

### 2.9. Built-in function for iterable objects : all

- | ឯកឧត្តមេថា ដើម្បីបង្កើតការបែងចែកទៅក្នុងក្រឡូវបាល់ គឺជាអនុញ្ញាតរបស់វា។
- | ក្នុងក្រឡូវបាល់មិនមែនត្រួតពិនិត្យថា មិនមែនមានការបង្កើតការបែងចែកទៅក្នុងក្រឡូវបាល់ទេ។
- | ក្នុងក្រឡូវបាល់មិនមែនត្រួតពិនិត្យថា មិនមែនមានការបង្កើតការបែងចែកទៅក្នុងក្រឡូវបាល់ទេ។

```
1 l1 = [1, 2, 3, 4] # all elements are not 0
2 l2 = [0, 2, 4, 8] # one element is 0
3 l3 = [0, 0, 0, 0] # all elements are 0
```

```
1 all(l1)           # returns True only when all elements are true
```

True

```
1 all(l2)           # returns True only when all elements are true
```

False

```
1 all(l3)           # returns True only when all elements are true
```

False

## 2. Iterator Object

### 2.10. ພັງຊັນໃນຕົວສໍາລັບ iterable objects : any

- | ໂຈມີການຄືນຄ່າເປັນ True ທັງ່າວ່າໃນ iterable object ຢ່າງໜ້ອຍຫົ່ງຕົວເປັນ true.
- | ກໍລະນີລຸ່ມນີ້ ມີແຕ່ l3 ຄືນຄ່າເປັນ False ສໍາລັບພັງຊັນ any(l3).

```
1 l1 = [1, 2, 3, 4] # all elements are not 0
2 l2 = [0, 2, 4, 8] # one element is 0
3 l3 = [0, 0, 0, 0] # all elements are 0
```

```
1 any(l1) # returns True if there is at least one element that is True
```

True

```
1 any(l2) # returns True if there is at least one element that is True
```

True

```
1 any(l3) # returns True if there is at least one element that is True
```

False

## 3. ການປຽບທຽບລະຫວ່າງ List Comprehension ແລະ Lambda Expression

### 3.1. Syntax ຂອງ list comprehension

| syntax ຂອງ list comprehension ແມ່ນສະແດງລຸ່ມນີ້.

```
[ {expression} for {variable} in {iterator/sequence} if {conditional expression} ]
```

| ໃນ list comprehension ບໍ່ຈໍາເປັນຕ້ອງມີ if conditional expression ກໍໄດ້.

```
[ {expression} for {variable} in {iterator/sequence} ]
```

### 3. ການປຽບທຽບລະຫວ່າງ List Comprehension ແລະ Lambda Expression

#### 3.1. Syntax ຂອງ list comprehension

| ການຄິດໄລ່ກໍາລັງສອງຂອງຄ່າໃນ list ໂດຍນຳໃຊ້ ພັງຊັນ map ແລະ lambda

```
1 a = [1, 2, 3, 4, 5, 6, 7]      # list of consecutive values
2 a = list(map(lambda x: x**2, a)) # apply lambda function for each element of the list
3 print(a)
```

[1, 4, 9, 16, 25, 36, 49]

| ການຄິດໄລ່ກໍາລັງສອງຂອງຄ່າໃນ list ໂດຍການນຳໃຊ້ list comprehension

```
1 a = [1, 2, 3, 4, 5, 6, 7]      # list of consecutive values
2 a = [x**2 for x in a]          # apply x**2 for each element of the list
3 print(a)
```

[1, 4, 9, 16, 25, 36, 49]

| ການຄິດໄລ່ກໍາລັງສອງຂອງຄ່າໃນ list ໂດຍການນຳໃຊ້ list comprehension ແລະ range

```
1 a = [x**2 for x in range(1, 8)]
2 print(a)
```

[1, 4, 9, 16, 25, 36, 49]

| ຜົນໄດ້ຮັບຂອງ ໂດດຄໍາສັ່ງທັງສາມແມ່ນຄືກັນ. ໃນ Python ຈະມີຄວາມຫຼາກໝາຍໃນການເຮັດວຽກ.

### 3. ການປຽບທຽບລະຫວ່າງ List Comprehension ແລະ Lambda Expression

#### 3.1. Syntax ຂອງ list comprehension

| ໂຄດຄໍາສັ່ງແມ່ນການຕອງຂຶ້ນແລ້ວໃຫ້ສະແດງຂຶ້ນເປັນ list, ເຊິ່ງການຕອງແມ່ນນຳໃຊ້ພັ້ງຊັນ filter ແລະ ພັ້ງຊັນ lambda.

```
1 ages = [34, 39, 20, 18, 13, 54]
2 adult_ages = list(filter(lambda x: x >= 19, ages))
3 print('adults list:', adult_ages)
```

adult list : [34, 39, 20, 54]

ໂຄດຄໍາສັ່ງທັງສອງແມ່ນໃຊ້ເງື່ອນໄຂການຕອງຄືກັນ ແລະ ເປັນວະລີທີ່ງ່າຍ

| List comprehension of the above code.

```
1 ages = [34, 39, 20, 18, 13, 54]
2 print('adults list:', [x for x in ages if x >= 19])
```

adult list : [34, 39, 20, 54]

| ທັງສອງແບບແມ່ນສິ່ງຜົນໄດ້ຮັບຄືກັນ.

### 3. ການປຽບທຽບລະຫວ່າງ List Comprehension ແລະ Lambda Expression

#### 3.2. List comprehension ແລະ lambda expression ເພື່ອຫຼຸດຄວາມຊັບຊ້ອນຂອງໂຄດຄໍາສັ່ງ

| List comprehension ແລະ lambda expression ແມ່ນມີຄວາມຄ້າຍເລີກນັ້ນໃນການຫຼຸດຄວາມຊັບຊ້ອນຂອງໂຄດຄໍາສັ່ງ.

```
1 [x for x in range(10)]      # list of numbers from 0 to 9
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
1 [x * x for x in range(10)] # square value of numbers from 0 to 9
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

```
1 [x for x in range(10) if x % 2 == 0] # value of even numbers from 0 to 9
```

```
[0, 2, 4, 6, 8]
```

```
1 [x for x in range(10) if x % 2 == 1] # value of odd numbers from 0 to 9
```

```
[1, 3, 5, 7, 9]
```

```
1 [x * x for x in range(10) if x % 2 == 0] # square value of even numbers from 0 to 9
```

```
[0, 4, 16, 36, 64]
```

```
1 [x * x for x in range(10) if x % 2 == 1] # square value of odd numbers from 0 to 9
```

```
[1, 9, 25, 49, 81]
```

### 3. ການປຽບທຽບລະຫວ່າງ List Comprehension ແລະ Lambda Expression

#### 3.2. List comprehension ແລະ lambda expression ເພື່ອຫຼຸດຄວາມຊັບຊ້ອນຂອງໂຄດຄໍາສັ່ງ

 Focus list comprehension ເຮັດໃຫ້ຂຽນໂຄດໃນແກວດຽວ ຕັ້ງສະແດງລຸ່ມນີ້.

```
1 s = input('enter multiple integers :').split()
```

```
enter multiple integers : 10 20 30 40 50
```

```
1 lst = [int(x) for x in s]
2 lst
```

```
[10, 20, 30, 40, 50]
```

```
1 [int(x) for x in input('enter multiple integers : ').split()]
```

```
enter multiple integers : 1 2 3
```

```
[1, 2, 3]
```

| ເຮົາສາມາດແປງຕົວເລກທີ່ມີປະເພດເປັນຂໍ້ຄວາມໃຫ້ເປັນປະເພດຂໍ້ມູນຈໍານວນເຕັມໄດ້ໂດຍ ພັງຊັນ int.

### 3. ການປຽບທຽບລະຫວ່າງ List Comprehension ແລະ Lambda Expression

#### 3.2. List comprehension ແລະ lambda expression ເພື່ອຫຼຸດຄວາມຊັບຊ້ອນຂອງໂຄດຄໍາ

ໂຄດຄໍາສົ່ງລຸ່ມນີ້ເປັນການສ້າງ list ໃໝ່ ໂດຍການຄຸນທັງສອງ list ນີ້. ຕ້ອງໃຊ້ for loop ສອງຕັ້ງ.

```
1 product_xy = []
2 for x in [1, 2, 3]: # computes multiplication of elements of two lists by using double for loop
3     for y in [2, 4, 6]:
4         product_xy.append(x * y)
5 print(product_xy)
```

[2, 4, 6, 4, 8, 12, 6, 12, 18]

| ໂຄດຂ້າງເທິງສາມາດຫຍໍ້ໃຫ້ເປັນແຖວດຽວໂດຍນຳໃຊ້ list comprehension.

```
1 product_xy = [x * y for x in [1, 2, 3] for y in [2, 4, 6]]
2 print(product_xy)
```

[2, 4, 6, 4, 8, 12, 6, 12, 18]

### 3. ការប្រើប្រាស់ List Comprehension និង Lambda Expression

#### 3.3. ការប្រើប្រាស់ list comprehension និង lambda expression

**Ex** ការស្វែនលទ្ធផល 2 និង 3 ដោយការអនុវត្ត list comprehension និង lambda expression

```
1 [n for n in range(1, 31) if n % 2 == 0 if n % 3 == 0]
```

```
[6, 12, 18, 24, 30]
```

```
1 list(filter(lambda x : (x % 2 == 0 and x % 3 == 0), range(1, 31)))
```

```
[6, 12, 18, 24, 30]
```

សមាជិកដំឡើងត្រូវបានខ្ចោះបែកចែកជាអ្នកស្វែនលទ្ធផល 2, 3, 5 ដែលមិនមែនមួយគ្នាលទ្ធផល 1 ទៅ 30

**Ex** ការស្វែនលទ្ធផល 2, 3 និង 5 ដោយការអនុវត្ត list comprehension និង lambda expression

```
1 [n for n in range(1, 31) if n % 2 == 0 if n % 3 == 0 if n % 5 == 0]
```

```
[30]
```

```
1 list(filter(lambda x : (x % 2 == 0 and x % 3 == 0 and x % 5 == 0), range(1, 31)))
```

```
[30]
```

# | Pair programming



## Pair Programming Practice

### | ແນວທາງ, ກິນໄກ ແລະ ແຜນສຸກເສີນ

ການຈັບຄຸ້ຂຽນໂປຣແກຣມ ເປັນການຈັບຄຸ້ຂອງນັກຮຽນເພື່ອຮັດວຽກມອບໝາຍ, ນັກຮຽນຄວນມີແຜນ ແລະ ສາມາດປ່ຽນແທນກັນໄດ້ ໃນ ກໍາລະນີມີຜູ້ໃຫ້ນີ້ບໍ່ສາມາດເຂົ້າຮ່ວມຮັດວຽກມອບໝາຍໄດ້ບໍ່ວ່າໃນກໍາລະນີໃດກຳຕາມ ເຊິ່ງບັນຫາເລື່ອນັ້ນຕ້ອງຮັດໃຫ້ຈະເຈັ້ງ ແລະ ກຳບໍ່ແມ່ນ ຄວາມຜິດຂອງນັກຮຽນທີ່ຈັບຄຸ້ບໍ່ດີ.

### | ຈັບຄຸ້ທີ່ຄ້າຍຄືກັນ, ບໍ່ຈໍາເປັນເຫົ້າທຽມກັນ, ຄວາມສາມາດເປັນຄຸ້ຮ່ວມງານ

ການຈັບຄຸ້ຂຽນໂປຣແກຣມ ຈະໄດ້ຮັບຜົນໃກ່ກໍ່ເນື້ອນັກຮຽນມີຄວາມສາມາດຄ້າຍຄືກັນ ຫາຍວ່າ ບໍ່ຈໍາເປັນຕ້ອງມີຄວາມສາມາດຄືກັນກຳໄດ້, ແຕ່ວ່າ ການຈັບຄຸ້ ນັກຮຽນທີ່ມີຄວາມສາມາດແຕກຕ່າງກັນຫຼາຍ ກໍຈະຮັດໃຫ້ບໍ່ສົມດຸນກັນ. ຄຸສອນຮູ້ດີວ່າ ການຈັບຄຸ້ກັນບໍ່ແມ່ນ ຍຸດທະສາດ “ແບ່ງເພື່ອເອົາຊະນະ” ແຕ່ເປັນຄວາມ ພະຍາຍາມຮັດວຽກຮ່ວມກັນຂອງນັກຮຽນໃຫ້ປະສົບຜົນສໍາເລັດ. ຄຄວນທີ່ກາເວັ້ນການຈັບຄຸ້ກັນລະຫວ່າງນັກຮຽນອ່ອນ ແລະ ນັກຮຽນເກົ່າງ.

### | ກະຕຸ້ນນັກຮຽນໂດຍການໃຫ້ສິ່ງຈຸງໃຈພື້ນເສດ

ຂໍສະເໜີແຮງຈຸງໃຈທີ່ຮັດໃຫ້ນັກຮຽນຈັບຄຸ້, ໂດຍສະເພາະນັກຮຽນທີ່ມີຄວາມສາມາດສຸງ. ບາງຄຸສອນໄດ້ພື້ນວ່າ ການຈັບຄຸ້ຮັດວຽກມອບໝາຍ ແມ່ນມີ ປະໂຫຍດ ສໍາລັບໜຶ່ງ ຫຼື ສອງວຽກມອບເທົ່ານັ້ນ



# Pair Programming Practice

## | ប៉ាងកំណត់របស់ខ្លួន

ສິ່ງທ້າທາຍສໍາລັບຄຸນແມ່ນເພື່ອຊອກຫາວິທີທີ່ຈະປະເມີນຜົນການຮຽນຂອງນັກຮຽນ, ຄຸນຫຼືບໍ່ວ່າ ນັກຮຽນ ໄດ້ຕັ້ງໃຈຮຽນ ຫຼື ບໍ່ຕັ້ງໃຈຮຽນ. ຜູ້ສ່ວວຊານໄດ້ແນະນຳໃຫ້ທົບທວນການອອກແບບຫຼັກສູດການຮຽນ ແລະ ຮູບແບບການປະເມີນ ພ້ອມທັງປີກສາຫາລືຢ່າງຈິງຈັງກັບນັກຮຽນ ກ່ຽວກັບພິດຕິກຳທີ່ຈະບໍ່ຕັ້ງໃຈຮຽນ ນອກຈາກນີ້ຢັ້ງໄດ້ແນະນຳມອບວຽກມອບໝາຍໃຫ້ນັກຮຽນ ພ້ອມທັງອະທິບາຍໃຫ້ເຂົ້າເຈົ້າຢ່າງຈະເຈັ້ງ

| ສະພາບແວດລ້ອມຂອງການຮຽນຮູ້ຮ່ວມກັນ

ສະພາບແວດລ້ອມການຮຽນຮູ້ຮ່ວມກັນເກີດຂຶ້ນໄດ້ທຸກເວລາທີ່ຜູ້ສອນຮຽກຮ້ອງໃຫ້ນັກຮຽນຮັດວຽກຮ່ວມກັນໃນກິດຈະກຳການຮຽນຮູ້ ເຊິ່ງອາດຈະເປັນກິດຈະກຳທີ່ເປັນທາງການ ແລະ ບໍ່ເປັນທາງການ ແລະ ອາດຈະບໍ່ລວມເຖິງການປະເມີນສິນການຮຽນໂດຍກິງ. ເຊັ່ນຕົວຢ່າງ ໃຫ້ນັກຮຽນຈັບຄຸ້ງກັນເພື່ອຮັດວຽກມອບໝາຍ ໂດຍນັກຮຽນຈະຕ້ອງທີ່ບໍ່ທວນກ່ຽວກັບການສອນຂອງອາຈານທີ່ຜ່ານມາ ແລະ ລະດົມແນວຄົດພາຍໃນກຸ່ມ ພ້ອມທັງມືການແບ່ງວຽກໃຫ້ແຕ່ລະຄົມຮັບຜິດຊອບ ຈາກນັ້ນກຳໃຫ້ມີການແລກປ່ຽນຄວາມຄົດເຫັນເຊິ່ງກັນ ແລະ ກັນ ເພື່ອຮັດວຽກມອບໝາຍໃຫ້ສໍາເລັດຕາມເປົ້າໝາຍທີ່ວ່າງໄວ້.

**Q1.** ຄະແນນສອບເສັງຂອງນັກຮຽນມີສາມວິຊາ ຄື English, Math ແລະ Science ປະກອບເປັນສະມາຊິກຂອງ List ເຊິ່ງສາມາດຊຽນໄດ້ດັ່ງນີ້ List = [100, 90, 95] ຕາມລຳດັບ, ຖ້າມີນັກຮຽນສອງຄົນ ຈະສາມາດຊຽນເປັນ List = [100, 90, 95, 90, 85, 93]. ຖ້າມີນັກຮຽນບໍ່ໄດ້ສອບເສັງວິຊາໃດໜຶ່ງກໍຈະມີຄະແນນເທົ່າ 0. ຈຶ່ງພິມຈຳນວນນັກຮຽນທັງໝົດທີ່ມີຄະແນນໃນ List ພ້ອມທັງພິມສະເພາະຈໍານວນນັກຮຽນທີ່ມີຄະແນນສອບເສັງທັງສາມວິຊາ ແລະ ຄະແນນແຕ່ລະວິຊາຂອງແຕ່ລະຄົນທີ່ມີການສອບເສັງ. ດັ່ງສະແດງລຸ່ມນີ້

## Example of Input

```
scores = [100, 90, 95, 90, 80, 70, 0, 80, 90, 90, 0, 90, 100, 75, 20, 30, 50, 90]
```

## Example of Output

```
scores = [100, 90, 95, 90, 80, 70, 0, 80, 90, 90, 0, 90, 100, 75, 20, 30, 50, 90]
```

The number of total students is 6.

The number of students with valid scores is 4.

```
[[100, 90, 95], [90, 80, 70], [100, 75, 20], [30, 50, 90]]
```



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