

# ပါတທီ 16.

# ຂໍ້ມູນປະເພດກຸມ (Set Data Types)

## Learning objectives

- ✓ ສາມາດສ້າງປະເພດຂໍ້ມູນກຸ່ມທີ່ບໍ່ອະນຸຍາດໃຫ້ຊ້າກັນຂອງຄ່າຂໍ້ມູນ.
- ✓ ສາມາດກວດສອບຄ່າໃນກຸ່ມຂໍ້ມູນຜ່ານຕົວດຳເນີນການ `in`.
- ✓ ສາມາດໃຊ້ກຸ່ມຂໍ້ມູນເພື່ອສ້າງກຸ່ມເປົ້າຫວ່າງຫຼືປົ່ງນຄ່າຂອງຂໍ້ມູນອື່ນເປັນຂໍ້ມູນປະເພດກຸ່ມ
- ✓ ສາມາດຊອກຫາຄວາມສຳພັນລະຫວ່າງອີງປະກອບຂອງສອງກຸ່ມຂໍ້ມູນໄດ້ຫຼັງຈາກເປົ້າການດຳເນີນງານທີ່ສາມາດນຳໃຊ້ກັບກຸ່ມຂໍ້ມູນ, ເຊັ່ນ: ການໂຮມ ແລະການຕັດ.
- ✓ ສາມາດທິດສອບໄດ້ວ່າສອງກຸ່ມຂໍ້ມູນຄືກັນຫຼືບໍ່ ໂດຍຜ່ານຕົວດຳເນີນການປົງບທູບ. ມັນຍັງສາມາດກວດສອບວ່າເປັນກຸ່ມຂໍ້ມູນຍ່ອຍ ຫຼື subset..
- ✓ ສາມາດພິມກຸ່ມຂໍ້ມູນໄດ້ຢ່າງຈ່າຍດາຍໂດຍໃຊ້ຄໍາສັງວິນລຸບ ແລະໃສ່ຄ່າໂດຍໃຊ້ຄໍາສັງແບບມີເງື່ອນໄຂ.

## ຈຸດປະສົງການຮຽນຮູ້

- ✓ ຮຽນຮູ້ວິທີການສ້າງຂໍ້ມູນປະເພດກຸ່ມໄດ້ຍື່ຕ້ອງລົງລຳດັບເຊິ່ງບໍ່ອະນຸຍາໃຫ້ຊ້າກັນຂອງມູນຄ່າຂໍ້ມູນ.
- ✓ ພວກເຮົາຮຽນຮູ້ວິທີການກວດສອບວ່າມີຄ່າຢູ່ໃນກຸ່ມຂໍ້ມູນໄດ້ຜ່ານຕົວດຳເນີນການ `in`,
- ✓ ຮຽນຮູ້ວິທີລືບຂໍ້ມູນທີ່ຊ້າກັນໄດ້ຍື່ນມາໃຊ້ກັບກຸ່ມຂໍ້ມູນທີ່ສະດວກສະບາຍ.
- ✓ ຮຽນຮູ້ການດຳເນີນງານທີ່ສະດວກສະບາຍທີ່ສາມາດນຳໃຊ້ກັບກຸ່ມຂໍ້ມູນ, ເຊັ່ນ: ການໂຮມ ແລະ ການຕັດ.
- ✓ ຮຽນຮູ້ `subsets` ໂດຍຜ່ານຕົວດຳເນີນການປັບປຸງ.

## ແນວຄວາມຄືດທີ່ເຈົ້າຈະຕັອງຮູ້ຈາກບົດຮຽນທີ່ຜ່ານມາ

- ✓ ວິທີການພິມອົງປະກອບທັງໝົດໄດ້ໃຊ້ຄວາມຍາວຂອງແຕ່ລະ `array` ເຊັ່ນ `len`, `range`, `in`, ແລະອື່ນໆ.
- ✓ ວິທີການນຳໃຊ້ ອອບເຈັກແລະຟັງຊັ້ນ ວັດຈະນານຸກົມ. (`del`, `index`, `pop...` ແລະອື່ນໆ)
- ✓ ຮູ້ຄວາມແຕກຕ່າງຂອງຂໍ້ມູນແຕ່ລະປະເພດເຊັ່ນ: `lists`, `tuples`, `dictionaries`, ແລະ `strings`, ແລະສາມາດນຳໃຊ້ປະເພດຂໍ້ມູນທີ່ເຫັນຈະສົມກັບສະຖານະການ.

# Keywords

set

Set calculation

Set comparison

Conditional  
statement

# Mission

## 1. Real world problem

### 1.1. ຊອກຫາຄວາມເປັນໄປໄດ້ ໃນການໂຍນໝາກກະລອກ



- ▶ ໃນພາລະກິດນີ້, ເຮົາຈະສົມມຸດວ່າຫຼາຍໆຄືນກຳລັງຫຼົ້ມ Monopoly.
- ▶ ເກມກະດານແມ່ນຫຼົ້ມຜ່ານການໂຍນ ໝາກກະລອກ, ຖ້າໂຍນສອງໜ່ວຍພ້ອມກັນ ແລະການເຄື່ອນຍ້າຍໄປໃນກະດານແມ່ນຂຶ້ນກັບຜົນບວກສອງໜ່ວຍທີ່ປາກິດໃນການໂຍນ,
- ▶ ເພື່ອຊະນະເກມ, ພວກເຮົາມາຊອກຫາຄ່າກະຕວງທີ່ຈະໄດ້ຄ່າຫຼາຍກ່ວ່າຈຳນວນທີ່ກຳນົດ.
- ▶ ການນຳໃຊ້ຄ່າກະຕວງນີ້, ມັນຈະເປັນປະໂຫຍດໃນເກມຄົ້ນຫາ

## 1. ບັນຫາໄລກຂອງແຫ່ງຄວາມເປັນຈິງ

### 1.2. Monopoly ແມ່ນຫຍັງ?



- ▶ Monopoly ແມ່ນເກມກະດານທີ່ເລີ່ມຕົ້ນໃນສະຫະລັດໃນປີ 1903. ໃນເກມ, ຜູ້ຫຼັນໄຍນໝາກກະລອກ, ເຄື່ອນຍ້າຍໃນກະດານ, ຊື້ອະສັງຫາລິມະຊັບ, ເກັບຄ່າເຊົ້າຈາກຜູ້ນອື່ນທີ່ມາຮອດຊັບສິນຂອງເຮົາ, ຜູກຂາດອະສັງຫາລິມະຊັບ, ກໍ່ສ້າງ, ພັດທະນາ, ແລະ ຄວບຄຸມອາຄານ. ນອກນັ້ນເຮົາຍັງສາມາດແລກປ່ຽນກັບຜູ້ນອື່ນໆເພື່ອບັນລຸເບົ້າທາມຢ່າງການຜູ້ກຂາດ. ຖ້າຜູ້ອື່ນອື່ນໆທັງໝົດລົ້ມລະລາຍ, ໃນຂະນະທີ່ໄດ້ຮັບຄ່າເຊົ້າ, ບຸກຄົນນີ້ຈະກາຍເປັນຜູ້ຊະນະ. ໃນປັດຈຸບັນເຈົ້າຂອງລິຂະສິດໂດຍ Hasbro, ເຊິ່ງເປັນບໍລິສັດເຄື່ອງຫຼັນທີ່ໃຫຍ່ທີ່ສຸດໃນສະຫະລັດ. ເກມນີ້ຍັງເປັນທີ່ນີ້ຍົມໝາຍໃນທຸກໆມັນ.

## 1. ບັນຫາໄລກຂອງແຫ່ງຄວາມເປັນຈີງ

### 1.3. ກຽນຮູ້ເພີ່ມເຕີມກ່ຽວກັບ Monopoly

| ເພື່ອກຽນຮູ້ເພີ່ມເຕີມກ່ຽວກັບ Monopoly , ເບິ່ງຈາກ <https://boardgamegeek.com/boardgame/1406/monopoly>.



## 1. ບັນຫາໄລກຊອງແຫ່ງຄວາມເປັນຈິງ

### 1.4. ກະຕິກາການຫຼັມ Monopoly (Monopoly Rules)



- ▶ ໃນເກມ Monopoly, ໃຊ້ໝາກກະລອກສອງໜ່ວຍ, ແຕ່ພວກເຮົາຈະສົມມຸດວ່າໝາກກະລອກສາມໜ່ວຍ.
- ▶ ສົມມຸດວ່າໝາກກະລອກສາມໜ່ວຍ ແຕ່ລະໜ່ວຍມີທີກໍ່າ
- ▶ ເຮົາໂຢນໝາກກະລອກນີ້ແລະພິມຈຳນວນຂອງກຳລະນີທີ່ສາມາດເກີດຂຶ້ນໄດ້.
- ▶ ຕໍ່ໄປ, ຄວາມເປັນໄປໄດ້ຂອງຜົນລວມຂອງໜ້າຈະໃຫຍ່ກວ່າຫຼືເທົ່າກັບຄ່າ  $x$  ທີ່ກຳນົດ ຈະຖືກຄໍານວນ ແລະ ພິມອອກມາ.
- ▶ ບັນຫານີ້ເກີດຂຶ້ນໃນກຳລະນີທີ່ (1, 2, 1) ແລະ (1, 1, 2) ແລະ (2, 1, 1) ຖື່ວ່າຄືກັນ. ດັ່ງນັ້ນ, ມັນເປັນບັນຫາທີ່ບໍ່ສາມາດສະແດງອອກດ້ວຍໂຄງສ້າງ list, tuple, ຫຼື dictionary ໄດ້.
- ▶ ມັນສະດວກໃນການນຳໃຊ້ປະເພດຂໍ້ມູນທີ່ກຳນົດໄວ້ເພື່ອແກ້ໄຂພາລະກິດນີ້.

## 2. Mission

### 2.1. Monopoly ເຮັດວຽກແນວໃດ

```
1 cases_3times = set()
2 for i in range(1, 7):
3     for j in range(1, 7):
4         for k in range(1, 7):
5             cases_3times.add((i, j, k))
6
7 total_cases = len(cases_3times)
8 print('Event that can happen by rolling the dice 3 times are', total_cases, 'cases.')
```

ມີ 216 ກໍາລະນີທີ່ສາມາດເກີດຂຶ້ນໄດ້ຍາກນໂຍ່ນໝາກກະລອກສາມຄັ້ງ.

## 2. Mission

### 2.1. Monopoly ເຮັດວຽກແນວໃດ

```
1 for i in range(3, 19): # Find the probability of getting more than 3 to more than 18
2     total_cases = len(cases_3times)
3     n_cases = 0
4     for c in cases_3times:
5         if sum(c) >= i:
6             n_cases += 1
7     prob = n_cases * 100 / total_cases
8     print('probability of getting more than {:2d} is {:.2f}%'.format(i,prob))
```

probability of getting more than 3 100.00%  
probability of getting more than 4. 99.54%  
probability of getting more than 5. 98.15%  
probability of getting more than 6. 95.37%  
probability of getting more than 7. 90.74%  
probability of getting more than 8. 83.80%  
probability of getting more than 9. 74.07%  
probability of getting more than 10 62.50%  
probability of getting more than 11. 50.00%  
probability of getting more than 12. 37.50%  
probability of getting more than 13. 25.93%  
probability of getting more than 14. 16.20%  
probability of getting more than 15. 9.26%  
probability of getting more than 16. 4.63%  
probability of getting more than 17. 1.85%  
probability of getting more than 18. 0.46%

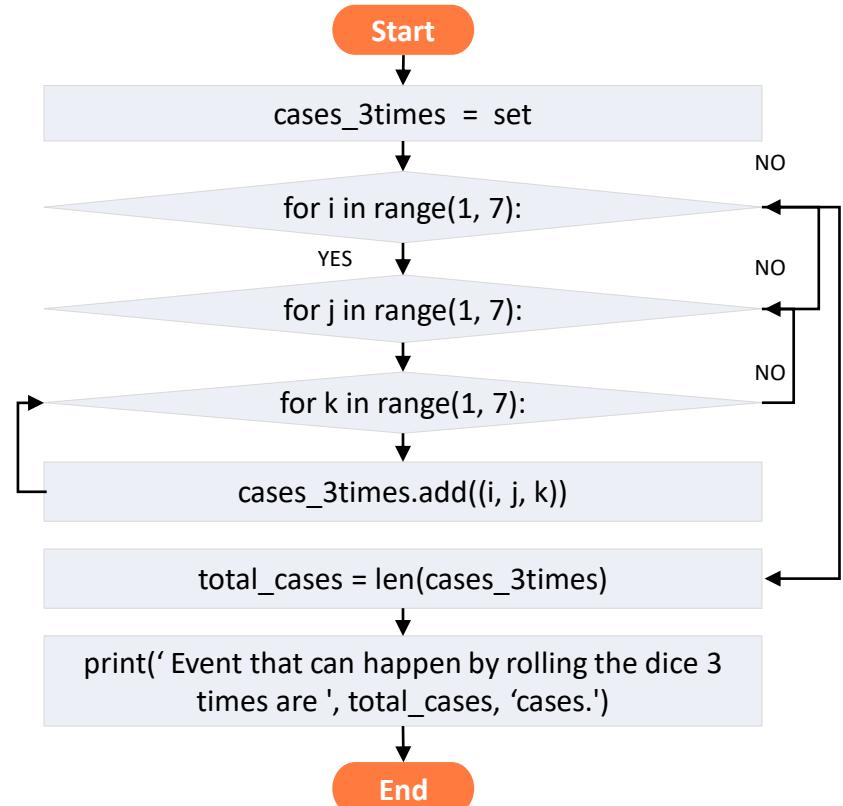
## 2. Mission

### 2.2. ແຜນການຂຽນໂປຣແກຣມ

#### Pseudocode

- [1] ເລີ່ມຕົ້ນ
- [2] ບະກາດຕົວປັບປຸງໝາກະລອກຫັງໝົດທີມີ.
- [3] for I to from1 to 7 do{
- [4]     for I to from1 to 7 do{
- [5]         for I to from1 to 7 do{
- [6]             ໃນກໍລະນີຂອງ dice, ບຸກສຶ່ງທຸກຍ່າງຍຸ່ນໃນຮູບ tuple.
- [7]             ຈຳນວນໃນ case\_3 ຄ້າທີ່ບັນທຶກໄວ້ໃນ total\_cases.
- [8]             ຈຳນວນກໍລະນີທີ່ກີມອອກ.
- }
- [9] ສັນສຸດ

#### Flowchart



## 2. Mission

### 2.2. ແຜນການຂຽນໂປຣແກນມ

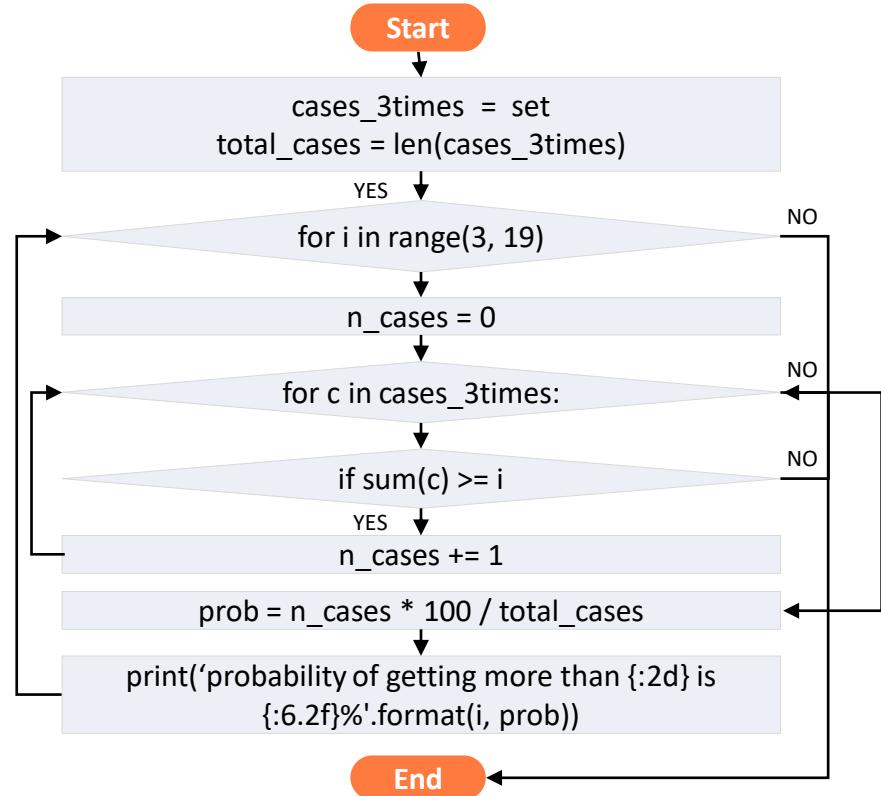
#### Pseudocode

```

[1] Start
[2] All dice cases are stored in cases_3times and the number of
    cases is stored in total_cases.
[3] for I to from 3 to 18 do{
[4]     Initialize n_cases, which accumulate and add more than a
        certain number of eyes, to zero.
[5]     for j to all number of cases do{
[6]         If the sum of the dice is greater than or equal to I,
            then{
[7]             add number of cases. }
[8]     (number of cases accumulated and added above) *
        100/total_cases are given to prob to obtain probability.
[9]     print probability. }
[10] End

```

#### Flowchart



## 2. Mission

### 2.3. ໄຄສະໜັບໜ້າຍ Monopoly #1

```
1 cases_3times = set()
2 for i in range(1, 7):
3     for j in range(1, 7):
4         for k in range(1, 7):
5             cases_3times.add((i, j, k))
6
7 total_cases = len(cases_3times)
8 print('Event that can happen by rolling the dice 3 times are' , total_cases, 'cases.')
```

## 2. Mission

### 2.3. . ໄຄສະໜັບໜ້າຍ Monopoly #2

```
1 for i in range(3, 19): # Find the probability of getting more than 3 to more than 18
2     total_cases = len(cases_3times)
3     n_cases = 0
4     for c in cases_3times:
5         if sum(c) >= i:
6             n_cases += 1
7     prob = n_cases * 100 / total_cases
8     print('probability of getting more than {:d} is {:.2f}%'.format(i,prob))
```

# បិទទី16. ປະយេណកូមខំមុន (Set Data Types)

## | Key concept

## 1. ການສ້າງກຸ່ມຂໍ້ມູນ

### 1.1. ຄໍານິຍາມຂອງກຸ່ມຂໍ້ມູນແລະ ວິທີການປະກາດ

| ປະເພດກຸ່ມຂໍ້ມູນທີ່ກໍານົດໄວ້ແມ່ນພື້ນຖານປະເພດຂໍ້ມູນທີ່ Python ກະກຽມໄວ້ ແຕ່ບໍ່ໄດ້ຖືກກະກຽມໄວ້ເປັນພື້ນຖານໃນພາສາ C ຫຼື Java.

- ▶ ກຸ່ມຂໍ້ມູນທີ່ກ່າວເຖິງໃນຄະນິດສາດແມ່ນການກຸ່ມຂອງອອບເຈັກທີ່ເປັນໄປຕາມກົດເການທີ່ຊັດເຈນຫຼືຄຸນລັກສະນະທີ່ກໍານົດ.
- ▶ ກຸ່ມຂໍ້ມູນໃນ Python ແມ່ນປະເພດຂໍ້ມູນທີ່ບໍ່ລົງລຳດັບ, ແລະຄ້າຍຄືກັນກັບວັດຈະນານຸ້າມ, ໃຊ້ວົງປຶກກາ {}.
- ▶ ບໍ່ອະນຸຍາດໃຫ້ຊື້ຄ່າໃນລາຍການທີ່ມີ.
- ▶ ມັນເປັນໄປໄດ້ທີ່ຈະດຳເນີນງານກັບກຸ່ມຂໍ້ມູນເຊັ່ນ: ການຕັດ, ການໂຮມ, ແລະຜົນຕ່າງຂອງກຸ່ມຂໍ້ມູນ.

| ວິທີຕ່າງໃນການປະກາດຊຸດຂໍ້ມູນ.

ການປະກາດກຸ່ມຂໍ້ມູນເປົ້າຫວ່າງ	<code>set0 = set()</code>
ການປະກາດກຸ່ມຂໍ້ມູນພື້ນຖານ	<code>set1 = {1, 2, 3, 4}</code>
ການປະກາດກຸ່ມຂໍ້ມູນຈາກ tuple	<code>n_tuple = (1, 2, 3, 4)</code> <code>set2 = set(n_tuple)</code>
ການປະກາດກຸ່ມຂໍ້ມູນຈາກ list	<code>n_list = [1, 2, 3, 4]</code> <code>set3 = set(n_list)</code>

## 1. ການສ້າງກຸມຂໍ້ມູນ

### 1.2. ວິທີການສ້າງກຸມຂໍ້ມູນ

- | ເວລາທີ່ສ້າງກຸມຂໍ້ມູນ, ເຮົາຍັງສາມາດໃຊ້ຟັງຊັ້ນ set ສໍາລັບລາຍການ ຫຼື tuple ຕ້ອງທີ່ສະແດງຂ້າງລຸ່ມນີ້.
- | ເນື່ອງຈາກກຸມຂໍ້ມູນແມ່ນການຮວບຮ່ວມຂອງອົງປະກອບທີ່ບໍ່ຢູ່ໃນລຳດັບ, ຈຶ່ງບໍ່ສາມາດໃຊ້ດັດສະນີໄດ້. ແຍະສະນັ້ນ, ການຕັດແມ່ນເປັນໄປບໍ່ໄດ້.

```

1 days_list = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun'] # list
2 days_set = set(days_list) # making sets from list
3 days_set

{'Fri', 'Mon', 'Sat', 'Sun', 'Thu', 'Tue', 'Wed'}
```

```

1 fruits_tuple = ('apple', 'orange', 'water melon') # tuple
2 fruits_set = set(fruits_tuple) # making sets from tuple
3 fruits_set

{'apple', 'orange', 'water melon'}
```

## 💡 ອີກບາດກ້າວໜຶ່ງ

- | ເມື່ອສ້າງກຸ່ມຂໍ້ມູນ, ເຮືອຍັງສາມາດໃຊ້ພັງຊັນ set. ແຕ່ລະອີງປະກອບສຸດທ້າຍຂອງກຸ່ມຂໍ້ມູນແມ່ນ {'h', 'l', 'e', ແລະ 'o'}
- | ເນື່ອງຈາກກຸ່ມຂໍ້ມູນບໍ່ອະນຸຍາດໃຫ້ຊຳຄ່າ, ຕົວອັກສອນ 'l' ຈາກ ຂໍ້ຄວາມ 'hello'. ຜົນຮັບຈະເໝືອນກັບ {'e', 'h', 'l', ແລະ 'o'}}.

```
1 h_str = 'hello'          # String
2 h_set = set(h_str)       # Making sets from string
3 h_set                         # Set does not allow duplication of string 'l'

{'e', 'h', 'l', 'o'}
```

## 2. ກວດສອບຄ່າໃນກຸ່ມຂໍ້ມູນ

### 2.1. ຕົວດຳເນີນການ `in`

| ເພື່ອກວດເບື່ງວ່າອີງປະກອບໄດ້ຢູ່ໃນກຸ່ມຂໍ້ມູນ, ເຮົາສາມາດໃຊ້ຕົວດຳເນີນການ `in` ຄືກັນກັບ `list`.

```
1 numbers = {2, 1, 3}          # check if 1 is in the numbers set
2 if 1 in numbers:
3     print("1 is in the set.")
```

1 is in the set.

## ຄືກບາດກ້າວໜຶ່ງ

| ຕົວດຳເນີນການ `in` ບໍ່ພຽງແຕ່ໃຊ້ສຳລັບກຸ່ມຂໍ້ມູນເທົ່ານັ້ນ

- ▶ ຕົວດຳເນີນການ `in` ທີ່ໃຊ້ໃນພາກນີ້ ບໍ່ແມ່ນຕົວດຳເນີນການທີ່ສາມາດໃຊ້ສຳລັບກຸ່ມຂໍ້ມູນເທົ່ານັ້ນ ແຕ່ສາມາດນຳໃຊ້ກັບຂໍ້ມູນຕ່າງໆທີ່ມີໝາຍອົງປະກອບໄດ້. ໃນຕົວຢ່າງ, ສາມາດກວດສອບໄດ້ວ່າມີອົງປະກອບໄດ້ໜຶ່ງໃນອົງປະກອບລາຍການດັ່ງທີ່ສະແດງຂາງລຸ່ມນີ້.

```
1 a_list = ['hello', 'world', 'welcome', 'to', 'python']  
2 'python' in a_list # ກວດສອບວ່າ 'python' ແມ່ນຢູ່ໃນ a_list
```

True

## 2. ກວດສອບຄ່າໃນກຸ່ມຂໍ້ມູນ

### 2.2. ດັດສະນີໃນກຸ່ມຂໍ້ມູນ

ສຶກສືກວນເອົາໃຈໄສ່

ເນື່ອງຈາກບໍ່ມີດັດສະນີໃນອົງປະກອບຂອງກຸ່ມຂໍ້ມູນ, ການດຳເນີນການສ້າງດັດສະນີ ຫຼືການແບ່ງສ່ວນແມ່ນບໍ່ມີຄວາມທາຍ.

| ເຮົາສາມາດເພີ່ມອົງປະກອບໂດຍການນຳໃຊ້ຟັງຊັນ add

```
1 numbers = [1, 2, 3]
2 numbers.add(4)
3 numbers
```

{1, 2, 3, 4}

| ພວກເຮົາຍັງສາມາດໃຊ້ຟັງຊັນ remove ເພື່ອລຶບອົງປະກອບໃນກຸ່ມຂໍ້ມູນ

```
1 numbers.remove(4)
2 numbers
```

{1, 2, 3}

### 3. ການນຳໃຊ້ Set

#### 3.1. ການສ້າງກຸ່ມຂໍ້ມູນ

- | ຍັງສາມາດສ້າງກຸ່ມຂໍ້ມູນຈາກລາຍການຕໍ່ໄປນີ້.
- | ຢ່າງໄດ້ກໍາຕາມ, ໃນປະເພດກຸ່ມຂໍ້ມູນທີ່ກໍານົດໄວ້, ເມື່ອອີງປະກອບຊ້າກັນ, ອົງປະກອບເຫຼົ່ານັ້ນຈະຖືກລຶບອອກໂດຍອັດຕະໂນມັດ.

```
1 set([1, 2, 3, 1, 2]) # Create set from list  
{1, 2, 3}
```

- | ແລະມັນເປັນໄປໄດ້ທີ່ຈະສ້າງກຸ່ມຂໍ້ມູນຈາກສະຕິງ.
- | ໃນກໍລະນີ້ນີ້, ແຕ່ລະຕົວອັກສອນກາຍເປັນອີງປະກອບ.

```
1 set("abcdefa")      # Since sting is also sequence type it is possible to convert to sets  
{'a', 'b', 'c', 'd', 'e', 'f'}
```

### 3. ການນຳໃຊ້ Set

#### 3.2. ວິທີການສ້າງກຸ່ມຂໍ້ມູນທວ່າງເປົ່າ

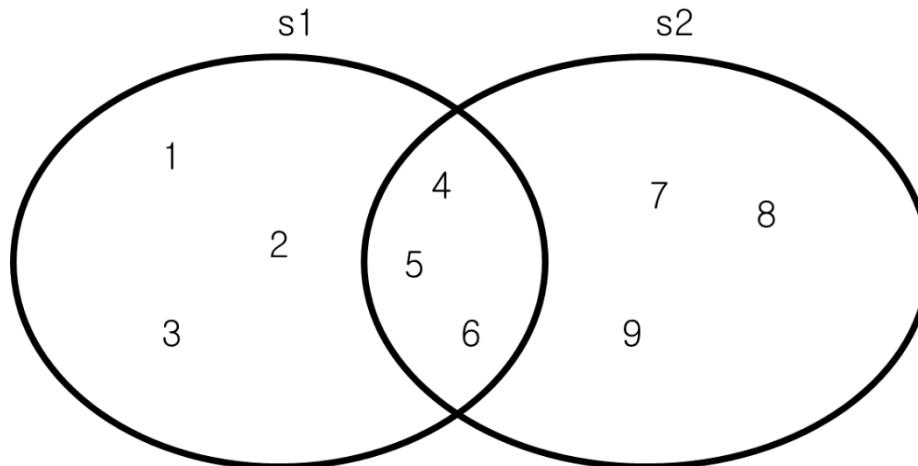
| ເພື່ອສ້າງກຸ່ມຂໍ້ມູນທວ່າງເປົ່າ, ໃຫ້ໃຊ້ຝັງຊັນ set ດັ່ງທີ່ສະແດງຕ້ອງລຸ່ມນີ້.

```
1 numbers = set()      # Create an empty set  
2 numbers  
set()
```

## 4. ການຄໍານວນ set

### 4.1. ຕົວດໍາເນີນການ Set ແລະ ພັງຊັນ

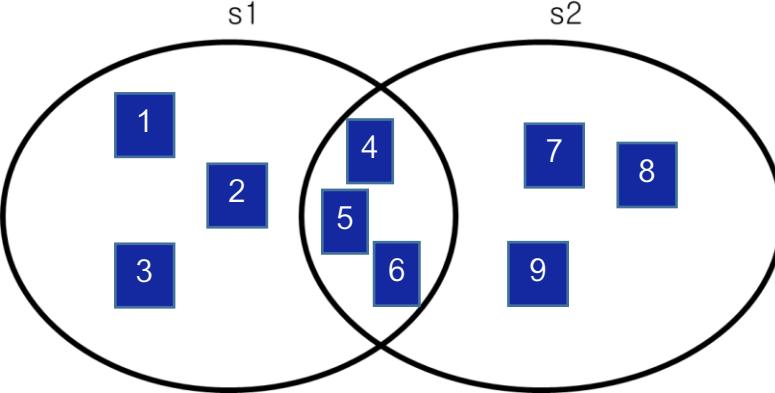
- | ໃຊ້ຄື່ອງໝາຍ & ສໍາລັບການຕັດກັນ, ເຄື່ອງໝາຍ | ສໍາລັບການໂຮມ, ເຄື່ອງໝາຍ — ສໍາລັບຜົນຕ່າງຂອງ set , ເຄື່ອງໝາຍ ^ ສໍາລັບຜົນລົບເຄິ່ງຄືຂອງກຸ່ມຂຶ້ນ.
- | ຖ້າມີກຸ່ມຂຶ້ນ s1 ແລະ s2, ເພື່ອທີ່ຈະດໍາເນີນງານຂອງຊູດຫຼັງນີ້, ພວກເຮົາຈໍາເປັນຕ້ອງສະແດງຄວາມສໍາພັນລວມກັນຂອງອີງປະກອບຂອງກຸ່ມຂຶ້ນໃນແຜນວາດ.



## 4. ການຄໍານວນ set

### 4.2. ການຄໍານວນການໂຮມ

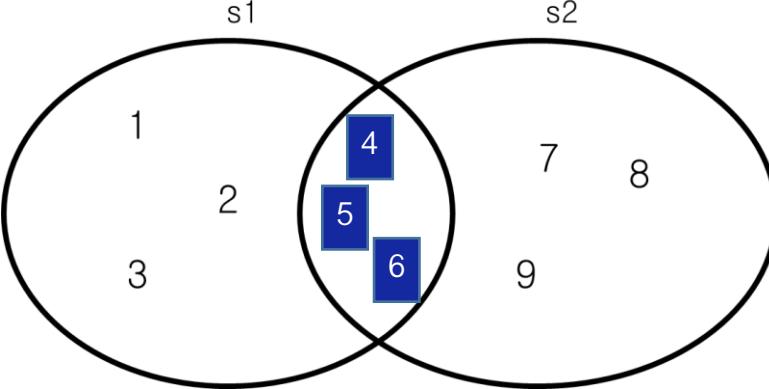
| ໃຊ້ເຕື່ອງໝາຍ | ຫຼື union ສໍາລັບການຄໍານວນການໂຮມ

ຄໍານວນ	ຜົນຮັບ
union	 $s1 \cup s2 = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$

## 4. ການຄໍານວນ set

### 4.3. ການຄໍານວນການຕັດ

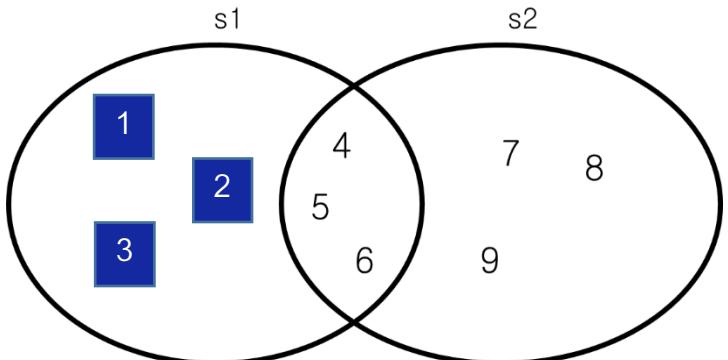
| ໃຊ້ເຕື່ອງໝາຍ & ຫຼື intersection ສໍາລັບການຄໍານວນການຕັດກັນ

ຄໍານວນ	ຜົນຮັບ
intersection	 $s1 \& s2 = \{4, 5, 6\} \text{ ຫຼື } s1.intersection(s2)$

## 4. ການຄໍານວນ set

### 4.4. ການຄໍານວນຜົນລົບຂອງ set

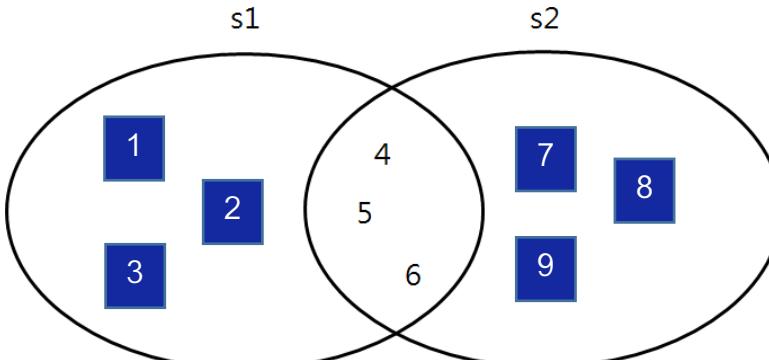
| ໃຊ້ເຕື່ອງໝາຍ – ດຳເນີນງານ ຫຼື difference ຄໍານວນຜົນລົບຂອງ set.

ຄໍານວນ	ຜົນຮັບ
Difference of sets	 $s1 - s2 = \{1, 2, 3\} \text{ ຫຼື } s1.difference(s2)$

## 4. . ການຄໍານວນ set

### 4.5. ການຄໍານວນຜົນລົບເຄື່ອງຄືຂອງ set

| ຜົນລົບເຄື່ອງຄື ແມ່ນການລົບຂອງການຕັດກັນຂອງ set ,  $(s1 \cup s2) - (s1 \cap s2)$ . ໃຊ້ເຄື່ອງໝາຍ ^ ຫຼື symmetric\_difference ເພື່ອຄໍານວນ.

ຄໍານວນ	ຜົນຮັບ
Symmetric difference of sets	 $s1 \Delta s2 = \{1, 2, 3, 7, 8, 9\}$ ຫຼື <code>s1.symmetric_difference(s2)</code>

## 4. . ການຄໍານວນ set

## 4.5. ການຄໍານວນຜົນລົບເຄື່ອງຄືຂອງ set

```
1 s1 = {1, 2, 3, 4, 5, 6}
```

```
1 s2 = {4, 5, 6, 7, 8, 9}
```

```
1 s1 | s2      # Find union
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

```
1 s1 & s2      # Find intersection
```

```
{4, 5, 6}
```

```
1 s1 - s2      # Find difference of set
```

```
{1, 2, 3}
```

```
1 s1 ^ s2      # Find symmetric difference of set
```

```
{1, 2, 3, 7, 8, 9}
```

## 4. ການຄໍານວນ set

### 4.6. ພັງຊັນຫລາຍພັງຊັນທີໃຊ້ໃນ set

ສິ່ງທີ່ຄວນເອົາໃຈໃໝ່

ໃນ Set ຕົວດໍາເນີນການ |, &, -, ^ ຈະໃຫ້ຜົນຮັບດຽວກັນກັບໃນໃຊ້ union, intersection, difference, symmetric\_difference ດັ່ງຕົວຢ່າງລຸ່ມນີ້

```
1 s1 = {1, 2, 3, 4, 5, 6}
```

```
1 s2 = {4, 5, 6, 7, 8, 9}
```

```
1 s1.union(s2)      # Find union
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

```
1 s1.intersection(s2)  # Find intersection
```

```
{4, 5, 6}
```

```
1 s1.difference(s2)      # Find difference of set
```

```
{1, 2, 3}
```

```
1 s1.symmetric_difference(s2)  # Find symmetric difference of set
```

```
{1, 2, 3, 7, 8, 9}
```

## 4. ການຄໍານວນ set

## 4.7. ການຄໍານວນຢ່າງຕໍ່ເນື້ອງໄດ້ເຊັ່ນດູວກັນ

```
1 s1 = {1, 2, 3, 4, 5, 6}
```

```
1 s2 = {4, 5, 6, 7, 8, 9}
```

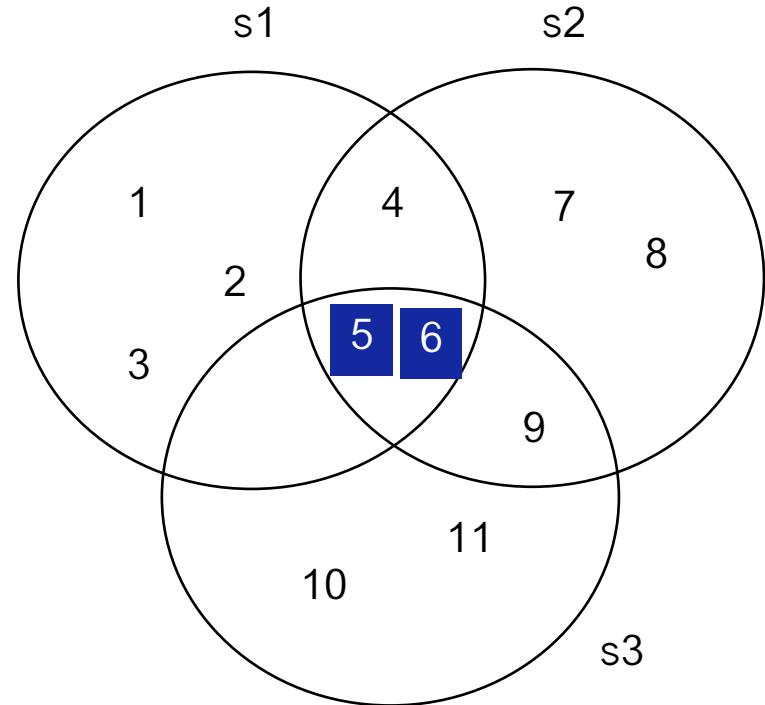
```
1 s3 = {5, 6, 9, 10, 11}
```

```
1 s1 & s2 & s3 # Find union of s1, s2, s3
```

```
{5, 6}
```

```
1 S1 - s2 - s3 # subtract elements of s2 and s3 from s1
```

```
{1, 2, 3}
```



$$s1 \& s2 \& s3 = \{ 5, 6 \}$$

## 4. ການຄໍານວນ set

### 4.8. ພັງຊັນທີ່ກ່ຽວຂ້ອງກັບ set

ພັງຊັນ	ບັນຫຼິກ
add(x)	ເພີ່ມອີງປະກອບ x ໃສ່ set.
discard(x)	ລຶບອີງປະກອບ x ຈາກ set.
clear	ລຶບອີງປະກອບທັງໝົດໃນ set.
union(s)	ຊອກຫາການໂຮມໃນ set s, ຕີກັນກັບການໃຊ້ເຕືອງໝາຍ
difference(s)	ຊອກຫາຜົນຕ່າງໃນ set s, ຕີກັນກັບການໃຊ້ເຕືອງໝາຍ –
intersection(s)	ຊອກຫາການຕັດໃນ set s, ຕີກັນກັບການໃຊ້ເຕືອງໝາຍ &
symmetric_difference(s)	ຊອກຫາຜົນລົບເຕິ່ງຕີໃນ set s, ຕີກັນກັບການໃຊ້ເຕືອງໝາຍ ^
issubset(s)	ຊອກຫາ id set s ເປັນຊຸດຍ່ອຍ, ສົ່ງຄືນຄ່າ True/False.
issuperset(s)	ຊອກຫາ id set s ເປັນ superset ສົ່ງຄືນຄ່າ True/False.
isdisjoint(s)	ຊອກຫາ id set s ເປັນ coprime ສົ່ງຄືນຄ່າ True/False.

## 4. ການຄໍານວນ set

### 4.9. ການຄໍານວນ subset ແລະ superset

| Subset: ຖ້າອີງປະກອບທັງໝົດຂອງ set A ເປັນຂອງ set B, ດັ່ງນັ້ນ A ແມ່ນ Subset ຂອງ B. ຕົວຢ່າງ: A = {1, 2}, B = {1, 2, 3} A ແມ່ນ Subset ຂອງ B.

```
1 s1 = {1, 2, 3, 4, 5}
2 s2 = {1, 2, 3}      # is s1's subset
3 s3 = {1, 2, 6}      # not s1's subset
```

```
1 s2.issubset(s1)    # method asking if s2 is subset of s1
```

True

```
1 s3.issubset(s1)    # method asking if s3 is subset of s1
```

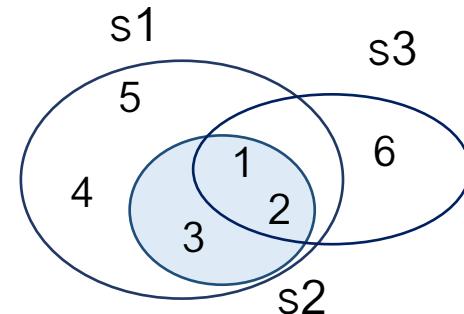
False

```
1 s1.issuperset(s2)  # method asking if s1 is superset of s2
```

True

```
1 s1.issuperset(s3)  # method asking if s1 is superset of s3
```

False



s2 ແມ່ນ subset ຂອງ s1  
s3 ແມ່ນ subset ຂອງ s1

## 4. ການຄໍານວນ set

## 4.10. ຕິວຢ່າງການຄໍານວນຂອງ coprime

Ex ການຄໍານວນ Coprime ຂອງ 2 sets

```
1 s1 = {1, 2, 3}
2 s2 = {10, 20, 30}
3 s1.isdisjoint(s2) # checking if s1 and s2 are coprime
```

True

Line 3

- ສິ່ງຄືນຄ່າ True ເນື່ອງຈາກ 2 sets ເປັນ coprime.

## 💡 ອີກໜຶ່ງຂັ້ນຕອນ

- | ຜົນຄຸນກຸ່ມ (Product set) ຫຼື ຜົນຄຸນກາດເຕຊຽນ (Product Cartesian)
- | ມີແນວຄວາມຄິດທີ່ເອີ້ນວ່າ ຜົນຄຸນກຸ່ມ (Product set) ຫຼື ຜົນຄຸນກາດເຕຊຽນ (Product Cartesian)
  - ▶ Product set ສາມາດຂຽນເປັນ Product Cartesian  $A \times B$
  - ▶ ມັນກາຍເປັນຊູດຂອງ tuples  $(a, b)$  ທີ່ສາມາດສ້າງຂຶ້ນໄດ້ໂດຍໃຊ້ອີງປະກອບທັງໝົດ  $a$  ໃນຊູດ  $A$  ແລະທຸກອີງປະກອບ  $b$  ໃນຊູດ  $B$ , ເຊັ້ນດຽວກັນ  $\{(a, b) | a \in A, b \in B\}$

$$\begin{array}{c}
 & & \text{AxB} \\
 \begin{array}{c} \text{A} \\ \boxed{1} \\ \boxed{3} \end{array} & \times & \begin{array}{c} \text{B} \\ \boxed{2} \\ \boxed{4} \end{array} = \boxed{\begin{array}{l} (1, 2) \\ (1, 4) \\ (3, 2) \\ (3, 4) \end{array}}
 \end{array}$$

	2	4	Set B
1	(1, 2)	(1, 4)	
3	(3, 2)	(3, 4)	

Set A

## 💡 ອີກໜຶ່ງຂັ້ນຕອນ

| Product set ຂອງ product sets A, B

```

1 A = {1, 3}      # element of set A
2 B = {2, 4}      # element of set B
3 res = set()
4 for i in A:
5     for j in B:
6         res = res | {(i, j)} # product set with double for loop
7 AxB = res # AxB # product set AxB of A and B(not A*B)
8 print('A =', A)
9 print('B =', B)
10 print('AxB =', AxB)       # print product set of A and B

```

A = {1, 3}  
B = {2, 4}  
AxB = {(3, 2), (1, 2), (3, 4), (1, 4)}

### Line 4-7

- ໂຄດຂ້າງເທິງແມ່ນການໂຮມຂອງອົງປະກອບຂອງ (i, j) ກັບ res ໂດຍໃຊ້ loop ຊ້ອນ loop.
- ເນື່ອງຈາກ set ບໍ່ອະນຸຍາດໃຫ້ຊ້າກັນ, ອົງປະກອບທີ່ຊ້າກັນຈະຖືກລືບອອກຈາກການໂຮມ.
- tuples ແມ່ນລະອົງປະກອບບໍ່ໄດ້ຊ້າກັນ.
- ໃນຕົວປ່ຽນ AxB, x ບໍ່ແມ່ນຕົວດໍາເນີນການຄຸນ

## 5. ການປຽບທຽບ Set

### 5.1. ການຄໍານວນການປຽບທຽບ Set

- | ເຮົາສາມາດກວດໄດ້ວ່າທັງສອງ set ຄືກັນຫຼືບໍ່.
- | ໂດຍການນຳໃຊ້ຕົວດໍາເນີນການ == ແລະ != ເປັນວິທີທີ່ຈ່າຍທີ່ສຸດ.

```

1 A = {1, 2, 3}
2 B = {1, 2, 3}
3 A == B

```

True

- | ການນຳໃຊ້ຕົວດໍາເນີນການ < ແລະ <= , ສາມາດກວດສອບເບິ່ງວ່າ set ແມ່ນເປັນຊຸດຍ່ອຍທີ່ແທ້ຈີງ ຫຼືຊຸດຍ່ອຍ.
- | ຊຸດຍ່ອຍຈີງໝາຍເຖິງຊຸດຍ່ອຍທີ່ແທ້ຈີງຂອງຊຸດຍ່ອຍທັງໝົດຢືນເວັ້ນເວົ້າຊຸດຍ່ອຍຂອງຕົນເອງ.
- | ດັ່ງທີ່ສະແດງຢູ່ໃນໂຄດຂ້າງລຸ່ມນີ້, B ແມ່ນຊຸດຍ່ອຍຂອງ A, ແລະ B ບໍ່ເທົ່າກັບ A. ໃນເວລານີ້, B ເອັນວ່າຊຸດຍ່ອຍທີ່ແທ້ຈີງ.

```

1 A = {1, 2, 3, 4, 5}
2 B = {1, 2, 3}
3 B < A

```

True

## 6. ຄໍາສັ່ງປະຕິບັດຊຳ ແລະ Sets

### 6.1. ວິທີການໃຊ້ຄໍາສັ່ງປະຕິບັດຊຳ

- | ເນື່ອງຈາກ set ເປັນອອບເຈັກທີ່ເຮັດຊຳຄືນໄດ້, ຈຶ່ງສາມາດໃຊ້ຄໍາສັ່ງການປະຕິບັດຊຳໄດ້.
- | ແຕ່ລະອົງປະກອບໃນກຸ່ມສາມາດເຂົ້າເຖິງແລະພິມອອກໄດຍໃຊ້ loop for ດັ່ງຕໍ່ໄປນີ້.

```
1 numbers = {2, 1, 3}
2 for x in numbers:
3     print(x, end=" ")
```

1 2 3

## 6. ຄໍາສັ່ງປະຕິບັດຊື້າ ແລະ Sets

### 6.2. ລຳດັບ set

ສຶກຫິຄວາມເອົາໃຈໃສ່

set ອາດຈະແຕກຕ່າງຈາກລຳດັບການປ້ອນຂໍ້ມູນເພາະວ່າມັນບໍ່ມີລຳດັບ.

| ຖ້າເຮົາຕ້ອງການພິມອີງປະກອບໃນກຸ່ມຕາມລຳດັບ, ເຮົາສາມາດນຳໃຊ້ຝັງຊັນການລຽງລຳດັບຕາມຮູບຂ້າງລຸ່ມນີ້.

```
1 for x in sorted(numbers):  
2     print(x, end=" ")
```

1 2 3

## 7. ການນຳໃຊ້ຄຳສັ່ງເງື່ອນໄຂໃນ Set

### 7.1. ການໃຊ້ສໍາລັບຄຳສັ່ງ for ໃນລຽງລຳດັບ

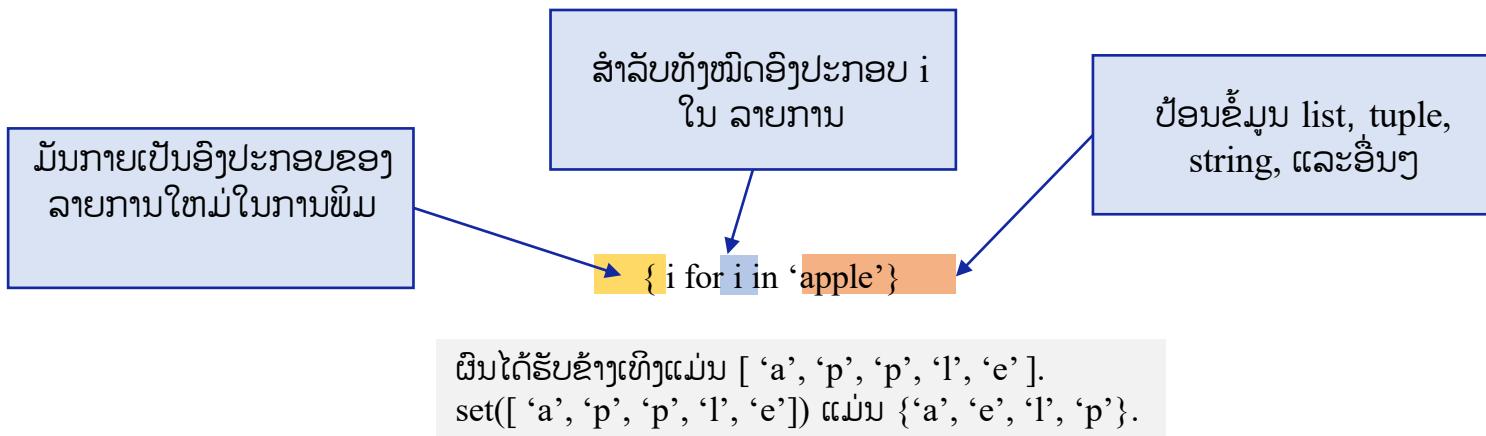
| set ສາມາດຖືກສ້າງຂຶ້ນໂດຍໃຊ້ Loop ແລະ ລຽງລຳດັບ ດັ່ງຕໍ່ໄປນີ້.

```
1 | a = {i for i in 'apple'}
```

```
1 | a
```

```
{'a', 'e', 'l', 'p'}
```

| ປຽບທຽບຜົນໄດ້ຮັບກັບລາຍການລຽງລຳດັບດັ່ງຕໍ່ໄປນີ້:



## 7. ການນຳໃຊ້ຄຳສັ່ງເງື່ອນໄຂໃນ Set

### 7.2. ຕົວຢ່າງຂອງການລຽງລຳດັບຂອງ set

- | ນຳໃຊ້ການລຽງລຳດັບ, ເຮົາຈະໄດ້ຜົນຮັບທີ່ຕ້ອງການດ້ວຍຂຽນໂຄດທີ່ສັນ.
- | ດັ່ງທີ່ສະແດງຢູ່ໃນຄຳສັ່ງດ້ານລຸ່ມນີ້, ຄຳສັ່ງແບບມີເງື່ອນໄຂໃນການລຽງລຳດັບຖືກລະບຸໄວ້ຫຼັງຈາກ for loop.
- | {ສົມຜົນຂອງຕົວປ່ຽນໃນ set ຄຳສັ່ງແບບມີເງື່ອນໄຂ }

```

1 a = {i for i in 'pineapple' if i not in 'apl'}
2 a
['e', 'i', 'n']

```

#### Line 1

- ການນຳໃຊ້ for, ແຕ່ລະຕົວອັກສອນຂອງ 'pineapple' ຖືກກຳນົດໃຫ້ກັບ i, ເຊິ່ງຖືກກຳນົດຕອງໂດຍເງື່ອນໄຂຂອງຄຳສັ່ງ if.
- ໃນໂຄດຂ້າງເທິງ, ຖ້າບໍ່ກົງກັນກັບຕົວອັກສອນ 'a', 'p', ແລະ 'l', ດັ່ງນັ້ນ 'e', 'T', ແລະ 'n' ຈະຖືກເກັບໄວ້ໃນ a.
- ໂຄດນີ້ແມ່ນວິທີການແບກສະຕຣິໂດຍບໍ່ລວມເອົາຕົວອັກສອນ 'a', 'p' ແລະ 'l' ຈາກຂໍ້ຄວາມ 'pineapple'.

# Paper coding

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

**Q1.** ໃຊြပ်ချမှု SET ဖော်လာရန် အသေးစိတ်လိုအပ်သူများမှာ s1 အကြောင်းအရာများကို ပေါ်လုပ်ပါ။

ပြန်လည်  
ပြန်လည်

```
lst = ['apple', 'mango', 'banana']      # A list of 3 fruit information  
s1 = {'apple', 'mango', 'banana'}       # Set s1 generated from lst
```

Time

7min

ပြန်လည်

```
s1 = {'banana', 'apple', 'mango'}
```



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

**Q2.** ຂຽນໄສດ້ຜົນການຄໍານວນສໍາລັບສອງ set ຕໍ່ໄປນີ້. ຊອກຫາຜົນໄດ້ຮັບຈາກ 1) ຫາ 7).

ໂປຣແກຣມຕົວ  
ປ່ຽນເງື່ອນໄຂ

```
s1 = {10, 20, 30, 40}
s2 = {30, 40, 50, 60, 70}
1) s1 | s2
2) s1 & s2
3) s1 - s2
4) s1 ^ s2
5) s1.issubset(s2)
6) s1.issuperset(s2)
7) s1.isdisjoint(s2)
```

ເວລາ

5 ນາທີ



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

## ပါဒီ16. ပုံဖောက်များ

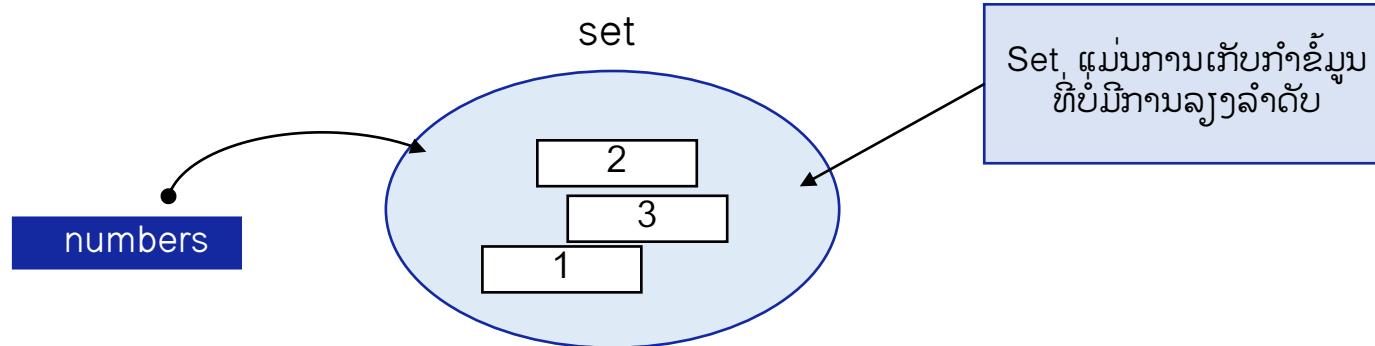
| Let's code

## 1. ຖ້າການລຽງລໍາດັບບໍ່ສໍາຄັນ: Set

### 1.1. ຄໍານິຍາມຂອງກຸ່ມຂໍ້ມູນ

| Set ທີ່ໃຊ້ໃນ Python ແມ່ນຄ້າຍຄືກັນກັບ Set ຂອງຄະນິດສາດ, ແລະແຕກຕ່າງຈາກ tuples ທີ່ປະເພດຂໍ້ມູນທີ່ບໍ່ມີລໍາດັບ. ໂດຍສະເພາະ, ບໍ່ອະນຸຍາດໃຫ້ມີຄ່າຂໍ້ກັນຂອງອົງປະກອບ ແລະ ສາມາດດຳເນີນການໃນຊຸດຂໍ້ມູນຕ່າງໆເຊັ່ນ: ຕັດກັນ, ການໂຮມ, ເຟັນລົບ ແລະຜົນລົບເຄິ່ງຄື ສາມາດປະຕິບັດໄດ້.

```
1 numbers = {2, 1, 3}    # set data type consisted of 3 numbers  
2 numbers  
{1, 2, 3}
```



## 1. ຖ້າການລຽງລໍາດັບບໍ່ສໍາຄັນ(If the Order is Not Important) : Set

### 1.2. ພັງຊັນຕ່າງໆທີ່ສາມາດຖືກນຳໃຊ້ກັບ set

- | ສາມາດໃຊ້ພັງຊັນຈໍານວນຫຼາຍເພື່ອປະມວນຜົນຂໍ້ມູນກ່ຽວກັບອີງປະກອບຕ່າງໆໃນກຸ່ມ.
- | ພັງຊັນເຊັນ: len, max, min, stored, sum, etc. ອາດຖືກໃຊ້ສໍາລັບ set ເຊັນກັນ.
- | ສ້າງກຸ່ມຂໍ້ມູນ ຫິກລາຍການດັ່ງທີ່ສະແດງຂ້າງລຸ່ມນີ້. ນຳໃຊ້ພັງຊັນຕ່າງໆ.

```
1 a_set = {1, 5, 4, 3, 7, 4}      # make set of six items
2 len(a_set)                      # number of item is 5 excluding duplication
```

5

```
1 max(a_set)                     # largest number within the item is 7
```

7

```
1 min(a_set)                     # smallest number within the item is 1
```

1

```
1 sorted(a_set)                  # make list by arranging the items, without duplication
```

[1, 3, 4, 5, 7]

```
1 sum(a_set)                     # since duplicated number is only used once, sum is 20
```

20

## 1. ຖ້າການລຽງລໍາດັບບໍ່ສໍາຄັນ(If the Order is Not Important) : Set

### 1.3. ພັງຊັນການຄໍານວນທາງຕັກກະສາດກັບ set

- | ການຄໍານວນການທາງຕັກກະສາດທີ່ສາມາດນຳໄປໃຊ້ກັບຂໍ້ມູນທີ່ມີຫຼາຍອີງປະກອບໄດ້ລວມເຖິງພັງຊັນທັງໝົດແລະພັງຊັນໃດໜຶ່ງ.
- | ພັງຊັນນີ້ສິ່ງຄືນຜົນຮັບຂອງການປະເມີນ boolean ສໍາເລັບແຕ່ລະອີງປະກອບຂອງຂໍ້ມູນທີ່ກຳນົດໃຫ້.
- | ສ້າງວ່າມີຂໍ້ມູນທີ່ກາລາຍການດັ່ງທີ່ສະແດງຂ້າງລຸ່ມນີ້. ນໍາໃຊ້ພັງຊັນຕ່າງໆ.

```
1 a_set = { 1, 0, 2, 3, 3}
2 all(a_set), any(a_set)    # Is a_set all true? Is there - in a_set? check
```

(False, True)

- | ປະເມີນທັງໝົດແຕ່ລະອີງປະກອບຂໍ້ມູນທີ່ເຮັດຊ້າໄດ້ພາຍໃນປະເພດ boolean ແລະສິ່ງຄືນຄ່າເປັນ true ຖ້າທຸກຢ່າງເປັນ true
 ▶ ຖ້າຫາກອັນໄດ້ອັນໜຶ່ງເປັນ false, ມັນຈະສິ່ງຄືນຄ່າເປັນ false.
- | ປະເມີນຢ່າງໃດຢ່າງໜຶ່ງຂອງແຕ່ລະອີງປະກອບຂໍ້ມູນທີ່ເຮັດຊ້າໄດ້ພາຍໃນປະເພດ boolean ແລະສິ່ງຄືນຄ່າເປັນ true ໂດຍບໍ່ຄໍານິງເຖິງຄ່າອື່ນຫາກອັນໄດ້ອັນໜຶ່ງເປັນ true
 ▶ ຖ້າທຸກຢ່າງເປັນ false, ສິ່ງຄືນຄ່າ false

## 1. ຖ້າການລຽງລໍາດັບບໍ່ສໍາຄັນ(If the Order is Not Important) : Set

### 1.4. ການຫານຂາດ (aliquot) ແລະ ການຄົ້ນຄິດຂຽນໂປຣແກຣມ

- | ລອງຄິດເບິ່ງວ່າຈະຫາການຫານຂາດຂອງຈໍານວນທຸວນສອງຈໍານວນໄດ້ແນວໃດ.
- | ຫານຂາດຂອງ 10 ສາມາດເປັນ 1, 2, 5, ແລະ 10, ເຊິ່ງ 1, 2 ແລະ 5 ບໍ່ລວມເອົາ 10 ຂອງຕົນເອງເຮັ້ນວ່າ ການຫານຂາດ.
- | ສ້າງອີງປະກອບລາຍການດັ່ງຕໍ່ໄປນີ້, ແລະຫານສືບດ້ວຍສອງ, ສາມ, ສີ, ຫ້າ ... ຫານຕົວເລກຫັງທີມິດເຖິງ 9 ແລະກວດເບິ່ງວ່າຕົວເສດມີຄ່າແມ່ນສູນຫຼືບໍ່, ແລະ ຖ້າແມ່ນ ດັ່ງນັ້ນ, ໃຫ້ເກັບໄວ້ໃນລາຍການ

```

1 num = 10
2 divisors = []
3
4 for i in range(2, num):
5     if num % i == 0:
6         divisors.append(i)
7
8 print(num,'s true aliquot : ', divisors)
```

10's true aliquot : [2,5]

#### Line 4, 5, 6

- i ເພີ່ມຂຶ້ນເທື່ອລະ 1 ຈາກ 2 ແລະ ວິນລຸບ ເຖິງ 9. ເຫດຜົນການເລີ່ມຕົ້ນດ້ວຍ 2 ແມ່ນເພື່ອໃຫ້ໄດ້ການຫານຂາດຖືກຕ້ອງ.
- ຫານ num ດ້ວຍ i ແລະ ເພີ່ມລົງໃນລາຍການ, ເນື່ອງຈາກເປັນຕົວຫານທີ່ສ່ວນເສດທີ່ຍັງເຫຼືອເປັນສູນ.
- ເຮົາສາມາດເບິ່ງການຫານຂາດຕົວເລກ 2 ແລະ 5 ຂອງ 10

## 1. ຖ້າການລຽງລໍາດັບບໍ່ສໍາຄັນ(If the Order is Not Important) : Set

### 1.4. ການຫານຂາດ(aliquot) ແລະ ການຄົ້ນຄິດຂຽນໂປຣແກຣມ

- | ນໍາໃຊ້ໂຄດນີ້ເພື່ອຊອກຫາຕົວຫານຮ່ວມສູງສຸດຂອງ 48 ແລະ 60, ໃຫ້ຊອກຫາຕົວຫານຮ່ວມຂອງ 48 ແລະ 60, ຫຼັງຈາກນັ້ນຊອກຫາຄ່າທີ່ໃຫຍ່ທີ່ສຸດ.
- | ດ້ວຍເຫດຜົນນີ້, ໃຫ້ປະກາດຊຸດຂໍ້ມູນທີ່ຫວ່າງເປົ້າ divisors1 ແລະ divisors2 ຈາກນັ້ນເພີ່ມຄ່າໃຫ້ກັບກຸ່ມຂໍ້ມູນນີ້ໂດຍທີ່ num % i ເກົ່າກັບສູນ.

```
1 num1 = 48
2 divisors1 = set()
3
4 for i in range(2, num1):
5     if num1 % i == 0:
6         divisors1.add(i)
7 print(num1,'s true aliquot : ', divisors1)
8
9 num2 = 60
10 divisors2 = set()
11
12 for i in range(2, num2):
13     if num2 % i == 0:
14         divisors2.add(i)
15 print(num2,'s true aliquot : ', divisors2)
```

48's true aliquot : {2,3,4,6,8,12,16,24}  
60's true aliquot : {2,3,4,5,6,10,12,15,20,30}

## 1. ຖ້າການລຽງລໍາດັບບໍ່ສໍາຄັນ(If the Order is Not Important) : Set

### 1.4. ການຫານຂາດ(aliquot) ແລະ ການຄົ້ນຄິດຂຽນໂປຣແກຣມ

- | ຕອນນີ້ໃຫ້ຊອກຫາຕົວຫານຮ່ວມໃນບັນດາຕົວຫານເຫຼື່ອນີ້.
- | ແລະ ໃຊ້ຝັງຊັນ max ເພື່ອຊອກຫາຄ່າທີ່ໃຫຍ່ທີ່ສຸດຂອງຄ່າເຫຼື່ອນີ້.
- | ແນ່ນອນ, ມີວິທີການຂອງ Euclid ໃນການຊອກຫາຕົວຫານໃຫຍ່ສຸດສອງຕົວໄດ້ໄວກວ່າວິທີການນີ້, ແຕ່ມັນຈະເປັນວິທີການຄິດໃນທາງໂປຣແກຣມເພື່ອຊອກຫາ ວິທີແກ້ໄຂໂດຍຜ່ານຂະບວນການແຕ່ລະຂັ້ນຕອນໃນການແກ້ໄຂບັນຫາເຂັ້ນນີ້.

```
1 print(divisors1.intersection(divisors2))
2 Print(num1, num2,'s maximum divisor : ', max(divisors1.intersection(divisors2)))
```

{2, 3, 4, 6, 12}

48 60's maximum divisor : 12

#### Line 2

- ຊອກຫາການຕັດກັນຂອງອົງປະກອບ A ແລະ B ແລະຫຼັງຈາກນັ້ນພິມຄ່າທີ່ໃຫຍ່ທີ່ສຸດໂດຍໃຊ້ຝັງຊັນ max .
- ຕົວຫານຮ່ວມໃຫຍ່ສຸດແມ່ນ 12 ຂອງ 40 ແລະ 60 ຮຶກພິມອອກ.

## 2. ການລວມໂດຍໃຊ້ຟັງຊັນ Zip(Aggregation Using Zip Function)

### 2.1. ຄຸນລັກສະນະຂອງຟັງຊັນ zip ແລະ ປະເພດຂໍ້ມູນທີ່ຊັ້າຄ່າໄດ້

- | ປະເພດຂໍ້ມູນເຊັ່ນ: lists, dictionaries, sets, ແລະ tuples ເອັນວ່າປະເພດຂໍ້ມູນທີ່ສາມາດປະຕິບັດຊ້າໄດ້.
- | ເມື່ອປະເພດຂໍ້ມູນທີ່ສາມາດປະຕິບັດຊ້າຖືກປ້ອນ, ພັງຊັນວິນລຸບສົ່ງຄືນ tuple ໂດຍມີການລວມເຂົ້າກັນດ້ວຍຟັງຊັນ zip.
- | ພັງຊັນ zip ສາມາດຮັບຂໍ້ມູນປະເພດປະຕິບັດຊ້າ ດັ່ງທີ່ສະແດງຢູ່ລຸ່ມນີ້. ອັນນີ້ເອັນວ່າການລວມ(Aggregation).

```
zip(*iterables)
```

## 2. ການລວມໂດຍໃຊ້ຟັງຊັນ Zip(Aggregation Using Zip Function)

### 2.2. ຕົວຢ່າງ code ຄໍາສັ່ງທີ່ໃຊ້ຟັງຊັນ zip

```
1 empty_iterator = zip()  
2 result = set(empty_iterator)  
3 result  
  
set()
```

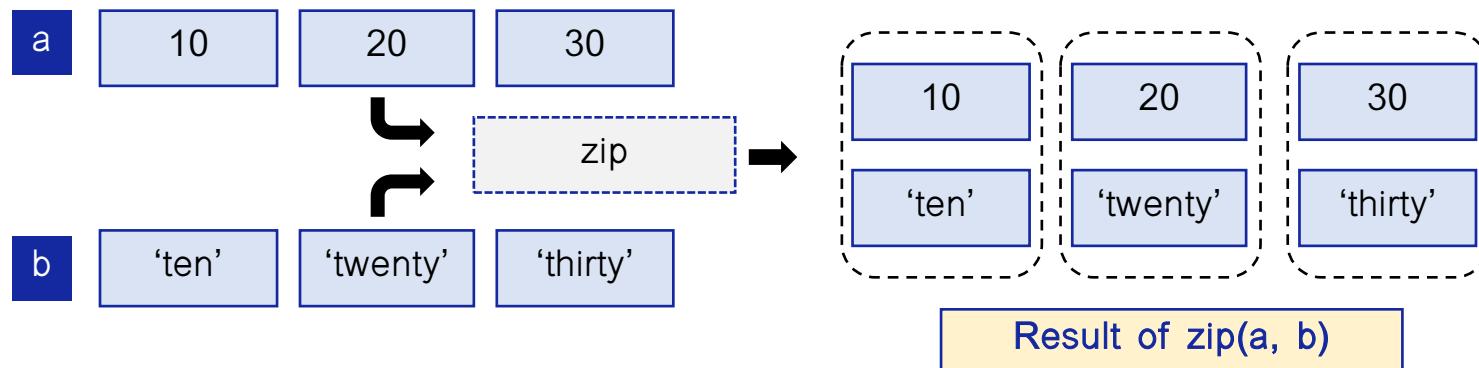
- | ພັງຊັນ zip ຈະສິ່ງຄືນຄ່າຫາວ່າງເປົ່າ ລ້າບໍ່ສິງຄ່າໃຫ້ກັບມັນ.
- | ພັງຊັນ zip ຈະວິນລຸບສິ່ງຄືນຄ່າສໍາລັບທີ່ຈະ tuple ເມື່ອມີການສິ່ງຄ່າໃຫ້ກັບມັນ.,
- | ແຕ່ລະອົງປະກອບຂອງ tuple ນີ້ຖືກສິ່ງທີ່ນີ້ຄ່າທີ່ໄດ້ຮັບຈາກຂັ້ມູນທີ່ມີການສິ່ງມາ
- | ເມື່ອປ້ອນປະເພດຂັ້ມູນ n ໂຕ, ຈຳນວນການວິນລຸບ ຂອງ tuple ກາຍເປັນວິນລຸບ tuple ທີ່ມີຈຳນວນ tuple ເທົ່າວັບຈຳນວນຕົວປ່ຽນການສິ່ງ.
- | ໃນທີ່ນີ້, ອົງປະກອບທີ່ i ຫຼື tuple ປະກອບດ້ວຍອົງປະກອບທີ່ i ຂອງແຕ່ລະປະເພດຂັ້ມູນແຕ່ລະຊະນິດໄດ້ຜ່ານໄປທາຟັງຊັນ

## 2. ການລວມໂດຍໃຊ້ຟັງຊັນ Zip(Aggregation Using Zip Function)

### 2.3. ຕົວຢ່າງ code ຄໍາສັ່ງທີ່ໃຊ້ຟັງຊັນ zip

```
1 a = [10, 20, 30]           # list a
2 b = ('ten', 'twenty', 'thirty') # tuple b
3 for val in zip(a, b):       # print tuple created by aggregating list a and tuple b
4     print(val)
```

(10, 'ten')  
(20, 'twenty')  
(30, 'thirty')



## 2. ການລວມໂດຍໃຊ້ຟັງຊັນ Zip(Aggregation Using Zip Function)

### 2.4. ຂໍ້ຄວນລະວົງສໍາລັບການນຳໃຊ້ຟັງຊັນ zip



ຖ້າມີການປ້ອນຂໍ້ມູນໃຫ້ກັບ list ຜ່ານຟັງຊັນ zip.

ໃຫ້ຮູ້ໄວ້ວ່າອີງປະກອບຜົນຮັບແມ່ນ tuple ດັ່ງນັ້ນຈຶ່ງປະກິດເປັນ tuple ('a',) ທີ່ມີອີງປະກອບໜຶ່ງເອັນວ່າ 'a'. ບໍ່ແມ່ນພຽງແຕ່ 'a' ເພາະວ່າມັນແມ່ນ tuple

```
1 lst = ['a', 'b', 'hello', 'this']
2 my_iterator = zip(lst)
3 result = set(my_iterator)
4 result
```

```
{('a',), ('b',), ('hello',), ('this',)}
```

## 2. ການລວມໂດຍໃຊ້ຝັງຂັນ Zip(Aggregation Using Zip Function)

### 2.5. ຕົວຢ່າງການ Zip

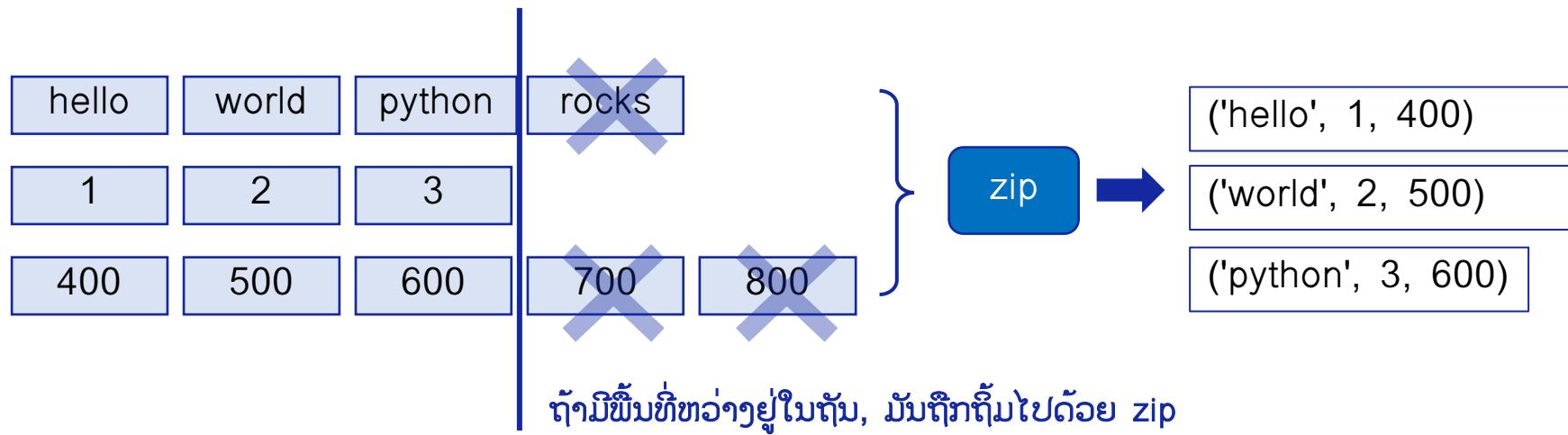
| ການຮອບຮວມຂໍ້ມູນໃນ list, ກໍາລະນີປະເພດຂໍ້ມູນຫລາຍຊະນິດ ແລະ ອົງປະກອບຕ່າງກັນ. ອັນໄດ້ແດ່ທີ່ຖືກຕັດອອກໃນທີ່ນີ້?

```

1 str_list = [ 'hello', 'world', 'python', 'rocks']
2 int_tuple = (1, 2, 3)
3 int_list = [400, 500, 600, 700, 800]
4 my_iterator = zip(str_list, int_tuple, int_list)
5 list(my_iterator)

```

[('hello', 1, 400), ('world', 2, 500), ('python', 3, 600)]



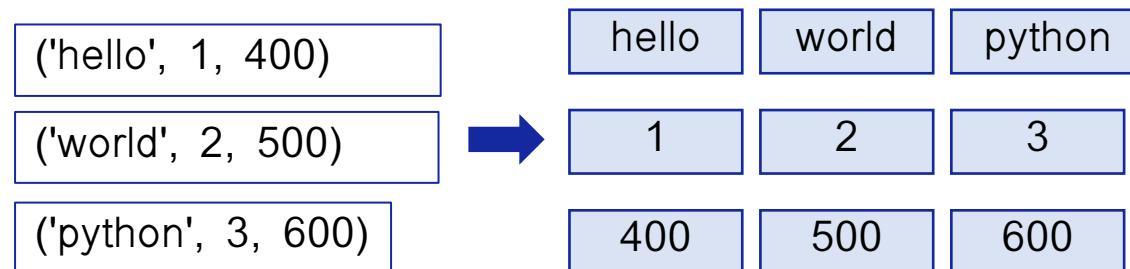
## 2. ການລວມໄດຍໃຊ້ຟັງຂັນ Zip(Aggregation Using Zip Function)

### 2.6. Zip ແລະ tuple unpacking

- | Unpacking ແມ່ນການດໍາເນີນງານຂອງການສົ່ງຄືນ tuples ຜ່ານຟັງຂັນ zip ກັບຂໍ້ມູນຕົ້ນສະບັບ.
- | ແທນທີ່ຈະຂຽນຟັງຂັນອື່ນ, ເຮົາຈໍາເປັນຕ້ອງປ້ອນຂໍ້ມູນທີ່ເຊື່ອມໄຍງໝັກຟັງຂັນ zip ເປັນອົງປະກອບ ແລະ ໄສ່ເຄື່ອງໝາຍ \* ຢູ່ທາງໜ້າຂອງຕົວປ່ຽນ.

```
1 zip_lst = list(zip(str_list, int_tuple, int_list))
2 a, b, c = zip(*zip_lst)      # unpack tuple
3 print(a, b, c)
```

```
('hello', 'world', 'python') (1, 2, 3) (400, 500, 600)
```



# | Pair programming



# Pair Programming Practice

## | ແນວທາງ, ກົມໄກ ແລະ ແຜນສຸກເສີນ

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

## | ຈັບຄຸ້ທີ່ຄ້າຍຄືກັນ, ບໍ່ຈໍາເປັນເຫົ້າທຽມກັນ, ຄວາມສາມາດເປັນຄຸ້ຮ່ວມງານ

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

## | ກະຕຸ້ນນັກຮຽນໂດຍການໃຫ້ສິ່ງຈູ່ໃຈພື້ເສດ

ຂໍສະເໜີແຮງຈູ່ໃຈທີ່ຮັດໃຫ້ນັກຮຽນຈັບຄຸ້, ໂດຍສະເພາະນັກຮຽນທີ່ມີຄວາມສາມາດສູງ. ບາງຄຸສອນໄດ້ພື້ນວ່າ ການຈັບຄຸ້ຮັດວຽກມອບໝາຍ ແມ່ນມີ ປະໂຫຍດ ສໍາລັບໜຶ່ງ ຫຼື ສອງວຽກມອບເທົ່ານັ້ນ



## Pair Programming Practice

### | ป้องกันงานบ่ตั้งในระบบของมัคกรูน

ສິ່ງທ້າທາຍສໍາລັບຄຸນແມ່ນເພື່ອຊອກຫາວິທີທີ່ຈະປະເມີນຜົນການຮຽນຂອງມັກຮຽນ, ຄຸນບໍ່ວ່າ ມັກຮຽນ ໄດ້ຕັ້ງໃຈຮຽນ ຫຼື ບໍ່ຕັ້ງໃຈຮຽນ. ຜູ້ສ່ວວຊານໄດ້ແນະນຳໃຫ້ທີບທວນການອອກແບບຫຼັກສຸດການຮຽນ ແລະ ຮູບແບບການປະເມີນ ພ້ອມທັງປີກາຫາລືຢ່າງຈົງຈ້າງກັບມັກຮຽນ ກ່ຽວກັບພິດຕິກຳທີ່ຈະບໍ່ຕັ້ງໃຈຮຽນ ນອກຈາກນີ້ຢັ້ງໄດ້ແນະນຳມອບວຽກມອບໝາຍໃຫ້ມັກຮຽນ ພ້ອມທັງອະທິບາຍໃຫ້ເຂົ້າເຈົ້າຢ່າງຈະແຈ້ງ

### | ສະພາບແວດລ້ອມຂອງການຮຽນຮູ້ຮ່ວມກັນ

ສະພາບແວດລ້ອມການຮຽນຮູ້ຮ່ວມກັນເກີດຂຶ້ນໄດ້ທຸກເວລາທີ່ຜູ້ສອນຮຽກຮ້ອງໃຫ້ມັກຮຽນເຮັດວຽກຮ່ວມກັນໃນກິດຈະກຳການຮຽນຮູ້ ເຊິ່ງ ອາດຈະເປັນກິດຈະກຳທີ່ເປັນທາງການ ແລະ ບໍ່ເປັນທາງການ ແລະ ອາດຈະບໍ່ລວມເຖິງການປະເມີນຜົນການຮຽນໂດຍກິງ. ເຊັ່ນຕົວຢ່າງ ໃຫ້ ມັກຮຽນຈັບຄຸນເພື່ອເຮັດວຽກມອບໝາຍ ໂດຍມັກຮຽນຈະຕ້ອງທີບທວນກ່ຽວກັບການສອນຂອງອາຈານທີ່ຜ່ານມາ ແລະ ລະດົມແນວຄິດພາຍໃນກຸ່ມ ພ້ອມທັງມືການແບ່ງວຽກໃຫ້ແຕ່ລະຄົນຮັບຜິດຊອບ ຈາກນັ້ນກໍໃຫ້ມີການແລກປ່ຽນຄວາມຄິດເຫັນເຊິ່ງກັນ ແລະ ກັນ ເພື່ອເຮັດວຽກມອບໝາຍໃຫ້ສໍາເລັດຕາມເປົ້າໝາຍທີ່ວ່າງໄວ້.

Q1.

ມີ list ຂີ່ມີ tuple(m, n) ເປັນອົງປະກອບຕາມທີສະແດງຂ້າງລຸ່ມນີ້. ຖ້າມີ tuple ທີ່ມີຄ່າ (a,b) ແຊ່ງກ່າວ a, b ແມ່ນບ້ອນຈາກຜູ້ໃຊ້, ໃຫ້ພິມ 'ມີອົງປະກອບ (a, b) ໃນ xth'. ຖ້າບໍ່ມີ (a, b) ແຕ່ມີ (b, a) ຢູ່ໃນນັ້ນ, ພິມ 'There is no (a, b) but (b, a) ມີຢູ່ທີ່ yth. ຖ້າບໍ່ມີ (a, b) ຫຼື (b, a), ໃຫ້ພິມ 'there is no' ໃນອົງປະກອບ.

```
| mylist = [(1, 2), (4, 5), (4, 2), (3, 1), (9, 4)]
```

## Output Example

```
Enter two integers: 1 2
```

```
There is (1,2) at the first.
```

```
Enter two integers: 5 4
```

```
There is no (5,4) but there is (4,5) at the second.
```

```
Enter two integers: 3 9
```

```
There is no (3,9) nor (9,3)
```