

SAMSUNG

Samsung Innovation Campus

| Coding, Programming & Data Science

Chapter 2

ພາສາ Python ຂັ້ນພື້ນຖານ - ປະເພດຂໍ້ມູນແບບລຳດັບ

Coding, Programming & Data Science

Chapter Description

● Learning objectives

- ✓ ໃນບົດນີ້ຈະໄດ້ຮຽນຮູ້ການນຳໃຊ້ໂຄງສ້າງຂໍ້ມູນທີ່ມີປະເພດຂໍ້ມູນເປັນແບບລຳດັບເຊັ່ນ: ຂໍ້ມູນແບບ List, ຂໍ້ມູນແບບ Dictionary, ຂໍ້ມູນແບບ tuple, ແລະ ຂໍ້ມູນແບບ Set, ເຊິ່ງສາມາດໃຊ້ໂຄງສ້າງຂໍ້ມູນເຫຼົ່ານີ້ ເພື່ອເພີ່ມທັກສະການຂຽນໂປຣແກຣມໃຫ້ສູງຂຶ້ນ. ໃນບົດນີ້ປະກອບມີ:

● Chapter contents

- ✓ Unit 10. ປະເພດຂໍ້ມູນແບບ List ແລະ ແບບ Tuple
- ✓ Unit 11. ປະເພດຂໍ້ມູນແບບ Dictionary
- ✓ Unit 12. ປະເພດຂໍ້ມູນແບບລຳດັບ
- ✓ Unit 13. 2D Lists
- ✓ Unit 14. Dictionary Method 1
- ✓ Unit 15. Dictionary Method 2
- ✓ Unit 16. ປະເພດຂໍ້ມູນແບບ Set

Unit 10.

ປະເພດຂໍ້ມູນແບບ List ແລະ ແບບ Tuple

● Learning objectives

- ✓ ເຂົ້າໃຈສິ່ງຈຳເປັນ ແລະ ແນວຄິດຂອງປະເພດຂໍ້ມູນແບບ List
- ✓ ເຂົ້າໃຈ ແລະ ໃຊ້ປະໂຫຍດຈາກ List indexing
- ✓ ສາມາດຄຳນວນຄ່າໃຫຍ່ສຸດ ແລະ ນ້ອຍສຸດ ຂອງຂໍ້ມູນແບບ List ດ້ວຍການນຳໃຊ້ຟັງຊັນຊະນິດຕ່າງໆ
- ✓ ເຂົ້າໃຈເຖິງຄວາມແຕກຕ່າງລະຫວ່າງ List ແລະ Tuple ພ້ອມທັງສາມາດເລືອກປະເພດຂໍ້ມູນທີ່ເໝາະສົມໃນການພັດທະນາດ້ວຍເງື່ອນໄຂຕ່າງໆ
- ✓ ສາມາດໃຊ້ tuple ເພື່ອກຳນົດຄ່າໃຫ້ກັບຕົວປ່ຽນຕ່າງໆໃນເວລາດຽວກັນ
- ✓ ສາມາດສະກັດຂໍ້ມູນທີ່ຈຳເປັນໂດຍຜ່ານບັນຊີລາຍການ List ແລະ tuple slicing

Lesson overview

- ✓ ຮຽນຮູ້ຄວາມແຕກຕ່າງລະຫວ່າງໂຄງສ້າງຂໍ້ມູນຂອງແຕ່ລະປະເພດ, ໂດຍປະກອບມີ List, Dictionary ແລະ Tuple
- ✓ ຮຽນຮູ້ລັກສະນະຂອງປະເພດຂໍ້ມູນແບບ List ແລະ ວິທີການຕ່າງໆ
- ✓ ຮຽນຮູ້ສິ່ງຈຳເປັນ ແລະ ການນຳໃຊ້ ປະເພດຂໍ້ມູນແບບ List
- ✓ ຮຽນຮູ້ກ່ຽວກັບຈຸດພິເສດຂອງປະເພດຂໍ້ມູນແບບ Tuple ແລະ List ພ້ອມທັງການປະຍຸກໃຊ້

Concepts You Will Need to Know From Previous Units

- ✓ ຜູ້ຮຽນຕ້ອງເຂົ້າໃຈກ່ຽວກັບຟັງຊັນຂອງຕົວດຳເນີນການຕ່າງໆ ເຊັ່ນ: ຕົວດຳເນີນການ Arithmetic, ຕົວດຳເນີນການ Comparason, ຕົວດຳເນີນການ Logical ແລະ ການນຳໃຊ້ມັນ.
- ✓ ຮູ້ຈັກນຳໃຊ້ຟັງຊັນ for() Loop ແລະ while() Loop
- ✓ ຮູ້ຈັກນຳໃຊ້ຟັງຊັນ input() ແລະ split method ສຳລັບຂໍ້ມູນທີ່ເປັນຂໍ້ຄວາມ

Keywords

List

Indexing

Slicing

in/not in operators

Tuple

| Mission

1. Real world problem

1.1. ຄວາມສໍາຄັນຂອງຂໍ້ມູນມະຫາສານ, ແຕ່ມັນຍາກທີ່ສຸດທີ່ຈະແກ້ໄຂມັນໄດ້



- ຂໍ້ມູນໃນທຸກມື້ນີ້ມັກຈະມີຂະໜາດໃຫຍ່ ແລະ ຊັບຊ້ອນຫຼາຍ, ເຊິ່ງມັນຍາກທີ່ຈະປະມວນຜົນດ້ວຍຊອບແວການຈັດການຂໍ້ມູນ.
- ຂໍ້ມູນຂະໜາດໃຫຍ່ເອີ້ນວ່າ ຂໍ້ມູນມະຫາສານ (big data) ນອກຈາກ big data ມີຂໍ້ມູນມະຫາສານແລ້ວ, ມັນຍັງຖືກສ້າງຂຶ້ນຢ່າງວ່ອງໄວ ແລະ ພົບເຫັນທົ່ວໄປຢ່າງຫຼວງຫຼາຍ, ສະນັ້ນຂໍ້ມູນມະຫາສານຈຶ່ງມີຄວາມສໍາຄັນຫຼາຍ ແລະ ຍາກທີ່ຈະເຮັດວຽກກັບມັນ.
- ເຕັກໂນໂລຊີຂໍ້ມູນມະຫາສານໄດ້ຖືກນໍາໃຊ້ໃນການຕະຫຼາດ ເພື່ອຕິດຕາມປະຫວັດການຊື້ຂອງລູກຄ້າ ແລະ ວິເຄາະສາຍເຫດທີ່ພວກເຂົາຊື້ຜະລິດຕະພັນແຕ່ລະຢ່າງ ທັງນີ້ກໍເພື່ອສ້າງມູນຄ່າເພີ່ມຂອງຜະລິດຕະພັນທີ່ມີຢູ່ ແລະ ທໍາການສົ່ງເສີມການຊື້ຂອງລູກຄ້າ.

1. Real world problem

1.2. ວິເຄາະບັນຫາຂອງຂໍ້ມູນ

A close-up photograph of a white form with green headers. The header 'Personal information' is in a green box. Below it, 'Your details' is also in a green box. The form has several fields: 'First name(s):' with a grid of boxes, 'Date of birth:' with a grid of boxes, and 'Address:' with a grid of boxes. A yellow pencil with a blue eraser is lying diagonally across the 'Date of birth' and 'Address' fields.

- ຖານຂໍ້ມູນຂອງບໍລິສັດ A ມີຂໍ້ມູນປະຫວັດການຊື້ຜະລິດຕະພັນຂອງລູກຄ້າຈາກບໍລິສັດຈຳນວນຫຼາຍ. ສົມມຸດຂໍ້ມູນລູກຄ້າປະກອບມີ: ຊື່, ອາຍຸ, ຄວາມສູງ, ນ້ຳໜັກ ແລະ ຂໍ້ມູນອື່ນໆ.
- ເພື່ອຄວາມສະດວກໃນການເກັບຮັກສາ, ຂໍ້ມູນລູກຄ້າແຕ່ລະຄົນຈະຕ້ອງເກັບເປັນຄ່າດ່ຽວ.
- ເມື່ອມີການເກັບຂໍ້ມູນລູກຄ້າເປັນຄ່າດ່ຽວ, ມັນຈະສະດວກໃນການປ້ອນຂໍ້ມູນ ຊື່, ອາຍຸ, ຄວາມສູງ, ນ້ຳໜັກ ແລະ ຂໍ້ມູນອື່ນໆຂອງລູກຄ້າ.
- ຈາກສິ່ງທີ່ໄດ້ອະທິບາຍຂ້າງເທິງ, ຈະສາມາດວິເຄາະຂໍ້ມູນດ້ວຍໂປຣແກຣມ ໂດຍການປ້ອນຂໍ້ມູນລູກຄ້າທີ່ມີຈຳນວນຫຼາຍຂອງແຕ່ລະຄົນໄດ້.

1. Real world problem

1.3. Spreadsheet?

| Spreadsheet ທຳອິດຂອງໂລກ



The screenshot shows a VisiCalc spreadsheet window titled 'C11 (L) TOTAL'. The spreadsheet has a grid with columns labeled A, B, C, and D. The data is as follows:

	A	B	C	D
1	ITEM	NO.	UNIT	COST
2	MUCK RAKE	4	12.95	55.60
3	BUZZ CUT	15	6.65	101.00
4	TOE TONER	25	49.95	1248.75
5	EYE SNUFF	2	4.95	9.90
			SUBTOTAL	13155.50
			9.75% TAX	1282.66
			TOTAL	14438.16

- ໃນໂປຣແກຣມ MS Office, Spreadsheet ເປັນໂປຣແກຣມທີ່ມີການປະມວນຜົນຂໍ້ມູນໃນຮູບແບບຕາຕະລາງ.
- MS Excel ເປັນຊອບແວທີ່ເປັນ Spreadsheet ໝາຍຄວາມວ່າການເກັບຂໍ້ມູນແມ່ນຢູ່ໃນຮູບແບບຕາຕະລາງ ແລະ ໄດ້ຖືກນຳໃຊ້ຢ່າງກວ້າງຂວາງ.
- Spreadsheet ເປັນໂປຣແກຣມທີ່ເກັບຂໍ້ມູນໃນຮູບແບບຕາຕະລາງ ທຳອິດຂອງໂລກ ເອີ້ນວ່າ VisiCalc ພັດທະນາໂດຍ VisiCorp ໃນປີ ຄສ 1979.
- ໂປຣແກຣມນີ້ ຖືກອອກແບບມາສຳລັບໃຊ້ງານໃນຄອມພິວເຕີ ແລະ ເອີ້ນວ່າ Apple 2

2. ການແກ້ໄຂບັນຫາ

2.1. ຖານຂໍ້ມູນສ່ວນບຸກຄົນ



- ສົມມຸດມີຂໍ້ມູນ 5 ຄົນທີ່ແຕກຕ່າງກັນຄື: ຊື່, ອາຍຸ, ເພດ, ລວງສູງ ແລະ ນ້ຳໜັກ ທີ່ບັນທຶກໄວ້ໃນ ລາຍການ.
- ເມື່ອມີຄົນຊື່ David Doe ອາຍຸ 20 ປີ ເປັນເພດ ຊາຍ, ລວງສູງ 180 ຊັງຕີແມັດ ແລະ ມີນ້ຳໜັກ 100 ກິໂລກຣາມ ສະນັ້ນຂໍ້ມູນຂອງລາວຈະຖືກຈັດເກັບໄວ້ໃນລາຍການ ດັ່ງນີ້
[‘David Doe, 20, 1, 180.0, 100.0].
- ຖ້າອີກຄົນຊື່ Jane Carter ອາຍຸ 22 ປີ, ເພດຍິງ, ມີຄວາມສູງ 169 ຊັງຕີແມັດ ແລະ ມີນ້ຳໜັກ 60 ກິໂລກຣາມ ຈະສາມາດເກັບໄວ້ໃນລາຍການ ດັ່ງນີ້
[‘Jane Carter, 22, 0, 169.0, 60.0]
- ຈາກຂໍ້ມູນຂ້າງເທິງ ສໍາລັບເພດ, ສາມາດແທນ 1 ເປັນເພດຊາຍ ແລະ 0 ເປັນເພດຍິງ
- ເຮົາສາມາດເກັບຂໍ້ມູນໃນຮູບແບບທີ່ໄດ້ກ່າວມາຂ້າງເທິງນີ້ສໍາລັບຄົນອື່ນໆ ທັງນີ້ກໍເພື່ອຄວາມສະດວກໃນການປະມວນຜົນສໍາລັບຂໍ້ມູນມະຫາສານ

2. ການແກ້ໄຂບັນຫາ

2.2. ສ້າງຖານຂໍ້ມູນສ່ວນບຸກຄົນ

- ປ້ອນຂໍ້ມູນລູກຄ້າທັງໝົດ ແລະ ໃຫ້ມີການປະມວນຜົນພ້ອມກັນ.
- ໃນນີ້ໃຫ້ປ້ອນຂໍ້ມູນທັງ 4 ຄົນ ເຂົ້າໄປໃນລາຍການ ທີ່ຊື່ `person_list` ດັ່ງສະແດງຂ້າງລຸ່ມນີ້ ຈາກນັ້ນໃຫ້ໂປຣແກຣມຄິດໄລ່ອາຍຸສະເລ່ຍຂອງຄົນທັງ 4 ແລະ ສະແດງຜົນອອກມາ ໂດຍການນຳໃຊ້ `slicing`. ໂດຍ `Slicing` ຈະຖືກນຳໃຊ້ໃນບົດນີ້ທັງບົດ

```
person1 = ['David Doe', 20, 1, 180.0, 100.0]
person2 = ['John Smith', 25, 1, 170.0, 70.0]
person3 = ['Jane Carter', 22, 0, 169.0, 60.0]
person4 = ['Peter Kelly', 40, 1, 150.0, 50.0]

person_list = person1 + person2 + person3 + person4
```

ຜົນໄດ້ຮັບ

ອາຍຸສະເລ່ຍເທົ່າກັບ 26.75.

3. Mission

3.1. ການເຮັດວຽກຂອງຂໍ້ມູນສ່ວນບຸກຄົນ

```
1 person1 = ['David Doe', 20, 1, 180.0, 100.0]
2 person2 = ['John Smith', 25, 1, 170.0, 70.0]
3 person3 = ['Jane Carter', 22, 0, 169.0, 60.0]
4 person4 = ['Peter Kelly', 40, 1, 150.0, 50.0]
5
6 person_list = person1 + person2 + person3 + person4
7
8 n_persons = int (len(person_list) / 5)
9 age_sum = 0.0
10 for age in person_list[1::5] :
11     age_sum += age
12 average_age = float(age_sum) / n_persons
13 print('the average age is' + str(average_age).)
```

ອາຍຸສະເລ່ຍເທົ່າ 26.75.

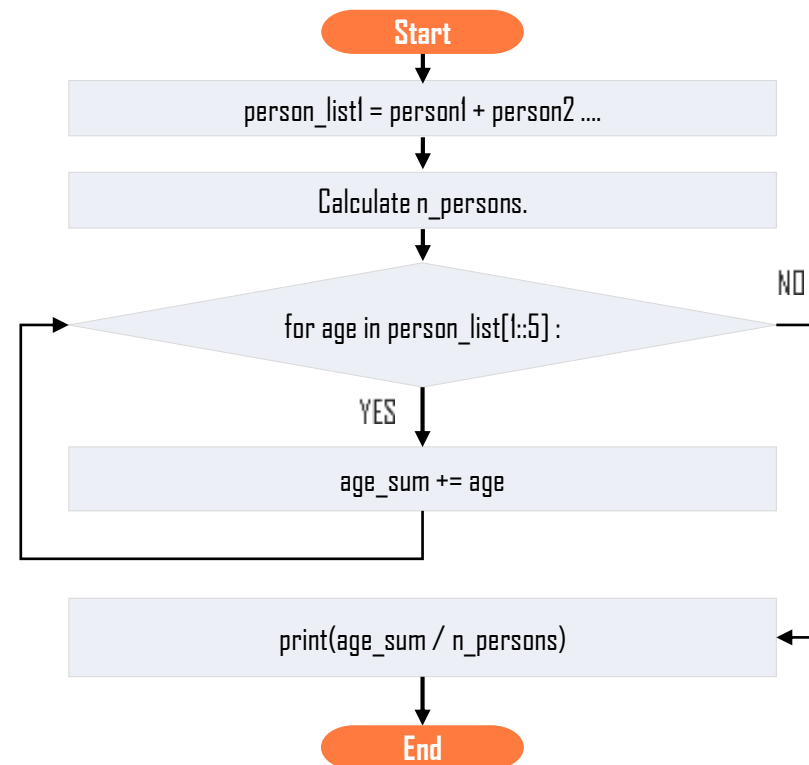
3. Mission

3.2. ການວາງແຜນໂປຣແກຣມ

| Pseudocode

- [1] Start
- [2] Create a list including the name, age, gender (0,1), height, and weight in the personal database.
- [3] Calculate the number of persons in the personal database.
- [4] **Traverse** the "for i in" list and skip the columns with age {
- [5] Add the sum of age. }
- [6] Divide the sum of age with number of persons to print average value
- [7] End

| Flowchart



3. Mission

3.3. ຖານຂໍ້ມູນສ່ວນບຸກຄົນ final code

```
1 person1 = ['David Doe', 20, 1, 180.0, 100.0]
2 person2 = ['John Smith', 25, 1, 170.0, 70.0]
3 person3 = ['Jane Carter', 22, 0, 169.0, 60.0]
4 person4 = ['Peter Kelly', 40, 1, 150.0, 50.0]
5
6 person_list = person1 + person2 + person3 + person4
7
8 n_persons = int (len(person_list) / 5)
9 age_sum = 0.0
10 for age in person_list[1::5] :
11     age_sum += age
12 average_age = float(age_sum) / n_persons
13 print('the average age is' + str(average_age).)
```


| Key concept

1. List ແລະ Dictionary

1.1. ການປະມວນຜົນຂໍ້ມູນທີ່ມີປະສິດທິພາບ

- ຈຸດປະສົງຂອງການນຳໃຊ້ຄອມພິວເຕີກໍຄື ເຮັດໃຫ້ການປະມວນຜົນມີປະສິດທິພາບ. ຖ້າມີການບັນທຶກ ແລະ ປະມວນຜົນຂໍ້ມູນຄະແນນທົດສອບຂອງແຕ່ລະຄົນ. ຄະແນນທົດສອບຈະຖືກເກັບໄວ້ໃນຕົວປ່ຽນ scores.
- ຈາກ code ຄຳສັ່ງລຸ່ມນີ້, ມີຂໍ້ມູນທີ່ເປັນຕົວເລກ 7 ຄ່າ, ສະນັ້ນຈຶ່ງມີການປະກາດຕົວປ່ຽນ 7 ຕົວ ແລະ ກຳນົດຄ່າໃສ່ກັບຕົວປ່ຽນດັ່ງກ່າວໃຫ້ຄົບທຸກຕົວ.

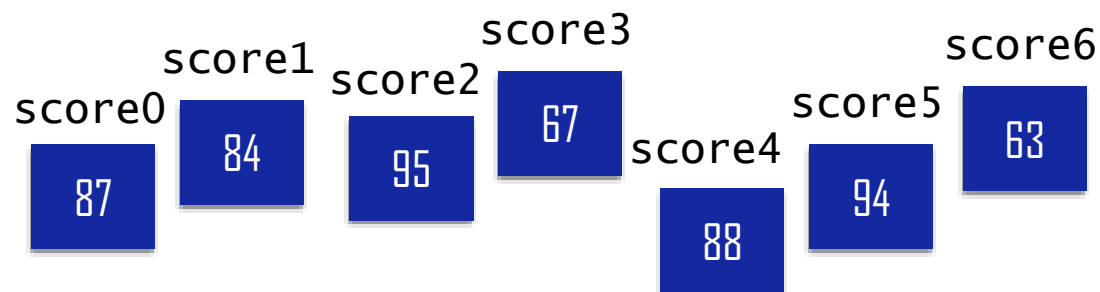
```
1 score0 = 87
2 score1 = 84
3 score2 = 95
4 score3 = 67
5 score4 = 88
6 score5 = 94
7 score6 = 63
8 print(score0, score3)
```

87 67

1. List ແລະ Dictionary

1.2. ສິ່ງສໍາຄັນຂອງຂໍ້ມູນແບບ list

- | ໂຄງສ້າງຂອງໜ່ວຍຄວາມຈໍາປະກອບດ້ວຍຊື່ບັນດາຕົວປ່ຽນ ພ້ອມທັງກໍານົດຄ່າໃຫ້ກັບບັນດາຕົວປ່ຽນເຫຼົ່ານັ້ນ, ດັ່ງນັ້ນ ຖ້າມີຫຼາຍຄ່າ ກໍຕ້ອງມີຫຼາຍຕົວປ່ຽນຕ່າງກັນ.
- | ໂດຍສະເພາະ, ການເພີ່ມ ແລະ/ຫຼື ສະເລ່ຍຕົວປ່ຽນທີ່ແຕກຕ່າງກັນຫຼາຍມັນຈະມີການເຂົ້າລະຫັດທີ່ສັບສົນ ແລະ ເກີດຄວາມຜິດພາດຂຶ້ນ
- | ສະນັ້ນ ຈຶ່ງມີຄວາມຈໍາເປັນຕ້ອງໄດ້ລວມພວກມັນເຂົ້າເປັນປະເພດຂໍ້ມູນດຽວກັນ



1. List ແລະ Dictionary

1.3. Dictionary ແລະ ສິ່ງສໍາຄັນຂອງຂໍ້ມູນແບບ list

- | ພວກເຮົາໄດ້ອະທິບາຍເຖິງຄວາມສໍາຄັນຂອງປະເພດຂໍ້ມູນແບບ list
- | ຂັ້ນຕອນລຸ່ມນີ້ແມ່ນອະທິບາຍຄວາມສໍາຄັນຂອງປະເພດຂໍ້ມູນແບບ dictionary.

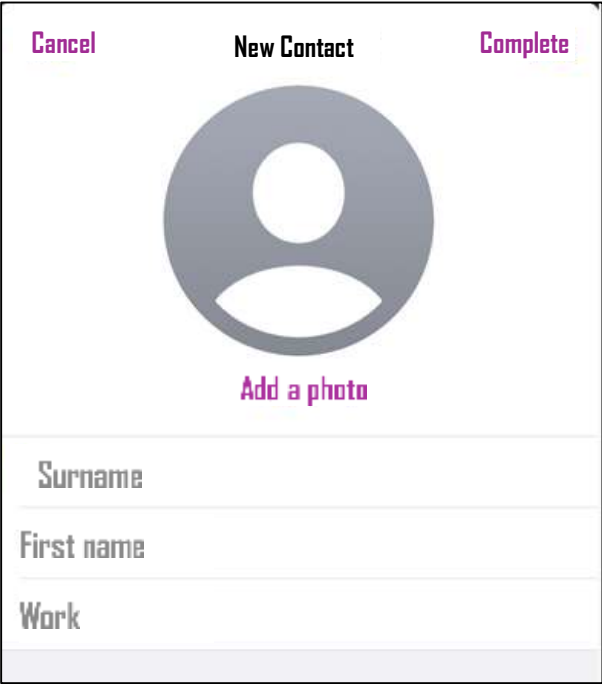
- ▶ ເມື່ອມີການປ້ອນຂໍ້ມູນດັ່ງລຸ່ມນີ້.

```
Surname: Doe  
First name: David  
Work: Company A
```

- ▶ ກົງກັນຂ້າມກັບຂໍ້ມູນແບບ list, dictionary ມີໂຄງສ້າງຂໍ້ມູນແບບຈັບຄູ່ລະຫວ່າງ key : value

1 ການສົມທຽບຂໍ້ມູນໃນ dictionary

- ▶ Dictionary ແມ່ນປະເພດຂໍ້ມູນ ທີ່ມີການຈັບຄູ່ກັນຄື key ແລະ value.
- ▶ ມັນແມ່ນລັກສະນະພິເສດ ດ້ວຍການນຳໃຊ້ key ເພື່ອອ້າງອີງ value.



2. ການປະກາດຕົວປ່ຽນຂໍ້ມູນແບບ List

2.1. list ແມ່ນຫຍັງ?

| List ແມ່ນໂຄງສ້າງຂອງຂໍ້ມູນທີ່ປະກອບດ້ວຍຄ່າຕ່າງໆ.

- ▶ List ໃຊ້ສໍາລັບເກັບຂໍ້ມູນໄວ້ໃນຕົວປ່ຽນດຽວກັນ.
- ▶ ການດໍາເນີນການຕ່າງໆສາມາດເຮັດໄດ້ໃນເວລາດຽວກັນ.

| item ຫຼື element

- ▶ Items ຫຼື elements ແມ່ນອົງປະກອບທີ່ບັນຈຸຢູ່ໃນ list.
- ▶ ຄ່າຂອງຂໍ້ມູນຈະຖືກຂຶ້ນດ້ວຍຈຸດ.
- ▶ ການອ້າງອີງໃນ List ມີຄວາມສໍາຄັນ ໂດຍໃຊ້ index ໃນການອ້າງອີງ.
- ▶ ການປະກາດ list ພ້ອມດ້ວຍອົງປະກອບຂອງມັນ ສາມາດເຮັດໄດ້ດັ່ງລຸ່ມນີ້.

```
1 score_list = [87, 84, 95, 67, 88, 94, 63]
2 score_list
```

[87, 84, 95, 67, 88, 94, 63]

```
1 score_list = [87, 84, 95, 67, 88, 94, 63]
2 print(score_list[0], score_list[3])
```

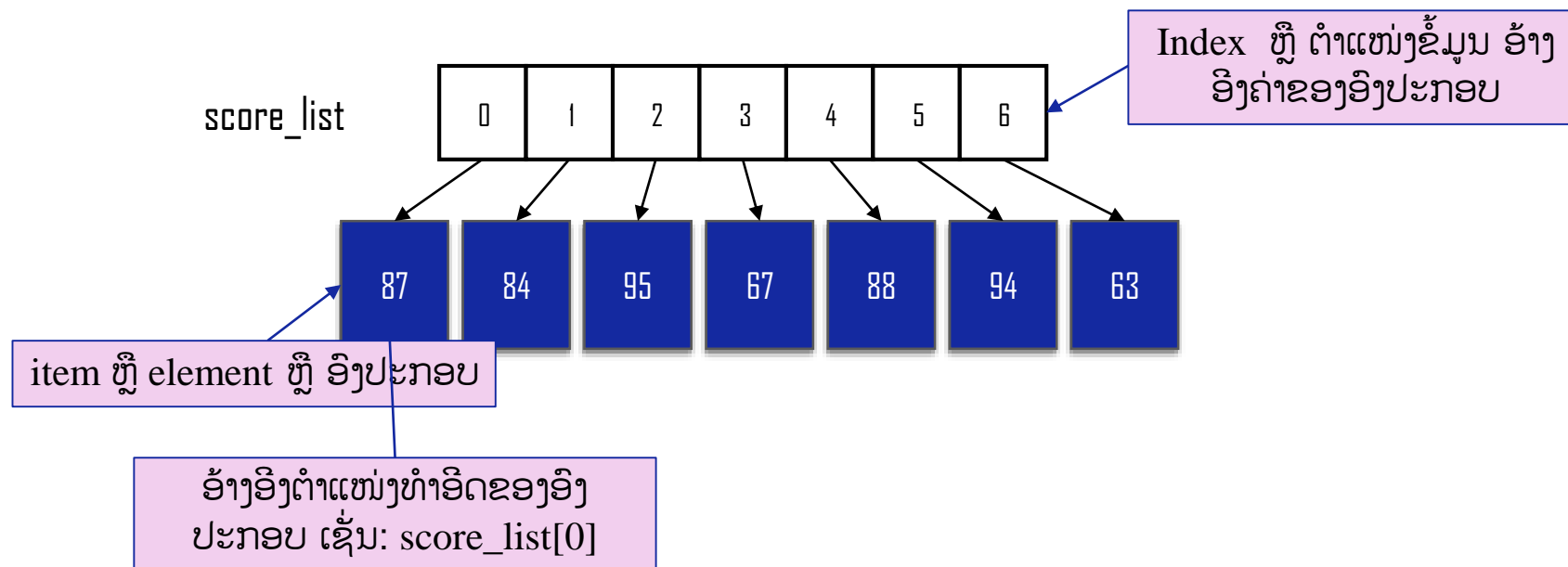
87 67

ອົງປະກອບຂອງ list ສາມາດໃຊ້ຕໍາແໜ່ງຂໍ້ມູນອ້າງອີງໄດ້

2. ການປະກາດຕົວປ່ຽນຂໍ້ມູນແບບ List

2.2. List ແລະ index

- ຈາກຕົວຢ່າງ `score_list`, ເຮົາສາມາດອ້າງອີງໂດຍໃຊ້ຕົວປ່ຽນ `score_list` ແລະ `index` (ຕຳແໜ່ງຂໍ້ມູນ) ດັ່ງສະແດງໃນຮູບລຸ່ມນີ້.
- ຢູ່ໃນ `list`, ອົງປະກອບ ຫຼື ຄ່າ ແມ່ນໃຊ້ເຄື່ອງໝາຍຈຸດ (,) ຂຶ້ນລະຫວ່າງອົງປະກອບຕ່າງໆ.
- ເຮົາສາມາດນຳເອົາອົງປະກອບເຂົ້າໄປ ດ້ວຍການນຳໃຊ້ ຕຳແໜ່ງຂໍ້ມູນເພື່ອກຳນົດທີ່ຕັ້ງໃຫ້ກັບຄ່າທຳອິດໄດ້.



2. ການປະກາດຕົວປ່ຽນຂໍ້ມູນແບບ List

2.2. List ແລະ index

❗ ບໍ່ມີຂໍ້ຈຳກັດໃດທີ່ຖືກນຳມາໃຊ້ໃນປະເພດຂໍ້ມູນແບບ list ໝາຍຄວາມວ່າ ຄ່າຂໍ້ມູນໃນ List ບໍ່ເປັນປະເພດຂໍ້ມູນດຽວກັນກໍໄດ້.

- ▶ ສ້າງ list ທີ່ຊື່ fruits ທີ່ມີອົງປະກອບ 4 ລາຍການຄື: 'banana', 'apple', 'orange', 'kiwi'
- ▶ ຈາກນັ້ນ ສ້າງ list ທີ່ຊື່ mixed_list ທີ່ມີອົງປະກອບ 4 ລາຍການຄື: 100, 200, 'apple', 400 ດັ່ງສະແດງລຸ່ມນີ້

```
1 fruits = ['banana', 'apple', 'orange', 'kiwi'] # List with strings
2 print(fruits)
3 mixed_list = [100, 200, 'apple', 400]
4 print(mixed_list)
```

```
['banana', 'apple', 'orange', 'kiwi']
[100, 200, 'apple', 400]
```

Python lists ສາມາດມີປະເພດຂໍ້ມູນທີ່ແຕກຕ່າງກັນໄດ້ ເຊັ່ນມີທັງ ຈຳນວນຖ້ວນ, ຈຳນວນຈິງ, ຂໍ້ຄວາມ ແລະ ອື່ນໆ

2. ການປະກາດຕົວປ່ຽນຂໍ້ມູນແບບ List

2.3. ຟັງຊັນ range

- | ສ້າງ list ລາຍການໄດ້ ດ້ວຍການນຳໃຊ້ຟັງຊັນ range.
- | ຟັງຊັນ range ເປັນຟັງຊັນທີ່ໃຫ້ຄ່າເປັນລຳດັບ ເຊັ່ນ ຖ້າສ້າງ range(1, 10) ມັນຈະສົ່ງອົງປະກອບ ຫຼື ຄ່າ ຈາກ 1 ເຖິງ 9 ອອກມາ.

```
1 list4 = list(range(1, 10))  
2 list4
```

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

ນຳໃຊ້ຟັງຊັນ range ໃນ Python ເພື່ອສ້າງ array ແບບຕໍ່ເນື່ອງ.

3. ຕຳແໜ່ງຂໍ້ມູນ ແລະ ການຄິດໄລ່ຄວາມຍາວຂອງ List

3.1. ການນິຍາມ: List, index, indexing

▮ ລຸ່ມນີ້ແມ່ນ ການນິຍາມ List, Index, Indexing.

► List

- ເປັນໂຄງສ້າງຂໍ້ມູນໃນພາສາ Python ທີ່ສາມາດບັນຈຸໄດ້ຫຼາຍອົງປະກອບ ຫຼື ຫຼາຍຄ່າທີ່ແຕກຕ່າງກັນ ແລະ ສາມາດປ່ຽນແທນຄ່າເຫຼົ່ານັ້ນໄດ້.
- ສາມາດເລືອກເອົາອົງປະກອບ ຫຼື ຄ່າທີ່ຈຳເປັນຈາກ ອົງປະກອບໃນ list ໄດ້.

► index

- ອ້າງອີງຕົວເລກຕຳແໜ່ງຂໍ້ມູນເພື່ອຊີ້ໃຫ້ເຫັນອົງປະກອບ ຫຼື ຄ່າ ໃນ list.
- ຕຳແໜ່ງຂໍ້ມູນໃນ list ຈະເລີ່ມຈາກ 0 ເຖິງ $n-1$, ເຊິ່ງ list ມີ n ອົງປະກອບ
- ສາມາດນຳໃຊ້ ຕຳແໜ່ງຂໍ້ມູນທີ່ມີຄ່າລົບ (negative index)ໄດ້ .

► indexing

- ແມ່ນການເຂົ້າຫາຄ່າຂອງອົງປະກອບດ້ວຍຕຳແໜ່ງຂໍ້ມູນ.

3. ຕຳແໜ່ງຂໍ້ມູນ ແລະ ການຄິດໄລ່ຄວາມຍາວຂອງ List

3.2. ຕຳແໜ່ງຂໍ້ມູນ ແລະ ການຄິດໄລ່ຄວາມຍາວຂອງ list

ໃນ list, ຕຳແໜ່ງຂໍ້ມູນຂອງອົງປະກອບທຳອິດແມ່ນ 0 ສ່ວນອົງປະກອບສຸດທ້າຍແມ່ນ n-1.

ນຳໃຊ້ຟັງຊັນ len ເພື່ອຄິດໄລ່ຈຳນວນອົງປະກອບ ຫຼື ຄວາມຍາວຂອງ list, ດັ່ງສະແດງໃນ Code ຄຳສັ່ງລຸ່ມນີ້.

```
1 n_list = [11, 22, 33, 44, 55, 66]
2 print(n_list)
3 print(len(n_list))
```

```
[11, 22, 33, 44, 55, 66]
6
```

index	[0]	[1]	[2]	[3]	[4]	[5]
n_list =	11	22	33	44	55	66

ຄວາມຍາວຂອງ n_list ແມ່ນ 6, ແຕ່ການເຂົ້າເຖິງຕຳແໜ່ງຂອງຂໍ້ມູນ ແມ່ນ 0 ~ 5.

```
1 n_list[0]      # The index of the first item of list
11              is 0.
```

```
1 n_list[1]      # The index of the second item of list
22              is 1.
```

```
n_list[0] =
11
n_list[1] =
22
n_list[2] =
33
n_list[3] =
44
n_list[4] =
55
n_list[5] =
66
```

3. ຕຳແໜ່ງຂໍ້ມູນ ແລະ ການຄິດໄລ່ຄວາມຍາວຂອງ List

3.3. ຂໍ້ຄວນລະວັງຂອງຕຳແໜ່ງຂໍ້ມູນໃນ list

! ບໍ່ຄວນກຳນົດຕຳແໜ່ງຂໍ້ມູນເກີນຕຳແໜ່ງທີ່ມີຢູ່ ດັ່ງຕົວຢ່າງລຸ່ມນີ້

```
1 n_list = [11, 22, 33, 44, 55, 66] # Index with 6 elements
2 n_list[5]                        # The last element value of the list
```

66

```
1 n_list[6]                        # The 7th element value of the index does not
                                exist
```

```
-----
IndexError                                Traceback (most recent call last)
<ipython-input-16-856e6c489946> in <module>
----> 1 n_list[6]                        # The 7th element value of the index does not
                                exist

IndexError: list index out of range
```

- ▶ ບໍ່ຄວນໃສ່ຕຳແໜ່ງຂໍ້ມູນຫຼາຍກວ່າຕຳແໜ່ງສູງສຸດຂອງຂໍ້ມູນ.
- ▶ ຕຳແໜ່ງສູງສຸດຂອງຂໍ້ມູນແມ່ນ `len(n_list)-1`.
- ▶ ເມື່ອການເຂົ້າເຖິງຕຳແໜ່ງຂໍ້ມູນມີບັນຫາ, ຈະເກີດຄວາມຜິດພາດຂຶ້ນດັ່ງນີ້: `IndexError: list index out of range`

3. ຕຳແໜ່ງຂໍ້ມູນ ແລະ ການຄິດໄລ່ຄວາມຍາວຂອງ List

3.4. ຕຳແໜ່ງຂໍ້ມູນທີ່ເປັນຄ່າລົບ (Negative index)

❗ ລຸ່ມນີ້ແມ່ນຕົວຢ່າງການເຂົ້າເຖິງຂໍ້ມູນດ້ວຍຕຳແໜ່ງຂໍ້ມູນທີ່ເປັນຄ່າລົບ.

```
1 n_list = [11, 22, 33, 44, 55, 66]  
2 n_list[-1]
```

66

```
1 n_list[-2]
```

55

```
1 n_list[-3]
```

44

 Line 2

- ຄ່າຂໍ້ມູນສຸດທ້າຍຂອງ list ສາມາດເຂົ້າເຖິງດ້ວຍຕຳແໜ່ງຂໍ້ມູນ -1.

3. ຕຳແໜ່ງຂໍ້ມູນ ແລະ ການຄິດໄລ່ຄວາມຍາວຂອງ List

3.4. ຕຳແໜ່ງຂໍ້ມູນທີ່ເປັນຄ່າລົບ (Negative index)

- | ໃນຟັງຊັນ `len(n_list)` ຈະສົ່ງຄວາມຍາວ ຫຼື ຈຳນວນອົງປະກອບຂອງ list ອອກມາ.
- | ອົງປະກອບສຸດທ້າຍໃນ list ແມ່ນ `[-1 + len(n_list)]`.
- | ອົງປະກອບທຳອິດແມ່ນຢູ່ໃນຕຳແໜ່ງຂໍ້ມູນທີ `[-len(n_list) + len(n_list)]=[0]`, ໝາຍວ່າ `len(n_list) = 6`, ສະນັ້ນ `[-6 + 6] = [0]`.
- | ການອ້າງອີງຂອງແຕ່ລະອົງປະກອບສາມາດນຳໃຊ້ຕຳແໜ່ງຂໍ້ມູນທີ່ເປັນຄ່າລົບໄດ້.
- | ສຳລັບຕຳແໜ່ງຂໍ້ມູນທີ່ເປັນຄ່າລົບ ຈະຫຼຸດລົງເທື່ອລະ -1 ເຊິ່ງເລີ່ມຈາກຕຳແໜ່ງຂໍ້ມູນທີ່ມີອົງປະກອບຕົວສຸດທ້າຍ ເຊັ່ນ: -1, -2, -3.

`n_list[-1] = 66`

`n_list[-2] = 55`

`n_list[-3] = 44`

`n_list[-4] = 33`

`n_list[-5] = 23`

`n_list[-6] = 11`

`n_list =`
Negative index

11	22	33	44	55	66
[-6]	[-5]	[-4]	[-3]	[-2]	[-1]

4. ການເພີ່ມ, ການລຶບ ອົງປະກອບໃນ List

4.1. ການນຳໃຊ້ເຄື່ອງໝາຍ + ໃນ List

- ນຳໃຊ້ເຄື່ອງໝາຍ + ເພື່ອບວກອົງປະກອບຂອງແຕ່ລະ list ເຂົ້າກັນ.
- ຈາກຕົວຢ່າງລຸ່ມນີ້, ບັນດາອົງປະກອບທັງໝົດຂອງ list ທີ່ຊື່ person1 ແລະ person2 ຈະຖືກລວມເຂົ້າໄປໃນ list ດຽວກັນ ທີ່ຊື່ person_list.

```
1 person1 = ['David Doe', 20, 1, 180.0, 100.0]
2 person2 = ['John Smith', 25, 1, 170.0, 70.0]
3
4 person_list = person1 + person2 # Add items of two lists to make a single
5 print(person_list)              list
```

```
['David Doe', 20, 1, 180.0, 100.0, 'John Smith', 25, 1, 170.0, 70.0]
```

Key Concept

4. ການເພີ່ມ, ການລຶບ ອົງປະກອບໃນ List

4.2. ການນຳໃຊ້ append method ເພື່ອເພີ່ມອົງປະກອບໃນ list

- | ເຮົາສາມາດເພີ່ມ ແລະ ລຶບ ບັນດາອົງປະກອບໃນ list ໄດ້.
- | append method ເປັນ method ທີ່ສາມາດເພີ່ມອົງປະກອບໃໝ່ເຂົ້າໄປໃນ list, ໂດຍອົງປະກອບ ທີ່ເພີ່ມເຂົ້າມາໃໝ່ຈະຢູ່ຕໍ່ຈາກອົງປະກອບສຸດທ້າຍໃນ list.
- | method ເປັນ built-in function ຂອງ list ແລະ ສາມາດເອີ້ນໃຊ້ດ້ວຍການນຳໃຊ້ເຄື່ອງໝາຍ . (dot)
- | ສຳລັບລາຍລະອຽດກ່ຽວກັບ method ຈະອະທິບາຍໃນ Unit ຕໍ່ໄປ.

```
1 a_list = ['a', 'b', 'c', 'd', 'e']
2 a_list.append('f')    # Add 'f'
3 a_list
```

```
['a', 'b', 'c', 'd', 'e', 'f']
```

```
1 n_list = [10, 20, 30, 40]
2 n_list.append(50)    # Add 50
3 n_list
```

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

4. ການເພີ່ມ, ການລຶບ ອົງປະກອບໃນ List

4.3. ການນຳໃຊ້ extend method ເພື່ອເພີ່ມອົງປະກອບໃນ list

| extend method: ເພີ່ມ list ຫຼື ອົງປະກອບເຂົ້າໄປຕໍ່ທ້າຍໃນ list.

| ຈາກຕົວຢ່າງລຸ່ມນີ້ ແມ່ນການໃຊ້ extend method ເພື່ອເພີ່ມບັນດາອົງປະກອບໃນ list2 ເຂົ້າໄປໃນ list1 ແລະ ໄດ້ຜິນຮັບ ['a', 'b', 'c', 1, 2, 3].

| ຈາກນັ້ນ ແຊກອົງປະກອບໃໝ່ເຊັ່ນ 'd' ເຂົ້າໄປໃນ list1 ຈະໄດ້ຜິນຮັບ ດັ່ງລຸ່ມນີ້.

```
1 list1 = ['a', 'b', 'c']
2 list2 = [1, 2, 3]
3 list1.extend(list2)
4 list1
```

['a', 'b', 'c', 1, 2, 3]

```
1 list1.extend('d')
2 list1
```

['a', 'b', 'c', 1, 2, 3, 'd']

4. ການເພີ່ມ, ການລຶບ ອົງປະກອບໃນ List

4.3. ການນຳໃຊ້ extend method ເພື່ອເພີ່ມອົງປະກອບໃນ list

- ຖ້ານຳໃຊ້ append method ດັ່ງສະແດງໃນ Code ຄຳສັ່ງລຸ່ມນີ້, ຜົນໄດ້ຮັບຈະບໍ່ເປັນແບບນີ້ [11, 22, 33, 44, 55, 66]
- ໝາຍວ່າ ຖ້າເອີ້ນໃຊ້ append method ເຊັ່ນ list1.append([55, 66]) , ອົງປະກອບ [55, 66] ຈະໄປຢູ່ດ້ານຫຼັງຂອງ [11, 22, 33, 44], ສະນັ້ນ list1 ຈະສະແດງ [11, 22, 33, 44, [55, 66]] ອອກມາ.

```
1 list1 = [11, 22, 33, 44]
2 list1.append([55, 66])
3 list1
```

[11, 22, 33, 44, [55, 66]]

- ແຕ່, ຖ້າຕ້ອງການເພີ່ມອົງປະກອບໃໝ່ເຂົ້າໄປໃນ list , ໃຫ້ໄດ້ຜົນອອກມາເປັນ [11, 22, 33, 44, 55, 66] ຕ້ອງໃຊ້ extend method ດັ່ງ Code ຄຳສັ່ງລຸ່ມນີ້
- ທົດລອງດ້ວຍຕົວເອງຕາມ Code ຄຳສັ່ງລຸ່ມນີ້.

```
1 list1 = [11, 22, 33, 44]
2 list1.extend([55, 66])
3 list1
```

[11, 22, 33, 44, 55, 66]

4. ການເພີ່ມ, ການລຶບ ອົງປະກອບໃນ List

4.4. Methods ສຳລັບລຶບຂໍ້ມູນໃນ List

- | ໃນການລຶບອົງປະກອບ ຫຼື ຂໍ້ມູນໃນ List ສາມາດເຮັດໄດ້ຫຼາຍແບບດັ່ງນີ້:
- | ໃຊ້ຄຳສັ່ງ del ໃນ Python
- | ໃຊ້ remove method ໃນ list class
- | ໃຊ້ pop method
 - ▶ ເວລານຳໃຊ້ pop method, ມັນຈະລຶບອົງປະກອບໃນຕຳແໜ່ງທີ່ແນ່ນອນຂອງ list ແລະ ຈະສົ່ງຄ່າອົງປະກອບນັ້ນ.

4. ການເພີ່ມ, ການລຶບ ອົງປະກອບໃນ List

4.5. ການລຶບດ້ວຍຄໍາສັ່ງ del

- ❑ ລຶບອົງປະກອບຈະຕ້ອງລຶບດ້ວຍຕຳແໜ່ງຂໍ້ມູນ ເທົ່ານັ້ນ
- ❑ ຈະບໍ່ສາມາດລຶບອົງປະກອບໂດຍກົງ ເຊັ່ນ `del 44` ດັ່ງສະແດງລຸ່ມນີ້

```
1 n_list = [11, 22, 33, 44, 55, 66]
2 print(n_list) # Print the entire
3               list
4 del n_list[3] # Delete 44
5 print(n_list)
```

[11, 22, 33, 44, 55, 66]

[11, 22, 33, 55, 66]

🔍 Line 4

- ນຳໃຊ້ຕຳແໜ່ງຂໍ້ມູນເຊັ່ນ: `n_list[3]` ເຊິ່ງເປັນຕຳໜ່ງທີ 3 ຂອງອົງປະກອບໃນ list ທີ່ມີຄ່າເທົ່າ 44.

4. ການເພີ່ມ, ການລຶບ ອົງປະກອບໃນ List

4.6. ການລຶບດ້ວຍ pop method

pop method ຈະລຶບອົງປະກອບສຸດທ້າຍຂອງ list.

```
1 n_list = [10, 20, 30]
2 print(n_list) # print the entire items
```

[10, 20, 30]

```
1 n = n_list.pop()
2 print('n =', n)
3 print('n_list =', n_list)
```

n = 30
n_list = [10, 20]

n_list = [10, 20, 30] :

n_list before pop: 10 20 30

after n =
n_list.pop:

n_list after pop: 10 20
n after pop: 30

4. ການເພີ່ມ, ການລຶບ ອົງປະກອບໃນ List

4.7. ການລຶບດ້ວຍ remove method

- ເປັນການລຶບສະເພາະອົງປະກອບໃນ list ດ້ວຍການນຳໃຊ້ remove method .
- ຖ້າຕ້ອງການລຶບຄ່າຂໍ້ມູນ 44 ໃນ list ດັ່ງຕົວຢ່າງລຸ່ມນີ້ ຕ້ອງໃຊ້ method ເຊັ່ນ remove(44).

```
1 n_list = [11, 22, 33, 44, 55, 66]
2 print(n_list)
3
4 n_list.remove(44)
5 print(n_list)
```

[11, 22, 33, 44, 55, 66]

[11, 22, 33, 55, 66]



Line 4

- ໃຊ້ remove method ໃນ n_list ເພື່ອລຶບອົງປະກອບທີ່ມີຄ່າເທົ່າ 44.

4. ການເພີ່ມ, ການລຶບ ອົງປະກອບໃນ List

4.8. ຂໍ້ຄວນລະວັງ ໃນການນຳໃຊ້ remove method

ບັນຫາຂອງການໃຊ້ remove method

```
1 n_list = [11, 22, 33, 44, 55, 66]
2 print(n_list)
3
4 n_list.remove(88)
5 print(n_list)
```

[11, 22, 33, 44, 55, 66]

```
-----
ValueError                                Traceback (most recent call last)
<ipython-input-46-e5b8ba8b9713> in <module>
      2 print(n_list)
      3
----> 4 n_list.remove(88)
      5 print(n_list)
```

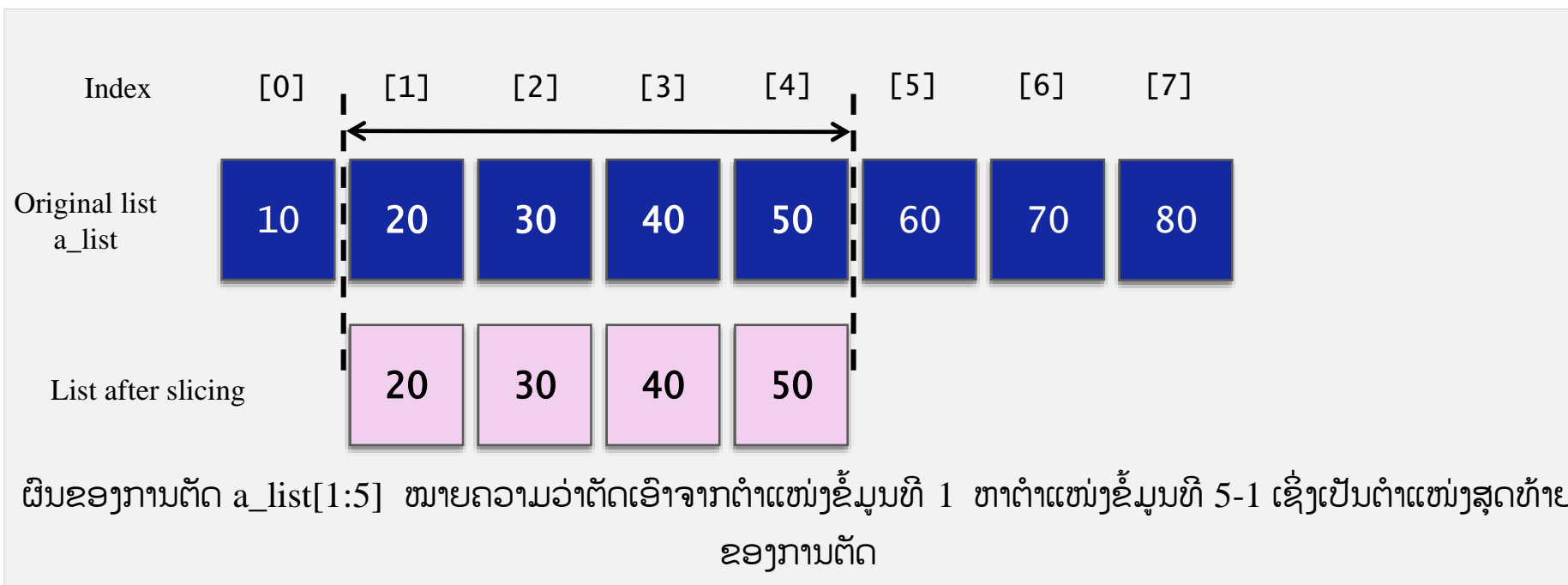
ValueError: list.remove(x): x not in list

- ▶ ມັນຈະເກີດ error ເວລາລຶບຂໍ້ມູນທີ່ບໍ່ມີໃນ list.
- ▶ ບໍ່ຄວນປ້ອນຄ່າຂໍ້ມູນທີ່ບໍ່ມີໃນ list ເພື່ອລຶບ.

5. ການຕັດໃນ List

5.1. ການຕັດ

- | ການຕັດ ແມ່ນການຕັດເອົາສະເພາະອົງປະກອບບາງສ່ວນໃນ list
- | ໂດຍນຳໃຊ້ຊື່ຂອງ list ຕາມດ້ວຍ [start : end] ເພື່ອຕັດເອົາອົງປະກອບຈາກຕຳແໜ່ງເລີ່ມຕົ້ນ ໄປຫາຕຳແໜ່ງສຸດທ້າຍ, ແຕ່ຕຳແໜ່ງສຸດທ້າຍແມ່ນ end-1.



5. ການຕັດໃນ List

5.2. ຕົວຢ່າງການຕັດ

❗ ລຸ່ມນີ້ແມ່ນຕົວຢ່າງການຕັດໃນ List.

```
1 a_list = [10, 20, 30, 40, 50, 60, 70, 80]  
2 a_list[1:5]
```

[20, 30, 40, 50]

```
1 a_list[0:5]
```

[10, 20, 30, 40, 50]

```
1 a_list[1:]
```

[20, 30, 40, 50, 60, 70, 80]

```
1 a_list[:5]
```

[10, 20, 30, 40, 50]

```
1 a_list[:]
```

[10, 20, 30, 40, 50, 60, 70, 80]

 Line 2

- ຕັດຕໍາແໜ່ງຂໍ້ມູນທີ 1 ຄືຄ່າ 20 ໄປຫາຕໍາແໜ່ງຂໍ້ມູນທີ 4 ມີຄ່າເທົ່າ 50 ເນື່ອງຈາກຕໍາແໜ່ງສຸດທ້າຍແມ່ນ $5-1=4$
- ຜົນໄດ້ຮັບກໍຄື 20, 30, 40, 50.

5. ການຕັດໃນ List

5.2. ຕົວຢ່າງການຕັດ

```
1 a_list = [10, 20, 30, 40, 50, 60, 70, 80]  
2 a_list[1:5]
```

[20, 30, 40, 50]

```
1 a_list[0:5]
```

[10, 20, 30, 40, 50]

```
1 a_list[1:]
```

[20, 30, 40, 50, 60, 70, 80]

```
1 a_list[:5]
```

[10, 20, 30, 40, 50]

```
1 a_list[:]
```

[10, 20, 30, 40, 50, 60, 70, 80]



Line 1

- ຕັດຈາກຕຳແໜ່ງທີ 1 ທີ່ມີຄ່າເທົ່າ 10 ຕຳແໜ່ງ 4 ທີ່ມີຄ່າເທົ່າ 50
- ຈະໄດ້ຜົນຮັບດັ່ງນີ້ 10, 20, 30, 40, 50.

5. ການຕັດໃນ List

5.2. ຕົວຢ່າງການຕັດ

```
1 a_list = [10, 20, 30, 40, 50, 60, 70, 80]  
2 a_list[1:5]
```

[20, 30, 40, 50]

```
1 a_list[0:5]
```

[10, 20, 30, 40, 50]

```
1 a_list[1:]
```

[20, 30, 40, 50, 60, 70, 80]

```
1 a_list[:5]
```

[10, 20, 30, 40, 50]

```
1 a_list[:]
```

[10, 20, 30, 40, 50, 60, 70, 80]



Line 1

- ຕັດຈາກອົງປະກອບທີ 2 ຄື 20 ຈົນຮອດອົງປະກອບສຸດທ້າຍ.
- ເຮົາສາມາດຕັດຈົນຮອດອົງປະກອບສຸດທ້າຍ.

5. ການຕັດໃນ List

5.2. ຕົວຢ່າງການຕັດ

```
1 a_list = [10, 20, 30, 40, 50, 60, 70, 80]  
2 a_list[1:5]
```

[20, 30, 40, 50]

```
1 a_list[0:5]
```

[10, 20, 30, 40, 50]

```
1 a_list[1:]
```

[20, 30, 40, 50, 60, 70, 80]

```
1 a_list[:5]
```

[10, 20, 30, 40, 50]

```
1 a_list[:]
```

[10, 20, 30, 40, 50, 60, 70, 80]

 Line 1

- ຕັດຈາກອົງປະກອບທີ 1 ຄື 10 ເຖິງອົງປະກອບທີ 5 ຄື 50.
- ສາມາດຂ້າມຕຳແໜ່ງອົງປະກອບທຳອິດຂອງຂໍ້ມູນໄດ້.

5. ການຕັດໃນ List

5.2. ຕົວຢ່າງການຕັດ

```
1 a_list = [10, 20, 30, 40, 50, 60, 70, 80]
2 a_list[1:5]
```

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

```
1 a_list[0:5]
```

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

```
1 a_list[1:]
```

```
[20, 30, 40, 50, 60, 70, 80]
```

```
1 a_list[:5]
```

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

```
1 a_list[:]
```

```
[10, 20, 30, 40, 50, 60, 70, 80]
```

Line 1

- ຖ້າບໍ່ມີການກຳນົດຕຳແໜ່ງ ກໍຈະຕັດເອົາທັງໝົດ ໝາຍວ່າຈະເອົາຕັ້ງແຕ່ຕຳແໜ່ງເລີ່ມຕົ້ນ ຈົນຮອດຕຳແໜ່ງສຸດທ້າຍຂອງຂໍ້ມູນ.
- ເຮົາສາມາດຕັດເອົາຕຳແໜ່ງເລີ່ມຕົ້ນຈົນຮອດຕຳແໜ່ງສຸດທ້າຍດ້ວຍການກຳນົດດັ່ງນີ້ [:].

5. ການຕັດໃນ List

5.3. ສັງລວມການຕັດໃນ list

ໃນ Python ການຕັດ ສາມາດໃຊ້ໄດ້ທັງຕຳແໜ່ງຂໍ້ມູນທີ່ເປັນຄ່າລົບ ແລະ step ທີ່ເປັນຄ່າລົບ.

Syntax	Function
a_list[start:end]	ຕັດ item ຈາກຕຳແໜ່ງເລີ່ມຕົ້ນຮອດຕຳແໜ່ງສຸດທ້າຍລົບໜຶ່ງ ໝາຍວ່າ end-1
a_list[start:]	ຕັດ item ຈາກຕຳແໜ່ງເລີ່ມຕົ້ນຫາຕຳແໜ່ງສຸດທ້າຍ.
a_list[:end]	ຕັດຈາກຕຳແໜ່ງເລີ່ມຕົ້ນຫາຕຳແໜ່ງ end-1
a_list[:]	ຕັດເອົາໝົດ list
a_list[start:end:step]	ຕັດເອົາຕຳແໜ່ງດ້ວຍການກະໂດດໄປຕາມ step ຈາກຕຳແໜ່ງເລີ່ມຕົ້ນຫາຕຳແໜ່ງ end-1
a_list[-2:]	ຕັດເອົາ 2 items ຈາກຕຳແໜ່ງສຸດທ້າຍ
a_list[:-2]	ຕັດເອົາທຸກ items ຍົກເວັ້ນ 2 ຕຳແໜ່ງສຸດທ້າຍ (2 item ສຸດທ້າຍ)
a_list[::-1]	ຕັດເອົາທຸກ items ໃນ list ແຕ່ຈະລຽງລຳດັບຂອງ items ກົງກັນຂ້າມກັນກັບຕົວເດີມ
a_list[1::-1]	ຕັດເອົາ 2 items ທຳອິດ ແລະ ລຽງລຳດັບກົງກັນຂ້າມ

5. ການຕັດໃນ List

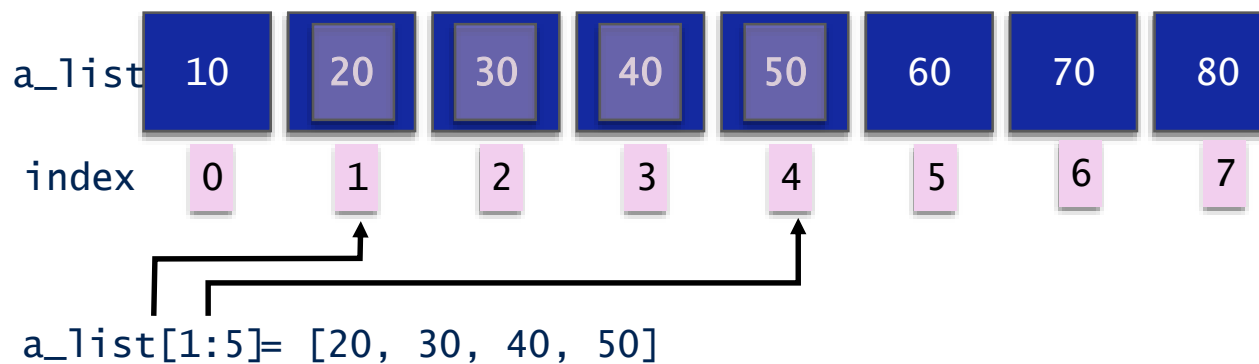
5.4. ການຕັດຕໍາແໜ່ງຂໍ້ມູນ ຕັ້ງແຕ່ຕໍາແໜ່ງເລີ່ມຕົ້ນຈົນຮອດຕໍາແໜ່ງສຸດທ້າຍ

| `a_list[1:5]`: ໝາຍເຖິງຕັດເອົາຈາກຕໍາແໜ່ງ `a_list[1]` ຫາ `a_list[5-1]`.

```
1 a_list = [10, 20, 30, 40, 50, 60, 70, 80]
```

```
2 a_list[1:5]
```

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



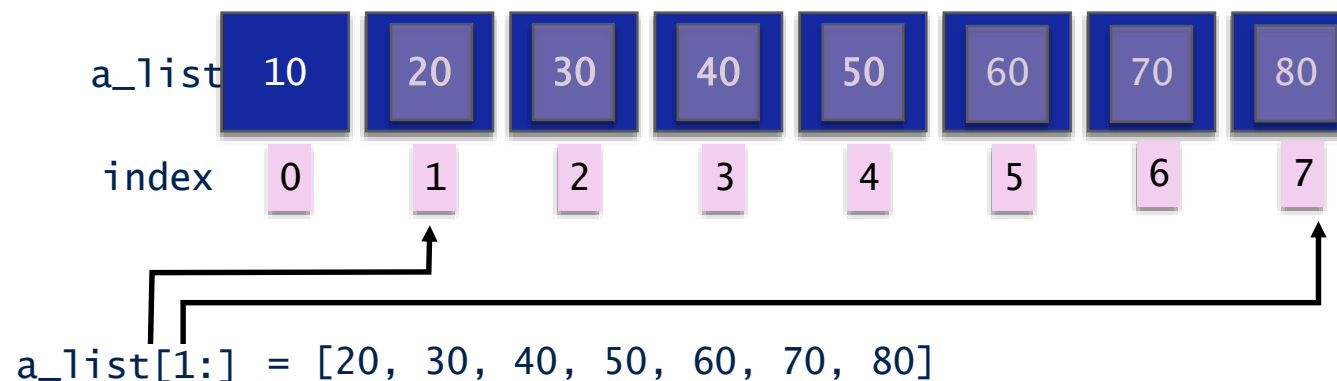
5. ການຕັດໃນ List

5.5. ບໍ່ເອົາຕຳແໜ່ງທຳອິດ

| `a_list[1:]`: ໝາຍເຖິງເອົາທຸກອົງປະກອບຂອງ list ຍົກເວັ້ນອົງປະກອບທຳອິດ

```
1 a_list[1:]
```

```
[20, 30, 40, 50, 60, 70, 80]
```



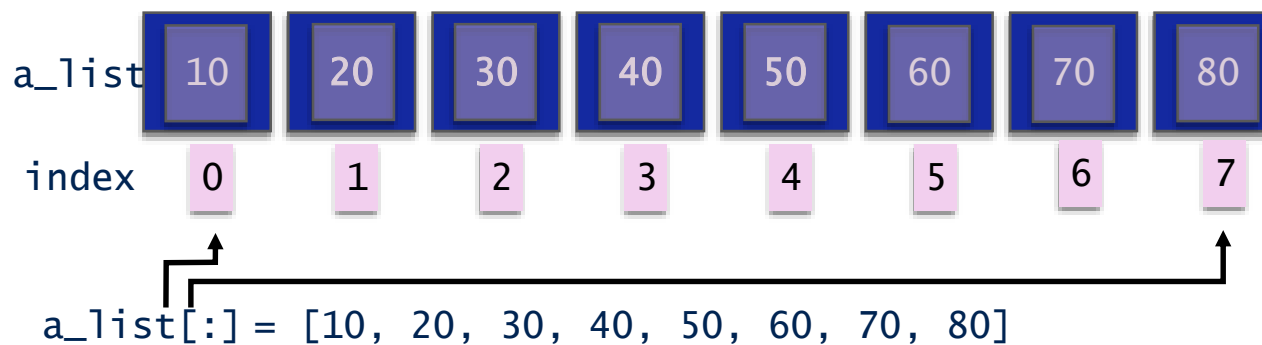
5. ການຕັດໃນ List

5.6. ເອົາທຸກຕຳແໜ່ງ

| [:] ໝາຍເຖິງເອົາທຸກຂອງອົງປະກອບ ໃນ list

```
1 a_list[:]
```

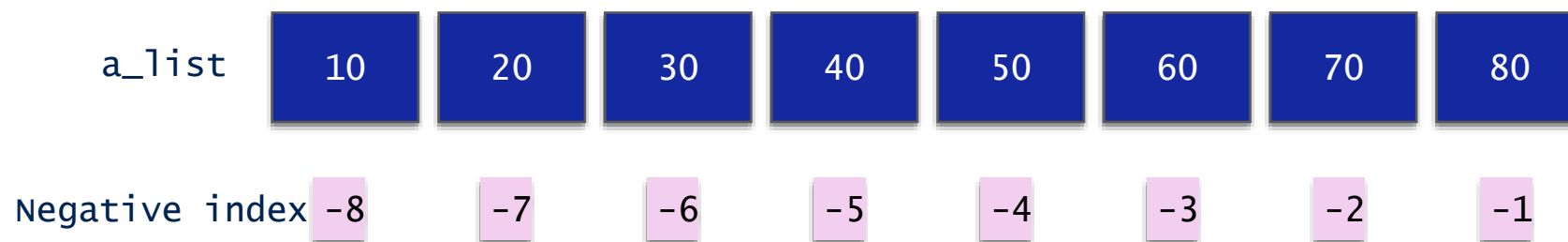
```
[10, 20, 30, 40, 50, 60, 70, 80]
```



5. ການຕັດໃນ List

5.7. ການຕັດຕຳແໜ່ງຂໍ້ມູນທີ່ເປັນຄ່າລົບ

❗ ຕຳແໜ່ງຂໍ້ມູນສຸດທ້າຍຈະແມ່ນ -1, ແລະ ສ່ວນຕຳແໜ່ງຖັດມາຈະແມ່ນ -2, -3, ... ດັ່ງສະແດງໃນຮູບລຸ່ມນີ້



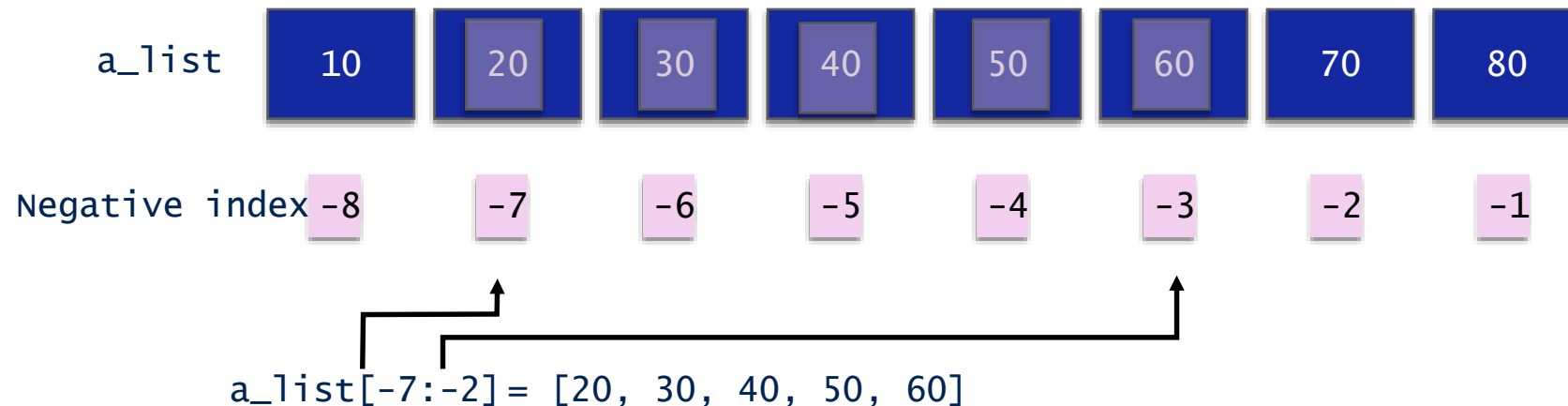
5. ການຕັດໃນ List

5.8. ການຕັດດ້ວຍຕຳແໜ່ງທີ່ເປັນຄ່າລົບ

| `a_list[-7:-2]`: ໝາຍວ່າເອົາອົງປະກອບຈາກ `a_list[-7]` ຫາ `a_list[-2-1]` ດ້ວຍການນຳໃຊ້ຕຳແໜ່ງທີ່ເປັນຄ່າລົບ ແລະ ໄດ້ຜົນຮັບດັ່ງນີ້ `[20, 30, 40, 50, 60]`.

```
1 a_list[-7:-2]
```

```
[20, 30, 40, 50, 60]
```



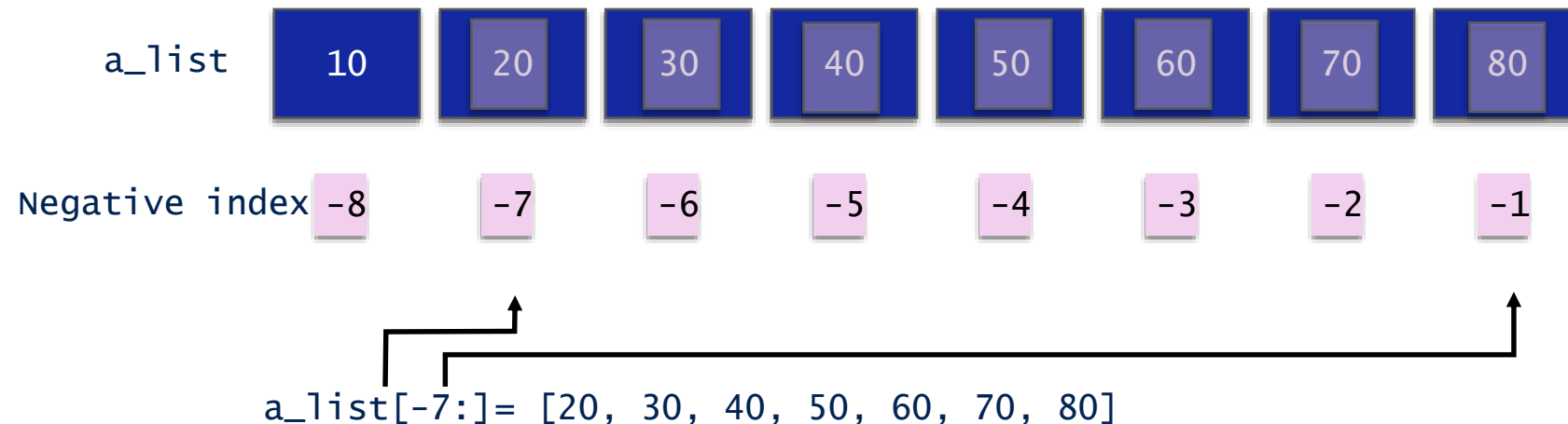
5. ການຕັດໃນ List

5.9. ບໍ່ເອົາຕຳແໜ່ງສຸດທ້າຍຈາກຕຳແໜ່ງຂໍ້ມູນທີ່ເປັນຄ່າລົບ

❗ ຖ້າບໍ່ເອົາຕຳແໜ່ງສຸດທ້າຍ ດ້ວຍການໃຊ້ຕຳແໜ່ງຂໍ້ມູນທີ່ເປັນລົບ ຈະໄດ້ອົງປະກອບດັ່ງລຸ່ມນີ້.

```
1 a_list[-7:]
```

```
[20, 30, 40, 50, 60, 70, 80]
```



5. ການຕັດໃນ List

5.10. ບໍ່ເອົາຕຳແໜ່ງທຳອິດຂອງຕຳແໜ່ງຂໍ້ມູນເປັນຄ່າລົບ

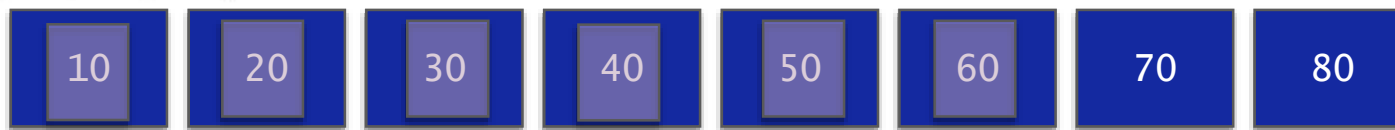
▮ ບໍ່ເອົາຕຳແໜ່ງທຳອິດສຳລັບການຕັດໂດຍນຳໃຊ້ຕຳແໜ່ງຂໍ້ມູນທີ່ເປັນຄ່າລົບ.

▮ `a_list[:-1]`: ຈະເອົາຕຳແໜ່ງຈາກ item ທຳອິດຫາ ຕຳແໜ່ງ $(-1-1) = -2$.

```
a_list[:-1]
```

```
[10, 20, 30, 40, 50, 60, 70]
```

a_list



Negative index



`a_list[:-1]` = [10, 20, 30, 40, 50, 60, 70]

6. ການກວດສອບຄ່າພາຍໃນ List

6.1. ຕົວດຳເນີນການ in

- ❑ ຕົວດຳເນີນການ “in” ຈະສົ່ງຄ່າ True ຫຼື False. ຫຼັງຈາກມີການກວດສອບອົງປະກອບໃນ list , ຖ້າວ່າມີຈະສົ່ງຄ່າ True ອອກມາ, ບໍ່ດັ່ງນັ້ນຈະສົ່ງຄ່າ False. (ຄຳສັ່ງ “not in” ຈະສົ່ງຄ່າກົງກັນຂ້າມກັບ in)
- ❑ ເຖິງຢ່າງໃດກໍຕາມ, ຟັງຊັນ for() ຈະສາມາດໃຊ້ຄຳສັ່ງ “in” ໄດ້.
- ❑ ນອກຈາກນີ້ ຄຳສັ່ງ in ຍັງສາມາດນຳໃຊ້ໃນໂຄງສ້າງຂໍ້ມູນແບບ strings, lists, ແລະ tuple.

```
1 a_list = [10, 20, 30, 40]
2 10 in a_list
```

True

```
1 50 in a_list
```

False

```
1 10 not in a_list
```

False

```
1 50 not in a_list
```

True

```
1 for n in a_list:
2     print(n, end=' ')
```

10 20 30 40

ກວດສອບອົງປະກອບທີ່ມີຄ່າເປັນ 10 ວ່າມີໃນ a_list ຫຼື ບໍ່ ຖ້າມີຈະສົ່ງຄ່າ True ບໍ່ດັ່ງນັ້ນ ຈະສົ່ງຄ່າ False.

6. ການກວດສອບຄ່າພາຍໃນ List

6.2. ນຳໃຊ້ຕົວດຳເນີນການ in

■ ນຳໃຊ້ຕົວດຳເນີນການ in ໃນ Python.

■ ຖ້າມີອົງປະກອບໃນ list, ຈະເຮັດໃຫ້ຄຳສັ່ງ in ສິ່ງຄ່າ True, ບໍ່ດັ່ງນັ້ນ ຈະສິ່ງຄ່າ False ດັ່ງສະແດງໃນ code ຄຳສັ່ງລຸ່ມນີ້.

```
1 n_list = [11, 22, 33, 44, 55, 66]
2
3 print(88 in n_list)      # 88 is not in the n_list
4 print(55 in n_list)      # 55 is in the n_list
```

False

True

6. ການກວດສອບຄ່າພາຍໃນ List

6.2. ນຳໃຊ້ຕົວດຳເນີນການ in

- | ການປ້ອງກັນການເກີດຄວາມຜິດພາດດ້ວຍການນຳໃຊ້ຕົວດຳເນີນການ “in” .
- | ກວດສອບອົງປະກອບໃນ list ກ່ອນຈະລຶບ ໂດຍໃຊ້ remove method ດັ່ງສະແດງລຸ່ມນີ້.

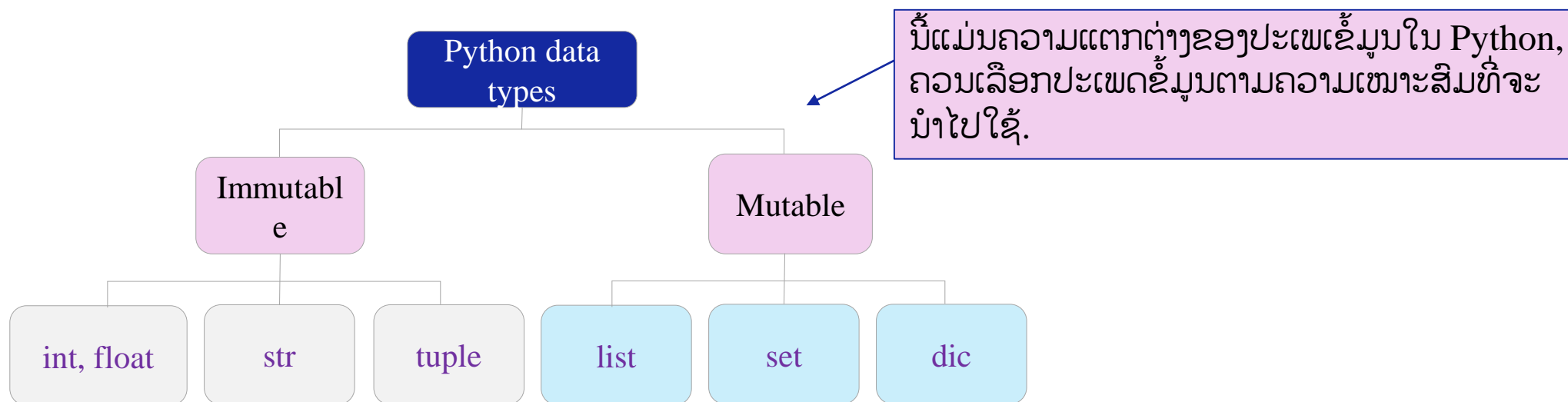
```
1 n_list = [11, 22, 33, 44, 55, 66]
2 if (55 in n_list) :      # If 55 is an element of the
3     n_list.remove(55)    # Delete 55 from the list
4 if (88 in n_list) :      # If 88 is an element of the
5     n_list.remove(88)    # Delete 88 from the list
6 print(n_list)
```

[11, 22, 33, 44, 66]

7. Tuple

7.1. ປະເພດຂໍ້ມູນປ່ຽນແປງບໍ່ໄດ້ (Immutable) ແລະ ປ່ຽນແປງໄດ້ (mutable)

- ໃນ Python ໄດ້ແບ່ງປະເພດຂໍ້ມູນປ່ຽນແປງບໍ່ໄດ້ ແລະ ປ່ຽນແປງໄດ້ ດັ່ງສະແດງໃນຮູບລຸ່ມນີ້. Tuple ແມ່ນຄ້າຍຄືກັບ list ຫຼາຍທີ່ສຸດ. ເຖິງຢ່າງໃດກໍຕາມ, ຂໍ້ມູນຂອງ tuple ເມື່ອມີການກຳນົດຄ່າແລ້ວ ຈະບໍ່ສາມາດປ່ຽນແປງໄດ້, ເຊິ່ງເອີ້ນວ່າປະເພດຂໍ້ມູນປ່ຽນແປງບໍ່ໄດ້ (immutable)
- ຂໍ້ດີຂອງຂໍ້ມູນແບບ tuple ແມ່ນເປັນໂຄງສ້າງທີ່ງ່າຍດາຍ ແລະ ເຂົ້າເຖິງໄດ້ໄວເມື່ອທຽບກັບ list. ການເລືອກໃຊ້ຂໍ້ມູນ tuple ຫຼື list ນັ້ນ ແມ່ນຂຶ້ນກັບຈຸດປະສົງທີ່ເໝາະສົມຈະນຳໄປໃຊ້.



7. Tuple

7.2. ເມື່ອມີການສ້າງ Tuple ແລະ ກໍານົດຄ່າຂໍ້ມູນແລ້ວຈະບໍ່ສາມາດປ່ຽນແປງໄດ້

■ ສ້າງ tuple ດັ່ງສະແດງໃນ Code ຄໍາສັ່ງລຸ່ມນີ້.

■ ຈາກ Code ຄໍາສັ່ງດັ່ງກ່າວເບິ່ງຄ້າຍຄືກັບ list, ແຕ່ມັນມີຄວາມແຕກຕ່າງຈາກ list.

```
1 colors = ("red", "green", "blue")
2 colors
```

('red', 'green', 'blue')

Tuple ທີ່ບັນຈຸອົງປະກອບເປັນຂໍ້ຄວາມ

```
1 numbers = (1, 2, 3, 4, 5 )
2 numbers
```

(1, 2, 3, 4, 5)

Tuple ທີ່ບັນຈຸອົງປະກອບເປັນເລກຈຳນວນຖ້ວນ

7. Tuple

7.2. ເມື່ອມີການສ້າງ Tuple ແລະ ກຳນົດຄ່າຂໍ້ມູນແລ້ວຈະບໍ່ສາມາດປ່ຽນແປງໄດ້

 Tuple ແມ່ນ object ທີ່ບໍ່ສາມາດປ່ຽນແປງຄ່າໄດ້.

```
1 t1 = (1, 2, 3, 4, 5)
2 t1[0] = 100
```

```
-----
TypeError                                 Traceback (most recent call last)
<ipython-input-107-614bcaadef71> in <module>
      1 t1 = (1, 2, 3, 4, 5)
----> 2 t1[0] = 100

TypeError: 'tuple' object does not support item assignment
```

- ລັກສະນະພິເສດທີ່ສຳຄັນຂອງຂໍ້ມູນແບບ tuple ແມ່ນບໍ່ສາມາດປ່ຽນແປງຄ່າໄດ້ ເມື່ອມີການກຳນົດ.
- ຖ້າວ່າມີການປ່ຽນແປງຄ່າຂໍ້ມູນຂອງ tuple ຈະສະແດງຄວາມຜິດພາດດັ່ງຂ້າງເທິງ ຫຼື TypeError occurs.
- ນອກຈາກນີ້ຍັງມີຄວາມຜິດພາດອື່ນໆທີ່ເກີດຂຶ້ນ ເຊັ່ນວ່າ ເມື່ອມີການລຶບຄ່າຂໍ້ມູນດ້ວຍການນຳໃຊ້ del t1[0].

| Paper coding

- ຕ້ອງເຂົ້າໃຈແນວຄິດພື້ນຖານຂອງຫຼັກສູດນີ້ໃຫ້ຄົບຖ້ວນກ່ອນຈະໄປສູ່ຂັ້ນຕອນຕໍ່ໄປ.
- ການທີ່ບໍ່ເຂົ້າໃຈແນວຄິດພື້ນຖານ ຈະເພີ່ມພາລະໃນການຮຽນຮູ້ຂອງຫຼັກສູດນີ້ ແລະ ຈະເຮັດໃຫ້ບໍ່ປະສົບຜົນສໍາເລັດ.
- ມັນອາດຈະເປັນເລື່ອງທີ່ຍາກຕອນນີ້, ແຕ່ຖ້າຢາກປະສົບຜົນສໍາເລັດໄດ້ນັ້ນ ພວກເຮົາຂໍແນະນຳໃຫ້ເຂົ້າໃຈແນວຄິດພື້ນຖານນີ້ຢ່າງເລິກເຊິ່ງ ແລະ ກ້າວໄປສູ່ຂັ້ນຕອນຕໍ່ໄປ.

Q1. ຈົ່ງສ້າງ list ທີ່ຊື່ prime_list ໂດຍມີອົງປະກອບທີ່ມີຄ່າ ເປັນຈຳນວນມູນ ຢູ່ລະຫວ່າງ 2~10 . ຈາກນັ້ນໃຫ້ນຳໃຊ້ຕຳແໜ່ງ
ຂໍ້ມູນໃນ list ເພື່ອຫາອົງປະກອບທຳອິດ ແລະ ພິມອອກມາທາງໜ້າຈໍ.

Conditions for Execution	1 st element of prime_list: 2
Time	5 Minutes



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

Q2. ຈົ່ງສ້າງ list ທີ່ຊື່ prime_list ໂດຍມີອົງປະກອບທີ່ມີຄ່າ ເປັນຈຳນວນມູນ ຢູ່ລະຫວ່າງ 2~10. ຈາກນັ້ນໃຫ້ນຳໃຊ້ append method ເພື່ອເພີ່ມອົງປະກອບທີ່ມີຄ່າເທົ່າ 11 ເຂົ້າໄປໃນ list ແລ້ວໃຫ້ພິມຜົນໄດ້ຮັບ ກ່ອນ ແລະ ຫຼັງ ເພີ່ມ ອອກທາງໜ້າຈໍ.

Conditions for Execution	Prime numbers : [2,3,5,7] Prime numbers after addition : [2,3,5,7,11]
Time	5 Minutes



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

Q3. ສໍາລັບ list1 ແລະ list2 ໃຫ້ນຳໃຊ້ nested for loop ເພື່ອຄູນຄ່າຂອງອົງປະກອບທຸກຕົວ ແລ້ວພິມຜົນໄດ້
ຮັບອອກມາທາງໜ້າຈໍ ດັ່ງສະແດງຜົນຮັບດັ່ງລຸ່ມນີ້.

Conditions for Execution

Declare list1 and list2 in the first and second rows. Use the nested for loop in the third and fourth row, and use the print loop in the fifth row.

Time

5 Minutes

```
list1 = [3,5,7]
```

```
list2 = [2,3,4,5,6]
```

Output example

```
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
7 * 2 = 14
7 * 3 = 21
7 * 4 = 28
7 * 5 = 35
7 * 6 = 42
```



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

| Let's code

1. List ແມ່ນຫຍັງ ແລະ ມີຄວາມສໍາຄັນແນວໃດ?

1.1. ຄວາມສໍາຄັນຂອງ list

▮ ທົດລອງບັນທຶກຂໍ້ມູນຄວາມສູງຂອງກຸ່ມຄົນ ດັ່ງສະແດງລຸ່ມນີ້.

```
1 height1 = 178.9 # Save the float type data.  
2 height2 = 173.5 # Save the float type data.  
3 height3 = 166.1 # Save the float type data.  
4 height4 = 164.3 # Save the float type data.  
5 height5 = 176.4 # Save the float type data.
```

ຖ້າມີຄົນ 100 ຄົນ, ຈະຕ້ອງໃຊ້ຕົວປ່ຽນຢູ່ 100 ໂຕທີ່ແຕກຕ່າງກັນ.

ສະນັ້ນ, ຖ້າມີ 1,000 ຄົນ, ຈະຕ້ອງໃຊ້ຕົວປ່ຽນ...??? ”



1. List ແມ່ນຫຍັງ ແລະ ມີຄວາມສໍາຄັນແນວໃດ?

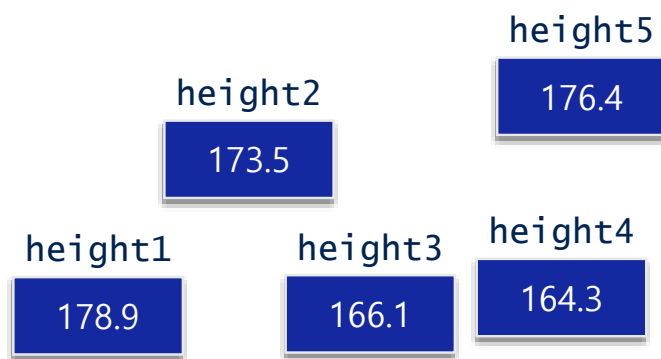
1.1. ຄວາມສໍາຄັນຂອງ list

ໃນ Python, ເຮົາສາມາດນໍາໃຊ້ເຄື່ອງໝາຍ [] ເພື່ອສ້າງ list. ເຊັ່ນຕົວຢ່າງ ບັນທຶກລວງສູງຂອງນັກສຶກສາ 5 ຄົນ ເຂົ້າໄປໃນ list ທີ່ຊື່ heights.

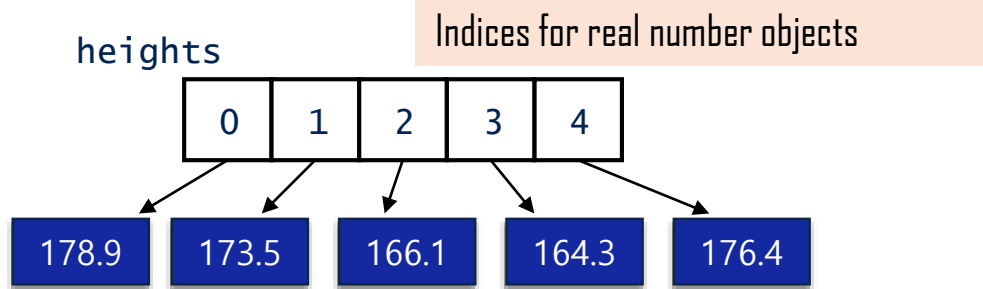
```
1 heights = [178.9, 173.5, 166.1, 164.3, 176.4]
2 heights
```

```
[178.9, 173.5, 166.1, 164.3, 176.4]
```

ໃຊ້ list ທີ່ມີຫຼາຍອົງປະກອບ ບັນຈຸຢູ່ໃນຕົວປ່ຽນດຽວ.



Data expressed in individual variables



Data expressed in the list

1. List ແມ່ນຫຍັງ ແລະ ມີຄວາມສໍາຄັນແນວໃດ?

1.2. ສ້າງ list ທີ່ມີຫຼາຍອົງປະກອບ

| ໃນ Python, list ສາມາດສ້າງຂຶ້ນພາຍໃຕ້ຕົວປ່ຽນ. ໃສ່ອົງປະກອບເຂົ້າໃນເຄື່ອງໝາຍນີ້ [] ແລະ ບັນທຶກຕົວປ່ຽນ ຈະເຮັດໃຫ້ຕົວປ່ຽນຂອງ list ຖືກສ້າງຂຶ້ນ. ລຸ່ມນີ້ແມ່ນຕົວຢ່າງ ການສ້າງ list ຊື່ bts ທີ່ປະກອບດ້ວຍອົງປະກອບ 3 ຕົວ.

```
1 bts = ['V', 'Jungkook', 'Jimin']
```

| ເຮົາສາມາດສ້າງ list ບໍ່ມີອົງປະກອບ ໄດ້ ດັ່ງສະແດງລຸ່ມນີ້.

```
1 bts = []
```

1. List ແມ່ນຫຍັງ ແລະ ມີຄວາມສໍາຄັນແນວໃດ?

1.2. ສ້າງ list ທີ່ມີຫຼາຍອົງປະກອບ

| ຈະນໍາໃຊ້ list ຫວ່າງເປົ້າແນວໃດ? ເຮົາສາມາດເພີ່ມ ອົງປະກອບເຂົ້າໄປໃນ list ຫວ່າງເປົ້າໄດ້ ດ້ວຍການນໍາໃຊ້ `append method` ດັ່ງສະແດງໃນ Code ຄໍາສັ່ງລຸ່ມນີ້.

```
1  bts = []  
2  bts.append("V")  
3  bts
```

`['V']`

| ວິທີທີ່ສະດວກທີ່ສຸດ ສໍາລັບການເພີ່ມອົງປະກອບເຂົ້າໄປໃນ list ແມ່ນນໍາໃຊ້ເຄື່ອງໝາຍ `+` ດັ່ງສະແດງຢູ່ທາງລຸ່ມນີ້.

```
1  bts = []  
2  bts = bts + ["V"]  
3  bts
```

`['V']`

2. ຄິດໄລ່ ຄວາມຍາວ, ຄ່າສູງສຸດ, ຄ່າຕໍ່າສຸດ, ຜົນບວກ ຂອງຂໍ້ມູນໃນ list

2.1. ນຳໃຊ້ຟັງຊັນໃນ list

■ ໃນພາສາ Python, list ຈະມີຟັງຊັນໃນຕົວ ປະກອບດ້ວຍ len, max, min, sum, ແລະ ຟັງຊັນອື່ນໆ ທີ່ສາມາດນຳໃຊ້ໄດ້ຢ່າງສະດວກ.

- ▶ ຟັງຊັນ len ຈະສົ່ງຄວາມຍາວ ຫຼື ຈຳນວນອົງປະກອບໃນ list ອອກມາ
- ▶ ຟັງຊັນ max ຈະສົ່ງອົງປະກອບທີ່ມີຄ່າໃຫຍ່ສຸດຂອງ list ອອກມາ ແລະ ຟັງຊັນ min ຈະສົ່ງອົງປະກອບທີ່ມີຄ່ານ້ອຍສຸດຂອງ list ອອກມາ
- ▶ ຖ້າວ່າ list ມີອົງປະກອບເປັນຕົວເລກ, ຟັງຊັນ sum ຈະສົ່ງຄ່າຜົນບວກຂອງອົງປະກອບເຫຼົ່ານັ້ນອອກມາ. ດັ່ງສະແດງໃນ Code ຄຳສັ່ງລຸ່ມນີ້

```
1 n_list = [200, 700, 500, 300, 400]
2 len(n_list)
```

5

```
1 max(n_list)
```

700

```
1 min(n_list)
```

200

```
1 sum(n_list)
```

2100

Let's code

2. ຄິດໄລ່ ຄວາມຍາວ, ຄ່າສູງສຸດ, ຄ່າຕໍ່າສຸດ, ຜົນບວກ ຂອງຂໍ້ມູນໃນ list

2.2. ຟັງຊັນອື່ນທີ່ສາມາດນຳໃຊ້ໄດ້ໃນ list

■ ເຮົາສາມາດເພີ່ມຂໍ້ມູນເຂົ້າໄປໃນ list ໄດ້ ດ້ວຍການນຳໃຊ້ຟັງຊັນ range ຄື: `list(range())` ດັ່ງສະແດງໃນ Code ຄຳສັ່ງລຸ່ມນີ້.

■ ນອກຈາກນີ້ ໃນ list, ຍັງມີ ຟັງຊັນ any ທີ່ສົ່ງຄ່າເປັນ True ຖ້າມີອົງປະກອບໃນ list ມີຢ່າງໜ້ອຍ 1 ຕົວ ຍົກເວັ້ນ ອົງປະກອບເປັນ 0. ໂດຍມີຮູບແບບການຂຽນຄື `any(n_list)`.

■ ໃນ `a_list` ບັນຈຸຄ່າເປັນ 0 ແລະ ຫວ່າງເປົ່າ (''), ສະນັ້ນ ຟັງຊັນ any ຈະສົ່ງຄ່າເປັນ False.

```
1 list(range(1,11))
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
1 n_list = [200, 700, 500, 300, 400]
2 a_list = [0, '']
```

```
1 any(n_list)
```

```
True
```

```
1 any(a_list)
```

```
False
```

Let's code

2. ຄິດໄລ່ ຄວາມຍາວ, ຄ່າສູງສຸດ, ຄ່າຕໍ່າສຸດ, ຜົນບວກ ຂອງຂໍ້ມູນໃນ list

2.3. ການເຂົ້າເຖິງອົງປະກອບໃນ list ໂດຍນຳໃຊ້ຕຳແໜ່ງຂໍ້ມູນ

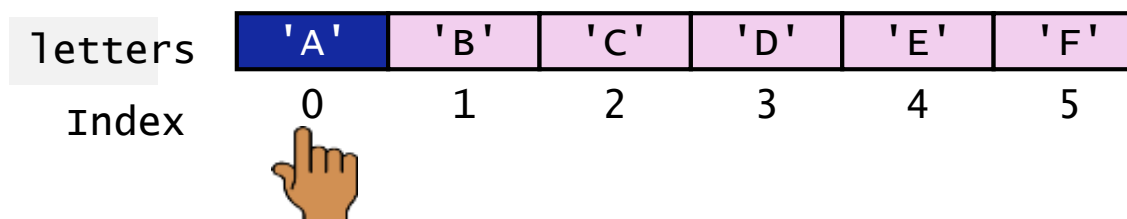
| ສົມມຸດມີ list ທີ່ຊື່ letters ທີ່ມີອົງປະກອບເປັນຕົວອັກສອນດັ່ງສະແດງໃນຮູບລຸ່ມນີ້.

```
1 letters = ['A', 'B', 'C', 'D', 'E', 'F']
```

| ຖ້າແຍກຕົວອັກສອນໃນ list ແລະ ນຳໃຊ້ຕຳແໜ່ງຂໍ້ມູນ ຈະໄດ້ຕຳແໜ່ງທີ່ 0 ແມ່ນ 'A', ຕຳແໜ່ງທີ່ 1 ແມ່ນ 'B', ຕາມລຳດັບ

```
1 letters[0] # Approaching to the first item of the list
```

'A'



2. ຄິດໄລ່ ຄວາມຍາວ, ຄ່າສູງສຸດ, ຄ່າຕໍ່າສຸດ, ຜົນບວກ ຂອງຂໍ້ມູນໃນ list

2.3. ການເຂົ້າຫາອົງປະກອບໃນ list ໂດຍນຳໃຊ້ຕຳແໜ່ງຂໍ້ມູນ

```
1 letters[1]
```

'B'

```
1 letters[2]
```


'C'

```
1 letters[-1]
```

'F'

ຕຳແໜ່ງຂໍ້ມູນທີ່ເປັນຄ່າລົບກໍສາມາດໃຊ້ໃນ list ໄດ້.

letter	A	B	C	D	E	F
Index	0	1	2	3	4	5
Negative index	-6	-5	-4	-3	-2	-1



Let's code

3. ການເພີ່ມອົງປະກອບຂອງ list elements

3.1. ການປະຕິບັດການຂອງອົງປະກອບໃນ list

▮ ການປະຕິບັດການຂອງອົງປະກອບໃນ list ຄື:

- ▶ ເພີ່ມອົງປະກອບດ້ວຍການນຳໃຊ້ຟັງຊັນ append ຫຼື insert. ເຊິ່ງຟັງ append ຈະເພີ່ມຂໍ້ມູນໄປຕໍ່ທ້າຍຂອງ list, ແຕ່ ຟັງຊັນ insert(index, item) ຈະເພີ່ມຂໍ້ມູນໄປໄວ້ໃນຕຳແໜ່ງທີ່ກຳນົດຕາມຕ້ອງການ.

```
1 slist = ['David', 178.9, 'John', 173.5, 'Jane', 176.1]
2 print(slist)
3 slist.insert(4, "Petter")
4 slist.insert(5, 168.1)
5 print(slist)
```

```
['David', 178.9, 'John', 173.5, 'Jane', 176.1]
```

```
['David', 178.9, 'John', 173.5, 'Petter', 168.1, 'Jane', 176.1]
```


Let's code

4. ການລຶບອົງປະກອບໃນ list

4.1. ລຶບອົງປະກອບໃນ list

- ▮ ກ່ອນໜ້ານີ້, ພວກເຮົາຮຽນຮູ້ກ່ຽວກັບຕົວດຳເນີນການ ແລະ methods ເພື່ອເພີ່ມ ຫຼື ປ່ຽນແປງອົງປະກອບ. ໃນຫົວຂໍ້ນີ້ ຈະມາຮຽນຮູ້ກ່ຽວກັບການລຶບອົງປະກອບໃນ list. ສໍາລັບການລຶບ ແມ່ນນຳໃຊ້ remove ຫຼື pop methods ຫຼື ຈະໃຊ້ຄໍາສັ່ງ del ກໍໄດ້.
- ▮ ກ່ອນຈະນຳໃຊ້ຟັງຊັນ remove, ໃຫ້ກວດສອບວ່າມີອົງປະກອບທີ່ຕ້ອງການລຶບ ຫຼື ບໍ່ ໂດຍໃຊ້ຕົວດຳເນີນການ “in” .

```
1 bts = [ "V", "J-Hope", "Suga", "Jungkook" ]  
2 bts.remove("Jungkook")  
3 bts
```

```
['V', 'J-Hope', 'Suga']
```

ໃຊ້ remove method ເພື່ອລຶບ ອົງປະກອບຈາກ list

```
1 if 'Suga' in bts:  
2     bts.remove('Suga')  
3 print(bts)
```

```
['V', 'J-Hope']
```

ຕົວດຳເນີນການ “in” ໃຊ້ເພື່ອກວດສອບອົງປະກອບໃນ list ຫຼື tuple. ກ່ອນການລຶບອົງປະກອບ ຖ້າ ອົງປະກອບທີ່ຈະລຶບມີໃນ list ຫຼື tuple ກໍສາມາດລຶບອົງປະກອບນັ້ນໄດ້.

- remove method ໃຊ້ລຶບອົງປະກອບທີ່ຕ້ອງການໃນ list .

Let's code

4. ການລຶບອົງປະກອບໃນ list

4.1. ລຶບອົງປະກອບໃນ list

❗ ມີຫຼາຍວິທີໃນການລຶບອົງປະກອບໃນ list.

- ▶ pop ເປັນຟັງຊັນໜຶ່ງທີ່ສາມາດລຶບອົງປະກອບໃນ list ໄດ້. ແຕ່ມັນຕ່າງຈາກຟັງຊັນ remove, ເນື່ອງຈາກຟັງຊັນ pop ຈະລຶບອົງປະກອບທ້າຍສຸດຂອງ list.

```
1  bts = ["V", "J-Hope", "Suga", "Jungkook"]
2  last_member = bts.pop() # Delete and return the last item 'Jungkook'
3  print(last_member)
4  print(bts)
```

```
Jungkook
['V', 'J-Hope', 'Suga']
```

4. ການລຶບອົງປະກອບໃນ list

4.2. ຄໍາເຕືອນສໍາລັບການລຶບອົງປະກອບຂອງ list

⚠ ຄໍາສັ່ງ del ບໍ່ແມ່ນ method ຂອງ list.

- ▶ ຄໍາສັ່ງ del ບໍ່ແມ່ນ method ຂອງ list. ຄໍາສັ່ງນີ້ ມັນແມ່ນຄໍາສັບສໍາຄັນຂອງ Python ແລະ ມັນສາມາດລຶບສະເພາະອົງປະກອບຈາກໜ່ວຍຄວາມຈໍາ ດ້ວຍການນໍາໃຊ້ຕໍາແໜ່ງຂໍ້ມູນດັ່ງສະແດງໃນລຸ່ມນີ້.
- ▶ ເວລານໍາໃຊ້ຄໍາສັ່ງ del ໃນຮູບແບບ method, ຈະມີ SyntaxError ເກີດຂຶ້ນ.

```
1 bts = [ "V", "J-Hope", "Suga", "Jungkook"]  
2 del bts[0]    # Command to delete the first item of the list  
3 bts
```

```
['J-Hope', 'Suga', 'Jungkook']
```

```
1 bts[0].del
```

```
File "<ipython-input-5-390ae27b77c9>", line 1  
    bts[0].del  
          ^
```

```
SyntaxError: invalid syntax
```

5. ການຕັດໃນ List

5.1. ວິທີຕັດອົງປະກອບ

ການຕັດເອົາອົງປະກອບໃນ list ທີ່ຕ້ອງການ

- ການສ້າງ list ໃໝ່ ດ້ວຍການຕັດເອົາບາງອົງປະກອບຈາກ list, ເຊິ່ງໃຊ້ເຄື່ອງໝາຍຈຳສອງເມັດ ':' ດັ່ງສະແດງໃນ Code ຄໍາສັ່ງລຸ່ມນີ້.

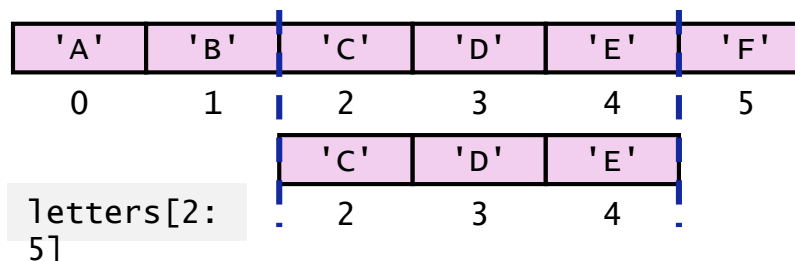
```
1 letters = ['A', 'B', 'C', 'D', 'E', 'F']
```

```
1 letters[2:5] # slicing from the 3rd item to 5th item of the list
```

```
['C', 'D', 'E']
```



letters
Index



Let's code

5. ການຕັດໃນ List

5.1. ວິທີຕັດ

- | ການຕັດ ຈະບໍ່ເຮັດໃຫ້ list ເດີມຖືກການປ່ຽນແປງ. ມັນເປັນການສ້າງ list ໃໝ່ຂຶ້ນມາ ດ້ວຍການຕັດເອົາບາງສ່ວນໃນ list ເດີມເທົ່ານັ້ນ.
- | Code ຄໍາສັ່ງລຸ່ມນີ້ແມ່ນການຕັດເອົາອົງປະກອບທີ 1 ຮອດອົງປະກອບທີ 3 ໝາຍຄວາມວ່າ ເອົາຕໍາແໜ່ງຂໍ້ມູນທີ 0 ຮອດຕໍາແໜ່ງທີ 2.

```
1 letters[:3]  
['A', 'B', 'C']
```

- | ຖ້າບໍ່ເອົາຕໍາແໜ່ງຂໍ້ມູນທີ 2 ມັນກໍຈະຕັດເອົາເລີ່ມແຕ່ຕໍາແໜ່ງທີ 3 ໄປຈົນຮອດຕໍາແໜ່ງສຸດທ້າຍ.

```
1 letters[3:]  
['D', 'E', 'F']
```

Let's code

5. ການຕັດໃນ List

5.1. ວິທີຕັດ

ການຕັດ ຈະບໍ່ເຮັດໃຫ້ list ເດີມຖືກການປ່ຽນແປງ. ມັນເປັນການສ້າງ list ໃໝ່ຂຶ້ນມາ ດ້ວຍການຕັດເອົາບາງສ່ວນໃນ list ເດີມເທົ່ານັ້ນ

```
1 letters[:]  
['A', 'B', 'C', 'D', 'E', 'F']
```

ຈຳສອງເມັດອ້າງອີງຈາກຕຳແໜ່ງເລີ່ມຕົ້ນຈົນຮອດຕຳແໜ່ງສຸດທ້າຍຂອງຂໍ້ມູນ

```
1 letters[:2]  
['A', 'C', 'E']
```

ຕັດເອົາອົງປະກອບທຸກໆສອງຕຳແໜ່ງ ໝາຍຄວາມວ່າ $tep = 2$

```
1 letters[::-1] # Read from the back to front to create elements  
['F', 'E', 'D', 'C', 'B', 'A']
```

Let's code

6. ບັນດາ method ຕ່າງໆ ໃນ list

6.1. method ສໍາລັບການລຽງລຳດັບ

| Method ການລຽງລຳດັບ - sort

- ▶ ການລຽງຂໍ້ມູນແຕ່ນ້ອຍຫາໃຫຍ່ ເປັນຄ່າ default
- ▶ ຂຽນ dot (.) ແລະ “sort” ຫຼັງ ຊື່ list

| ການລຽງຂໍ້ມູນແຕ່ນ້ອຍຫາໃຫຍ່



- ▶ ການລຽງຂໍ້ມູນກັບການເພີ່ມຂຶ້ນຂອງຄ່າຂໍ້ມູນ

| ການລຽງຂໍ້ມູນແຕ່ໃຫຍ່ຫານ້ອຍ



- ▶ ການລຽງຂໍ້ມູນແມ່ນການຫຼຸດລົງຂອງຄ່າຂໍ້ມູນ

Let's code

6. ບັນດາ method ຕ່າງໆ ໃນ list

6.2. method ການລຽງລຳດັບ

- ນຳໃຊ້ sort method ເພື່ອລຽງລຳດັບຂໍ້ມູນຈາກນ້ອຍໄປຫາໃຫຍ່. ແຕ່ຖ້າ ໃສ່ reverse = True ເຂົ້າໄປໃນຟັງຊັນ ຈະເປັນການລຽງລຳດັບຂໍ້ມູນຈາກໃຫຍ່ໄປຫານ້ອຍ, ດັ່ງສະແດງໃນ Code ຄຳສັ່ງລຸ່ມນີ້.
- ▶ argument ເປັນຄຳສະເພາະຂອງ Python ໃຊ້ອ້າງອີງ argument ໃນ method ຫຼື function, ເຊິ່ງຈະໄດ້ອະທິບາຍໃນພາຍຫຼັງ.

```
1 list1 = [20, 10, 40, 50, 30]
2 list1.sort()                # Ascending order sorting of the list1 elements
3 list1
```

[10, 20, 30, 40, 50]

```
1 list1.sort(reverse = True)
2 print(list1)
```

[50, 40, 30, 20, 10]

6. ບັນດາ method ຕ່າງໆ ໃນ list

6.3. Method ຕ່າງໆໃນ list

Method	Function
index(x)	ຄົ້ນຫາຕຳແໜ່ງຂອງອົງປະກອບ x.
append(x)	ເພີ່ມອົງປະກອບ x ເຂົ້າໄປຕໍ່ທ້າຍ ອົງປະກອບສຸດທ້າຍໃນ list.
count(x)	ນັບຈຳນວນອົງປະກອບພາຍໃນ list.
extend([x1, x2])	ແຊກ [x1, x2] ເຂົ້າໄປໃນ list.
insert(index, x)	ເພີ່ມ x ເຊິ່ງແມ່ນອົງປະກອບໃໝ່ເຂົ້າໄປໃນ list ແຕ່ການເພີ່ມນີ້ຕ້ອງລະບຸຕຳແໜ່ງ (index).
remove(x)	ລຶບ ອົງປະກອບ x ໃນ list.
pop(index)	ລຶບອົງປະກອບ ໂດຍອ້າງອີງຕຳແໜ່ງທີ່ລຶບ, ຖ້າບໍ່ອ້າງອີງຕຳແໜ່ງ ມັນຈະລຶບອົງປະກອບທ້າຍສຸດຂອງ list.
sort()	ເປັນການລຽບລຳດັບຂອງຂໍ້ມູນຈາກນ້ອຍໄປຫາໃຫຍ່, ແຕ່ຖ້າຢາກໃຫ້ລຽງລຳດັບແຕ່ໃຫຍ່ຫານ້ອຍຕ້ອງໄດ້ໃສ່ reverse = True ເຂົ້າໄປໃນຟັງຊັນ.
reverse()	ເປັນການປິ່ນລຳດັບຂອງອົງປະກອບໃນ list.

| Pair programming



Pair Programming Practice

Guideline, mechanism
Preparing pair programming
Effective preparation
student will not

ແນວທາງ, ກົນໄກ ແລະ ແຜນສຸກເສີນ

ການຈັບຄູ່ຂຽນໂປຣແກຣມ ເປັນການຈັບຄູ່ຂອງນັກຮຽນເພື່ອເຮັດວຽກມອບໝາຍ, ນັກຮຽນຄວນມີແຜນ ແລະ ສາມາດປ່ຽນແທນກັນໄດ້ ໃນກໍລະນີມີຜູ້ໃດໜຶ່ງບໍ່ສາມາດເຂົ້າຮ່ວມເຮັດວຽກມອບໝາຍໄດ້ບໍ່ວ່າໃນກໍລະນີໃດກໍຕາມ ເຊິ່ງບັນຫາເລົ່ານີ້ຕ້ອງເຮັດໃຫ້ຈະແຈ້ງ ແລະ ກໍບໍ່ແມ່ນຄວາມຜິດຂອງນັກຮຽນທີ່ຈັບຄູ່ບໍ່ດີ.

ຈັບຄູ່ທີ່ຄ້າຍຄືກັນ, ບໍ່ຈຳເປັນເທົ່າທຽມກັນ, ຄວາມສາມາດເປັນຄູ່ຮ່ວມ

Pairing similar
Pair programming
Teachers must encourage
very weak student

ງານ

ການຈັບຄູ່ຂຽນໂປຣແກຣມ ຈະໄດ້ຮັບຜົນດີກໍຕໍ່ເມື່ອນັກຮຽນມີຄວາມສາມາດຄ້າຍຄືກັນ ໝາຍວ່າ ບໍ່ຈຳເປັນຕ້ອງມີຄວາມສາມາດຄືກັນກໍໄດ້, ແຕ່ວ່າ ການຈັບຄູ່ນັກຮຽນທີ່ມີຄວາມສາມາດແຕກຕ່າງກັນຫຼາຍ ກໍຈະເຮັດໃຫ້ບໍ່ສົມດຸນກັນ. ຄູສອນຮູ້ດີວ່າ ການຈັບຄູ່ກັນບໍ່ແມ່ນ ຍຸດທະສາດ “ແບ່ງເພື່ອເອົາຊະນະ” ແຕ່ເປັນຄວາມພະຍາຍາມເຮັດວຽກຮ່ວມກັນຂອງນັກຮຽນໃຫ້ປະສິບຜົນສຳເລັດ. ຄູຄວນຫຼີກເວັ້ນການຈັບຄູ່ກັນລະຫວ່າງນັກຮຽນອ່ອນ ແລະ ນັກຮຽນເກັ່ງ.

Motivate student
Offering extra incentive

ກະຕຸ້ນນັກຮຽນໂດຍການໃຫ້ສິ່ງຈູງໃຈພິເສດ

ຂໍສະເໜີແຮງຈູງໃຈທີ່ເຮັດໃຫ້ນັກຮຽນຈັບຄູ, ໂດຍສະເພາະນັກຮຽນທີ່ມີຄວາມສາມາດສູງ. ບາງຄູສອນໄດ້ພົບວ່າ ການຈັບຄູ່ເຮັດວຽກມອບໝາຍ ແມ່ນມີປະໂຫຍດ ສຳລັບໜຶ່ງ ຫຼື ສອງວຽກມອບເທົ່ານັ້ນ

“...the mouse.”
...ar that the active

...ed participation.
...ould avoid pairing

...assignments.



Pair Programming Practice

Guideline, me

Preparing pair p
Effective prepar
student will not

ປ້ອງກັນການບໍ່ຕັ້ງໃຈໃນຮຽນຂອງ

ນັກຮຽນ

ສິ່ງທ້າທາຍສໍາລັບຄູແມ່ນເພື່ອຊອກຫາວິທີທີ່ຈະປະເມີນຜົນການຮຽນຂອງນັກຮຽນ, ຄູຮູ້ບໍ່ວ່າ ນັກຮຽນ ໄດ້ຕັ້ງໃຈຮຽນ ຫຼື ບໍ່ຕັ້ງໃຈຮຽນ. ຜູ້ສ່ຽວຊານໄດ້ແນະນຳໃຫ້ທົບທວນການອອກແບບຫຼັກສູດການຮຽນ ແລະ ຮູບແບບການປະເມີນ ພ້ອມທັງປຶກສາຫາລືຢ່າງຈິງຈັງກັບນັກຮຽນກ່ຽວກັບພຶດຕິກຳທີ່ຈະບໍ່ຕັ້ງໃຈຮຽນ ນອກຈາກນີ້ຍັງໄດ້ແນະນຳມອບວຽກມອບໝາຍໃຫ້ນັກຮຽນ ພ້ອມທັງອະທິບາຍໃຫ້ເຂົາເຈົ້າຢ່າງຈະແຈ້ງ

ing the mouse."
ar that the active

Pairing similar

Pair programi
Teachers must e
very weak stude

ສະພາບແວດລ້ອມຂອງການຮຽນຮູ້ຮ່ວມກັນ

ສະພາບແວດລ້ອມການຮຽນຮູ້ຮ່ວມກັນເກີດຂຶ້ນໄດ້ທຸກເວລາທີ່ຜູ້ສອນຮຽກຮ້ອງໃຫ້ນັກຮຽນເຮັດວຽກຮ່ວມກັນໃນກິດຈະກຳການຮຽນຮູ້ ເຊິ່ງອາດຈະເປັນກິດຈະກຳທີ່ເປັນທາງການ ແລະ ບໍ່ເປັນທາງການ ແລະ ອາດຈະບໍ່ລວມເຖິງການປະເມີນຜົນການຮຽນໂດຍກົງ. ເຊັ່ນຕົວຢ່າງ ໃຫ້ນັກຮຽນຈັບຄູ່ກັນເພື່ອເຮັດວຽກມອບໝາຍ ໂດຍນັກຮຽນຈະຕ້ອງທົບທວນກ່ຽວກັບການສອນຂອງອາຈານທີ່ຜ່ານມາ ແລະ ລະດົມແນວຄິດພາຍໃນກຸ່ມ ພ້ອມທັງມີການແບ່ງວຽກໃຫ້ແຕ່ລະຄົນຮັບຜິດຊອບ ຈາກນັ້ນກໍໃຫ້ມີການແລກປ່ຽນຄວາມຄິດເຫັນເຊິ່ງກັນ ແລະ ກັນ ເພື່ອເຮັດວຽກມອບໝາຍໃຫ້ສໍາເລັດຕາມເປົ້າໝາຍທີ່ວ່າໄວ້.

ed participation.
ould avoid pairing

assignments.

Motivate student

Offering extra in

Q1. ຈົ່ງສ້າງ List ທີ່ມີອົງປະກອບດັ່ງນີ້ `s_list = ['abc', 'bcd', 'bcdefg', 'abba', 'cddc', 'opq']`, ຈາກນັ້ນ ຂຽນ Code ຄໍາສັ່ງຕາມເງື່ອນໄຂທີ່ໄດ້ອະທິບາຍລຸ່ມນີ້.

| ບໍ່ໃຫ້ໃຊ້ຟັງຊັນ `min` ຫຼື `sort method` ເພື່ອພົບຄວາມຍາວຂອງອົງປະກອບທີ່ສັ້ນທີ່ສຸດໃນ `s_list`. (ຖ້າມີອົງປະກອບທີ່ມີຄວາມຍາວສັ້ນທີ່ສຸດຫຼາຍຕົວ, ແມ່ນໃຫ້ພິມອົງປະກອບທຳອິດອອກມາ, ດັ່ງສະແດງຜົນໄດ້ຮັບລຸ່ມນີ້.)

Output example

The shortest string : abc

Q2. ຈົ່ງສ້າງ List ທີ່ມີອົງປະກອບດັ່ງນີ້ `s_list = ['abc', 'bcd', 'bcdefg', 'abba', 'cddc', 'opq']`, ຈາກນັ້ນຂຽນ Code ຄໍາສັ່ງຕາມເງື່ອນໄຂທີ່ໄດ້ອະທິບາຍລຸ່ມນີ້.

▮ ບໍ່ໃຫ້ໃຊ້ຟັງຊັນ `min` ຫຼື `sort method` ເພື່ອພົບຄວາມຍາວຂອງອົງປະກອບຍາວທີ່ສຸດໃນ `s_list`. (ຖ້າມີອົງປະກອບທີ່ມີຄວາມຍາວ ຍາວທີ່ສຸດຫຼາຍຕົວ, ໃຫ້ພິມຕົວທໍາອິດອອກມາ ດັ່ງສະແດງລຸ່ມນີ້)

Output example

```
The longest string :  
bcdefg
```

Q3. ຈົ່ງສ້າງ List ທີ່ມີອົງປະກອບດັ່ງນີ້ `s_list = ['abc', 'bcd', 'bcdefg', 'abba', 'cddc', 'opq']`, ຈົ່ງຂຽນ code ຄໍາສັ່ງຕາມເງື່ອນໄຂທີ່ໄດ້ອະທິບາຍລຸ່ມນີ້.

| ຈາກ `s_list` ສັງເກດເຫັນວ່າ ມີອົງປະກອບສາມຕົວທີ່ມີຄວາມຍາວເທົ່າກັນຄື `'abc', 'bcd', 'opq'`. ຈົ່ງຂຽນໂປຣແກຣມ ເພື່ອພິມອົງປະກອບດັ່ງກ່າວອອກມາທາງໜ້າຈໍ ດັ່ງສະແດງລຸ່ມນີ້ ດ້ວຍການນຳໃຊ້ຟັງຊັນ `sort(key=len)` ເພື່ອລຽງລຳດັບບັນດາອົງປະກອບເຫຼົ່ານີ້.

Output example

The shortest strings : `'abc', 'bcd', 'opq'`