สัปดาห์ที่ 6 การสร้างลิสต์ เซต และการทำงานซ้ำด้วย *for*

1. ลิสต์ *(list)*

ลิสต์ในภาษา *Python* สามารถเก็บข้อมูลในลิสต์ได้หลายตัว โดยที่แต่ละตัวไม่จำเป็นต้องเป็นชนิดเดียวกันก็ได้ แต่ถ้าผู้ใช้ไม่ใช้ลิสต์ผู้ใช้จะต้องสร้างตัวแปรขึ้นมาหลายตัว เท่ากับจำนวนข้อมูล

1.1 การสร้างลิสต์

ผู้ใช้สามารถสร้างลิสต์ ได้ *2* รูปแบบ *1.* การสร้างลิสต์ว่าง *2.* การสร้างลิสต์แบบมีข้อมูล

การสร้างลิสต์ว่าง

เมื่อผู้ใช้ต้องการสร้างลิสต์ว่างสามารถสร้างได้โดยใช้เครื่องหมาย *square brackets* " [" และ "] "

ตัวอย่าง *1* การสร้าง *list* ว่าง

```
1:      x =[]
2:      print(x)
```

ผลลัพธ์

```
[]
```

อธิบาย

บรรทัด *1* สร้างตัวแปร *x* เก็บข้อมูลลิสต์ว่าง

บรรทัด *2* คำสั่ง *print(x)* แสดงข้อมูลในตัวแปร *x*

การสร้างลิสต์แบบมีข้อมูล

เมื่อผู้ใช้ต้องการสร้างลิสต์แบบมีข้อมูลให้ผู้ใช้ระบุข้อมูล ครอบด้วยเครื่องหมาย *square brackets* " [" และ "] " ถ้าหากผู้ใช้ต้องการสร้างลิสต์แบบมีข้อมูลหลายจำนวน จะใช้เครื่องหมาย *comma* " , " คั่นกลางระหว่างข้อมูล และถูกครอบด้วยเครื่องหมาย *square brackets* " [" และ "] "

ตัวอย่าง *2* การสร้าง *list* แบบมีข้อมูล

```
1:      a =['a']
2:      b =[1, 2, 3, 4, 'Buu', 5, 'Informatics']
3:      print(a)
4:      print(b)
```

ผลลัพธ์

```
['a']
[1, 2, 3, 4, 'Buu', 5, 'Informatics']
```

อธิบาย

บรรทัด *1* สร้างตัวแปร *a* เก็บข้อมูลชนิดลิสต์ ประกอบไปด้วยข้อความ

บรรทัด *2* สร้างตัวแปร *b* เก็บข้อมูลชนิดลิสต์ ประกอบไปด้วยตัวเลข และข้อความ

(บรรทัดอื่น ๆ เหมือนกับตัวอย่าง *1)*

1.2 การเลือกแสดงข้อมูลในลิสต์

การนับตำแหน่งเพื่อเลือกข้อมูลในลิสต์มาแสดงนั้น มีการนับ 2 รูปแบบ การนับตำแหน่งจากด้านหน้าและการนับตำแหน่งจากด้านหลัง โดยการนับตำแหน่งจากด้านหน้าจะเริ่มลำดับด้วย *0, 1, 2, 3, … , n – 1* และการนับตำแหน่งจากด้านหลังจะเริ่มลำดับด้วย *-1, -2, -3, -4 , … , -n* ซึ่งการกำหนดตำแหน่งจะอยู่ภายในเครื่องหมาย *square brackets “ [” และ “] ”* โดยใส่เป็นตัวเลข (หรือจะใส่ตัวแปรก็ได้)

ตัวอย่าง *3* การเลือกแสดงข้อมูลในลิสต์

```
1:      i =[1,2,3,4,'Buu',5,'Informatics']
2:      print(i[4])
3:      x =-1
4:      print(i[x])
```

ผลลัพธ์

```
Buu
Informatics
```

อธิบาย

บรรทัด *2* คำสั่ง *print(i[4])* เพื่อแสดงข้อมูลในตำแหน่ง *4* จากในลิสต์ ซึ่งคือคำว่า *Buu*

บรรทัด *3* สร้างตัวแปร *x* เพื่อเก็บข้อมูลตำแหน่งที่ *-1*

บรรทัด *4* คำสั่ง *print(i[x])* เพื่อแสดงข้อมูลในตำแหน่ง *-1* (หรือตำแหน่งสุดท้าย) จากในลิสต์ ซึ่งคือ คำว่า *Informatics*

(บรรทัดอื่น ๆ เหมือนในตัวอย่าง *2*)

1.3 การเพิ่มข้อมูลในลิสต์

ตัวอย่าง 4 การเพิ่มข้อมูล 1, 2, 6 และ 3 เข้าในลิสต์ตามลำดับ

```
1:    list = []
2:    list.append(1)
3:    list.append(2)
4:    list.append(6)
5:    list.append(3)
6:
7:    print(list)
```

ผลลัพธ์

```
[1, 2, 6, 3]
```

1.4 การแทรกข้อมูลในลิสต์

ตัวอย่าง 5 การแทรกข้อมูล *“net”* เข้าในลิสต์ในตำแหน่งอินเด็กซ์ที่ 1

```
1:    list = ["dot", "perls"]
2:
3:    # แทรกข้อมูลเข้าในตำแหน่งที่ 1
4:    list.insert(1, "net")
5:
6:    print(list)
```

ผลลัพธ์

```
['dot', 'net', 'perls']
```

1.5 การขยายลิสต์

ตัวอย่าง 6 การขยายลิสต์ *a* ด้วยข้อมูลในลิสต์ *b*

```
1:   a = [1, 2, 3]
2:   b = [4, 5, 6]
3:   # เพิ่มสมาชิกทั้งหมดในลิสต์ b ไปยังลิสต์ a
4:   a.extend(b)
5:   # ลิสต์ a จะมีสมาชิก 6 ตัว
6:   print(a)
```

ผลลัพธ์

```
[1, 2, 3, 4, 5, 6]
```

1.6 การนับขนาดของลิสต์

ตัวอย่าง 7 การนับขนาดของลิสต์

```
1:   animals = []
2:   animals.append("cat")
3:   animals.append("dog")
4:   count = len(animals)
5:   # แสดงข้อมูลในลิสต์และขนาดของลิสต์
6:   print(animals)
7:   print(count)
```

ผลลัพธ์

```
['cat', 'dog']
2
```

1.7 การใช้คำสั่ง in และ not in

ตัวอย่าง 8 การใช้คำสั่ง *in* และ *not in*

```
1:   items = ["book", "computer", "keys", "mug"]
2:   if "computer" in items:
3:      print(1)
4:   if "atlas" in items:
5:      print(2)
6:   else:
7:      print(3)
8:   if "marker" not in items:
9:      print(4)
```

ผลลัพธ์

```
1
3
4
```

1.8 การลบข้อมูลออกจากลิสต์

ตัวอย่าง 9 การลบข้อมูลออกจากลิสต์

```
1:   names = ["Tommy", "Bill", "Janet", "Bill", "Stacy"]
2:   # ลบข้อมูลที่ระบุคือ Bill
3:   names.remove("Bill")
4:   print(names)
5:   # ลบทุกตัว ยกเว้น 2 ตัวแรก
6:   del names[2:]
7:   print(names)
8:   # ลบทุกตัว ยกเว้นตัวสุดท้าย
9:   del names[:1]
10:  print(names)
```

ผลลัพธ์

```
['Tommy', 'Janet', 'Bill', 'Stacy']
['Tommy', 'Janet']
['Janet']
```

2. *For*

ในภาษา *Python* จะมีคำสั่ง *for* เป็นคำสั่งที่ใช้ในการวนซ้ำ ในช่วงของการทำงาน *for* จนกว่าเงื่อนไขทางตรรกศาสตร์จะเป็นเท็จ โดยมีการกำหนดช่วงการทำงาน ต่ำสุด และ มากสุด เป็นจำนวนครั้ง คำสั่งที่ถูกทำซ้ำ จะถูกใส่เข้าภายใน *for* โดยกลุ่มคำสั่งเหล่านี้จะต้องอยู่ในย่อหน้าเสมอ โดยทั่วไปเราจะใช้ *for* ร่วมกับคำสั่ง *in* และฟังก์ชัน *range* มีรูปแบบดังต่อไปนี้

for *var* ***in*** *range() :*

Statements

.....

- *for* คำสั่ง *for*
- *var* ตัวแปร (สามารถกำหนดได้ตามต้องการ)
- *in* *operator* ที่ใช้ในการเปรียบเทียบ
- *range(x, y, z)* ช่วงที่จะให้ *for* ทำงาน ตั้งแต่ *var* เป็น *x* จนถึง *var* เป็น *y – 1* โดยค่าของ *var* จะเพิ่มขึ้นทีละ *z*

ตัวอย่าง *13* คำสั่ง *for*

```
1:      for i in range(0, 5, 1):
2:          print('round :', i)
```

ผลลัพธ์

```
round :0
round :1
round :2
round :3
round :4
```

อธิบาย

บรรทัด *1* ใช้คำสั่ง *for* และสร้างตัวแปร *i* เพื่อใช้ในการเก็บเลขที่วิ่งสำหรับนับรอบที่จะวนซ้ำ ข้อมูลในตัวแปร *i* จะถูกเปลี่ยนในทุก ๆ รอบ (ในตัวอย่าง *i* จะเพิ่มขึ้นทีละ *1)* คำสั่ง *in* จะทำงานร่วมกับ *range* โดยกำหนดค่า *var* ให้เริ่มต้นที่ *0* โดย *i* จะวนไปจน *i* เป็น *5 – 1* และเพิ่มค่า *+ 1* ในทุก ๆ รอบการทำงาน

บรรทัด *2* คำสั่ง *print('round :' , i)* แสดงผลคำว่า *round :* ตามด้วยค่าที่อยู่ในตัวแปร

ตัวอย่าง 14 คำสั่ง *for* ทำงานร่วมกับ คำสั่ง *if else*

```
1:      for i in range(0, 5, 1):
2:          if i%2 ==0 :
3:              print(i, 'is an even number')
4:          else :
5:              print(i, 'is an odd number')
```

ผลลัพธ์

```
0 is an even number
1 is an odd number
2 is an even number
3 is an odd number
4 is an even number
```

อธิบาย

บรรทัด *2 – 5* กำหนดเงื่อนไข ถ้า *i mod 2* ได้ผลลัพธ์เป็น *0* จริง *(True)* ก็จะแสดง ค่า *i* ตามด้วย *is an even number* ผ่านทางหน้าจอ ถ้าเป็นเท็จ *(False)* ก็จะแสดง ค่า *i* ตามด้วย *is an odd number*

เราสามารถเขียน *for* ได้อีกรูปแบบหนึ่ง

for *var* ***in*** *sequence* :
    Statements
    ……

โดย *sequence* อาจเป็น *String* , *List* , *Set* เป็นต้น

ตัวอย่าง *15* ใช้ *for* ร่วมกับข้อความ

```
1:      for i in 'Buu-IT':
2:          print(i)
```

ผลลัพธ์

```
B
u
u
-
I
T
```

อธิบาย

บรรทัด *1* ใช้คำสั่ง *for* และสร้างตัวแปร *i* เพื่อใช้ในการเก็บเลขที่วิ่งสำหรับนับรอบที่จะวนซ้ำ ข้อมูลในตัวแปร *i* จะถูกเปลี่ยนในทุก ๆ รอบ คำสั่ง *in* เป็นคำสั่งที่กำหนดว่าโปรแกรมจะถูกทำซ้ำจนกว่าจะครบทุกตัวอักษร ในคำว่า *Buu-IT* โดยในรอบที่ *1* *i* จะมีค่าเท่ากับ *B* ในรอบที่ *2* *i* จะมีค่าเท่ากับ *u* ตามลำดับ

บรรทัด *2* ใช้คำสั่ง *print(i)* เพื่อแสดงข้อมูลที่อยู่ในตัวแปร *i* ผ่านทางหน้าจอ

3. Set

เซต *(Set)* ใช้แทนกลุ่มของข้อมูลต่าง ๆ โดยสมาชิกที่อยู่ภายในเซต อาจเป็นตัวเลข หรือตัวอักษรก็ได้และลำดับการเรียงลำดับของสมาชิกในเซตนั้นไม่มีความสำคัญสำหรับผู้ใช้ นิยมใช้ตัวอักษรภาษาอังกฤษตัวพิมพ์ใหญ่ในการแทนเซตใด ๆ เช่น *A = set([1, 5, 7])*

3.1 การสร้างเซต

เซตในภาษา *Python* นั้น เราสามารถสร้างเซตได้โดยใช้ฟังก์ชัน *set(var)* มีรูปแบบดังนี้

ตัวอย่าง 16 สร้างเซตโดยใช้ลิสต์

```
1:      A =set([1, 5, 7])
2:      print(A)
```

ผลลัพธ์ที่ได้จะเป็นดังนี้

```
{1, 5, 7}
```

หมายเหตุ *{ }* ใช้ระบุเซต ในขณะที่ *[ ]* ใช้ระบุลิสต์

อธิบาย

บรรทัด *1* สร้างตัวแปรชื่อ *A* เพื่อเก็บข้อมูลในรูปแบบลิสต์ และข้อมูลถูกทำให้เป็นข้อมูลแบบเซตด้วยฟังก์ชัน *set()*

บรรทัด *2* ใช้คำสั่ง *print(A)* เพื่อแสดงข้อมูลที่อยู่ในตัวแปร *A* ผ่านทางหน้าจอ

ตัวอย่าง 17 สร้างเซตว่าง

```
1:      A =set()
2:      print(A)
```

ผลลัพธ์ที่ได้จะเป็นดังนี้

```
set()
```

อธิบาย

บรรทัด *1* สร้างตัวแปร *A* เก็บข้อมูลในรูปแบบเซตว่าง

บรรทัด *2* ใช้คำสั่ง *print(A)* เพื่อแสดงข้อมูลที่อยู่ในตัวแปร *A* ผ่านทางหน้าจอ

ตัวอย่าง 18 ข้อมูลเซตในรูปแบบข้อความ

```
1:      A ={'a', 'b', 'c'}
2:      print(A)
```

ผลลัพธ์ที่ได้จะเป็นดังนี้

```
{'b', 'a', 'c'}          ผลรัน ครั้งที่ 1
{'c', 'a', 'b'}          ผลรัน ครั้งที่ 2
```

อธิบาย

บรรทัด *1* สร้างตัวแปรชื่อ *A* เก็บข้อมูลในรูปแบบเซต

บรรทัด *2* ใช้คำสั่ง *print(A)* เพื่อแสดงข้อมูลที่อยู่ในตัวแปร *A* ผ่านทางหน้าจอ

ข้อสังเกต จะเห็นได้ว่าผลลัพธ์จากการรัน *2* ครั้ง มีลำดับข้อมูลไม่เหมือนกัน เนื่องจากเซตใน *Python* ไม่ได้ถูกออกแบบให้คำนึงถึงความสำคัญในการแสดงลำดับก่อน หลัง

ตัวอย่าง *19* แปลงข้อมูลในรูปแบบข้อความให้อยู่ในรูปของเซต

```
1:      A =set('12345')
2:      print(A)
```

ผลลัพธ์ที่ได้จะเป็นดังนี้

```
{'4', '2', '1', '3', '5'}
```
ผลรัน ครั้งที่ *1*

```
{'1', '2', '5', '3', '4'}
```
ผลรัน ครั้งที่ *2*

อธิบาย

บรรทัด *1* สร้างตัวแปรชื่อ *A* ใช้เก็บเซต และใช้ฟังก์ชัน *set()* เพื่อแปลง ข้อความให้อยู่ในรูปแบบของเซต โดยข้อความจะถูกแยกทีละ *1* ตัวอักษร

บรรทัด *2* ใช้คำสั่ง *print(A)* เพื่อแสดงข้อมูลที่อยู่ในตัวแปร *A* ผ่านทางหน้าจอ

ข้อสังเกต จะเห็นว่าถ้าเราใส่ข้อความ *(String)* ในเครื่องหมาย วงเล็บ " (" และ ") " ข้อความจะถูกแยกออกทีละ *1* ตัวอักษรและถูกใส่เข้าไปในเซต แต่ถ้าเราอยากใส่ข้อความเป็นคำโดยไม่ถูกแยกออกเป็นตัวอักษร จะต้องมีเครื่องหมาย *square bracket* " [" และ "] " ครอบก่อน *1* ครั้ง (ทำเป็นลิสต์ก่อน) และครอบด้วย วงเล็บ " (" และ ") " เช่น *A = set(['Buu-IT' , 'Informatics'])* จะได้ผลลัพธ์เป็น *{'Buu-IT', 'Informatics'}*

3.2 ฟังก์ชันที่เกี่ยวกับ *Set* ที่ภาษา *Python* เตรียมไว้ให้

การเพิ่มสมาชิกในเซตครั้งละ *1* ตัว

ผู้ใช้สามารถเพิ่มสมาชิก ครั้งละ *1* ตัว ในเซตได้ โดยใช้คำสั่ง *add( )*

ตัวอย่าง *20* การเพิ่มสมาชิกในเซตครั้งละ *1* ตัว

```
A ={1, 5, 7}
A.add(9)
print(A)
```

ผลลัพธ์ที่ได้จะเป็นดังนี้

```
{1, 5, 9, 7}
```

ข้อสังเกต คำสั่ง *add* จะใส่ข้อมูลที่ตำแหน่งใดในเซตก็ได้ เนื่องจากลำดับไม่มีความสำคัญในเซต

การเพิ่มสมาชิกในเซตครั้งละหลายตัว

ผู้ใช้สามารถเพิ่มสมาชิก ครั้งละหลายตัว ในเซตได้โดยใช้คำสั่ง *update([ ])*

ตัวอย่าง *21* การเพิ่มสมาชิกในเซตครั้งละหลายตัว

```
A ={1, 5, 9, 7}
A.update([11, 'Buu'])
print(A)
```

ผลลัพธ์ที่ได้จะเป็นดังนี้

```
{1, 5, 9, 7, 11, 'Buu'}
```

การลบสมาชิกในเซตครั้งละ *1* ตัว

ผู้ใช้สามารถลบสมาชิก ครั้งละ *1* ตัว ในเซตได้โดยใช้คำสั่ง *discard()*

ตัวอย่าง *22* การลบสมาชิกในเซตครั้งละ *1* ตัว

```
A ={1, 5, 9, 7, 11, 'Buu'}
A.discard(9)
print(A)
```

ผลลัพธ์ที่ได้จะเป็นดังนี้

```
{1, 5, 7, 11, 'Buu'}
```

การลบสมาชิกในเซตทั้งหมด

ผู้ใช้สามารถลบสมาชิกทั้งหมดในเซตได้โดยใช้คำสั่ง *clear()*

ตัวอย่าง *23* การลบสมาชิกในเซตทั้งหมด

```
A ={1, 5, 9, 7, 11, 'Buu'}
A.clear()
print(A)
```

ผลลัพธ์ที่ได้จะเป็นดังนี้

```
set()
```

การตรวจสอบสมาชิก

ผู้ใช้สามารถตรวจสอบสมาชิกในเซตว่ามีข้อมูลอยู่ในเซตว่าเป็นจริง หรือเท็จ โดยใช้คำสั่ง *in*

ตัวอย่าง *24* การตรวจสอบว่ามีสมาชิกในเซต

```
A ={1, 5, 9, 7, 11, 'Buu'}
print(9 in A)
```

ผลลัพธ์ที่ได้จะเป็นดังนี้

```
True
```

ผู้ใช้สามารถตรวจสอบสมาชิกว่าในเซตไม่มีข้อมูลอยู่ในเซตว่าเป็นจริง หรือเท็จ โดยใช้คำสั่ง *not in*

ตัวอย่าง *25* การตรวจสอบว่าไม่มีสมาชิกในเซต

```
A ={1, 5, 9, 7, 11, 'Buu'}
print(9 not in A)
```

ผลลัพธ์ที่ได้จะเป็นดังนี้

```
False
```

การนับจำนวนสมาชิกในเซต

ผู้ใช้สามารถนับจำนวนข้อมูลในเซตได้ โดยใช้คำสั่ง *len()*

ตัวอย่าง *26* นับจำนวนสมาชิกในเซต

```
A ={1, 5, 9, 7, 11, 'Buu'}
print(len(A))
```

ผลลัพธ์ที่ได้จะเป็นดังนี้

```
6
```