เรื่องของ list

ให้ x เป็น list

x[a:b] ก็เป็น list

x[a:b] = c # c ต้องเป็น list

```
x = [1,2,[4,5],3]
```

x[0:1] คือ [1] แต่ x[0] คือ 1

x[2:3] คือ [[4,5]] แต่ x[2] คือ [4,5]

x[1:1] = [9,9] ได้ x เป็น [1,9,9,[4,5],3]

x[1:1] = 9 แบบนี้ผิด

x.append(c) เหมือนกับ x[len(x):] = [c]

x.insert(k,c) เหมือนกับ x[k:k] = [c]

x.pop(k) เหมือนกับ x[k:k+1] = []

สร้าง list ว่าง

```
x = list()
x = []
```

การสร้าง list ด้วยคำสั่ง

```
x = list(b)
```

ก็เหมือนกับ

```
x = []
for e in b:
  x.append(e)
```

```
x = list('1224')
```

ได้ ['1','2','2','4']

```
x = list([1,2,2,4])
```

ได้ [1,2,2,4]

```
x = list( {1,2,3,4,1} )
```

ได้ list ที่มีค่า 1,2,3,4 ลำดับเป็นอย่างไร ไม่แน่

```
x = list( {1:2,3:4} )
```

ได้ list ที่มีค่า 1,3 ลำดับเป็นอย่างไร ไม่แน่

```
x = list( range(1,7,2) )
```

ได้ [1,3,5]

คำสั่ง x[::-1] คือ list ใหม่ที่มีค่าการกลับลำดับกับค่าใน x (x ไม่เปลี่ยน)

```
x = [1,2,3]
```

x[::-1] ได้ [3,2,1] แต่ x เหมือนเดิม

```
x = [1,2,3,4,5]
```

x[-2] ได้ 4, x[-2:] ได้ [4,5], x[:-2] ได้ [1,2,3]

x[2] ได้ 3, x[2:] ได้ [3,4,5] x[: 2] ได้ [1,2]

การบวก list คือการต่อ list

การคูณ list ด้วยจำนวนเต็มคือการต่อ list หลายครั้ง

```
x = [ [1,2], 'x' ]
```

x + [9] ได้ [[1,2], 'x', 9]

x * 2 ได้ [[1,2], 'x', [1,2], 'x']

```
x = [e for e in range(n) if e > 5]
```

เหมือนกับ -->

```
x = []
for e in range(n):
  if e > 5 :
    x.append(e)
```

```
x = [(i,j) for i in range(n)
           for j in range(m)
           if i >= j ]
```

เหมือนกับ -->

```
x = []
for i in range(n):
  for j in range(m):
    if i >= j :
      x.append( (i,j))
```

```
x = [ [ (i,j) for i in range(n)]
      for j in range(m)      ]
```

```
x = []
for j in range(m):
   t = []
   for i in range(n):
     t.append( (i,j) )
   x.append(t)
```

"1, 2,3, 4".split() ได้ ['1,', '2,3,', '4']

"1, 2,3, 4".split(",") ได้ ['1', ' 2', '3', ' 4']

เรื่องของ set	
s = {} ไม่ใช่การสร้าง set (แต่ได้ dict) s = set() ได้เซตว่าง สร้าง set ด้วยคำสั่ง s = set(b) ก็เหมือนกับ s = set() for e in b: s.add(e)	s = set('1222') ได้ {'1','2'} s = set([2,1,2,2]) ได้ {1,2} s = set({1:2,3:4}) ได้ {1,3} s = set(range(1,7,2)) ได้ {1,3,5}
ข้อมูลใน list มีเลข index กำกับ แต่ข้อมูลใน set ไม่มี	x = [1,2,3]; y = {1,2,3} x[1] ได้ 2 แต่ y[1] ผิด
เก็บ list หรือ set หรือ dict ใน set ไม่ได้	s = set() s.add([1,2,3]) ผิด s.add({1:3}) ผิด s.add({1,3,2}) ผิด
เก็บ int, float, str, tuple ใน set ได้	s = set() s.add(1) s.add(1.2) s.add('abcd') s.add((1,2,3)) ได้หมด
s1.union(s2) เหมือนกับ s1 | s2 s1.intersection(s2) เหมือนกับ s1 & s2 s1.difference(s2) เหมือนกับ s1 - s2 ทั้งสามคำสั่งข้างบนนี้ s1 และ s2 ไม่เปลี่ยนแปลง	s1 = {1,2,3} s2 = {2,3,4} print(s1 & s2) ได้ {2,3} print(s1) ได้ {1,2,3} print(s2) ได้ {2,3,4}
เรื่องของ dict	
ให้ d เป็น dict เขียน d[x] แสดงว่า x ต้องเป็น key ของ d ก่อนจะหยิบ value ของ key อะไรใน dict ต้องมีการเก็บ value ใน dict ที่มี key นั้นก่อน	d = dict() d['x'] = [1,2,3] ถูก d['y'] = d['z'] ผิด เพราะ d ยังไม่มี key ที่มีค่าเป็น 'z' จึงไม่รู้ว่า d['z'] มีค่าเท่าไร d['y'] += 1 ผิด เพราะเหมือนกับเขียน d['y'] = d['y'] + 1 ซึ่งก็ผิดเพราะตอนนี้ d ยังไม่มี key ที่มีค่าเป็น 'y' จึงไม่รู้ว่า d['y'] มีค่าเท่าไร
ใช้ list หรือ set หรือ dict เป็น key ของ dict ไม่ได้	d = dict() d[[1,2,3]] = 1 ผิด d[{1:3}] = 1 ผิด d[{1,3,2}] ผิด
ใช้ int, float, str, tuple เป็น key ของ dict ได้	d = set() d[1] = 23 d[1.2] = 'abc' d['abcd'] = 12 d[(1,2,3)] = (1,2) ได้หมด
ใช้ข้อมูลอะไรก็ได้เก็บเป็น value ใน dict ได้ทั้งนั้น	d = {1:23,3:988,90:{1,2,3}} d[35] = [1,[2,3]] print(d[35][1][0]) ได้ 2
อย่าลืมว่าข้อมูลใน set ไม่ซ้ำกัน และข้อมูลที่เป็น key ของ dict ก็ไม่ซ้ำกันด้วย	

เรื่องของทั้ง list, set, tuple, dict, str, numpy array	
len ใช้ได้กับทั้ง str, list, set, tuple, dict, numpy array	d = {1:23, 40:{1,2,3,4}, 9:[34,0]} print(len(d)) ได้ 3 print(len(d[40]) ได้ 4
sum, max, min ใช้ได้กับทั้ง list, set, tuple, dict, numpy array	d = {1:23, 40:{1,2,41,3}, 9:[34,0]} print(sum(d)) ได้ 50 print(max(d[40]) ได้ 41
ถ้าเขียน x.sort() โดยที่ x คือ list ---> ใช้ได้ แต่ถ้า x คือ tuple หรือ set หรือ dict เขียน x.sort() ไม่ได้	แต่ใช้ sorted กับ list, tuple, set, dict ได้ เพราะเขียน y = sorted(x) ก็เหมือนกับเขียน y = sorted([e for e in x]) ดังนั้น d = {9:12, 3:27, 8:33} print(sorted(d)) ได้ [3, 8, 9]
for e in x :	ใช้ได้กับ x ที่เป็น str, list, set, tuple, dict, numpy array
for i in range(len(x)): e = x[i]	ใช้ได้กับ x ที่เป็น str, list, tuple, numpy array
if e in x:	ใช้ได้กับ x ที่เป็น str, list, set, tuple, dict, numpy array ถ้าต้องการค้น e ใน values ของ dict ก็ต้องใช้คำสั่ง if e in d.values() :
เรื่องของ function	

ตัวแปรที่มีการให้ค่าในฟังก์ชันต่าง ๆ
ถึงแม้จะชื่อเดียวกันก็เป็นคนละตัวกัน
	def f():
	  x = 12
	  print(x)
	def g():
	  x = 27
	  print(x)
	x = 9
	print(x) ได้ 9
	f() ได้ 12
	print(x) ได้ 9
	g() ได้ 27
	print(x) ได้ 9

ฟังก์ชันคืนค่า int, float, list, set, tuple, ... object อะไรก็ได้
ไม่คืนก็ได้ (ถ้าไม่คืนจะได้ผลเป็น None)
	def f(x):
	  if x > 0 :
	  return 123
	print(f(7)) ได้ 123
	print(f(-1)) ได้ None

ชื่อฟังก์ชันมีกฎการตั้งชื่อเหมือนชื่อตัวแปร	

เรื่องของ numpy	
คำสั่ง x = np.array(range(a,b,c)) เหมือนกับคำสั่ง x = np.arange(a,b,c)	
x.shape คืน tuple ระบุรูปร่างของอาเรย์ ถ้า x เป็นอาเรย์ 2 มิติ x.shape[0] คือจำนวนแถว x.shape[1] คือจำนวนคอลัมน์	x = np.array([[1],[2],[3]]) x.shape[0] ได้ 3 x.shape[1] ได้ 1 x.shape ได้ (3,1)
อย่าลืมการสร้าง numpy array ด้วย ndarray, zeros, ones, identity	x = np.ndarray((3,2), dtype=int) ได้อาเรย์ 3 แถว 2 คอลัมน์ แต่ละช่องมีค่าอะไรก็ไม่รู้ a = np.identity(4, dtype=int) b = np.zeros((4,5), dtype=float) c = np.ones((5,4) , dtype=int)
การเลือกแถวและคอลัมน์ในอาเรย์ x[a:b:c, d:e:f] a,b,c บอกลักษณะของแถวที่ต้องการ d,e,f บอกลักษณะของคอลัมน์ที่ต้องการ x[::,d:e:f] เหมือนกับ x[0:x.shape[0]:1, d:e:f] x[a:b:c] เหมือนกับ x[a:b:c,::] เหมือนกับ x[a:b:c, 0:x.shape[1]:1] x]::,::] เหมือนกับ x[0:x.shape[0]:1, 0:x.shape1]:1]	x = np.array([[1,2,3,4], [5,6,7,8], [9,8,7,6], [4,3,2,1]]) x[0:4:2, 1:4:2] หรือเขียน x[::2, 1::2] คือ เลือกเฉพาะแถวคู่ คอลัมน์คี่ ได้ array([[2,4], [8,6]]) ถ้าเขียน x[::2,1::2] = 0 จะได้ x กลายเป็น array([[1,0,3,0], [5,6,7,8], [9,0,7,0], [4,3,2,1]])
broadcasting & element-wise operations x = np.array([[1,2,3],[4,5,6]]) y = x + 2 y = x + [1,2,3] y = x + np.array([1,2,3]) y = x + np.array([[1,2,3]]) y = x + np.array([[1,2,3]]).T ผิด y = x + np.array([[1,2]]).T	 array([[3,4,5],[6,7,8]]) array([[2,4,6],[5,7,9]]) array([[2,4,6],[5,7,9]]) array([[2,4,6],[5,7,9]]) ผิด array([[2,3,4],[6,7,8]])
เข้าใจการใช้งานของ axis ใน np.sum np.max np.min np.argmax ...	x = np.array([[1,6,3],[4,2,5]]) np.sum(x,axis=0) ได้ array([5,8,8]) np.sum(x,axis=1) ได้ array([10,11]) np.argmax(x[1]) ได้ 2 np.argmax(x,axis=0) ได้ array([1,0,1])
x = np.array([1,2,3]) ได้อาเรย์ 1 มิติ 3 ช่อง y = np.array([[1,2,3]])ได้อาเรย์ 2 มิติ 1 แถว 3 คอลัมน์ x.T ได้อาเรย์เหมือนกับ x y.T ได้อาเรย์ 3 แถว 1 คอลัมน์	x.shape ได้ (3,) y.shape ได้ (1,3) x.T.shape ได้ (3,) y.T.shape ได้ (3,1) print(x.dot(x.T)) ได้ 14 print(y.dot(y.T)) ได้ array([[14]]) print(y.T.dot(y)) ได้ array([[1, 2, 3], [2, 4, 6], [3, 6, 9]])

เรื่องของ class

ข้อมูลย่อยของอ็อบเจกต์ จะเป็นอะไรก็ได้

```
import numpy as np
class A :
  def __init__(self):
    self.count = 0
  def __increment__(self):
    self.count += 1

class B :
  def __init__(self, c):
    self.a = np.identity(c, dtype=int)

class C :
  def __init__(self, a, b):
    self.x = dict()
    x[a] = b
```

```
a = A()
a,increment()
a.increment()
print(a.count)  # 2


b = B(2)
print(b.a)      # [[1 0]
                   [0 1]]

c = C('X', 2)
print(c.x['X']) # 2
```

คำสั่ง a < b ก็เหมือนกับ a.__lt__(b)

คำสั่ง str(a) ก็เหมือนกับ a.__str__()

คำสั่ง print(a) ก็เหมือนกับ print(str(a))

```
class A :
  def __init__(self, a, b):
    self.x = [a]*b

  def __str__(self):
    t = [str(e) for e in self.x]
    return '[' + ','.join(t) + ']'

  def __lt__(self, rhs):
    return self.x < rhs.x

  def increment(self):
    self.x.append(self.x[0])

  def addAll(self, c):
    for i in range(len(self.x)):
      self.x[i] += c
```

```
a = A(2,3)
b = A(9,2)
t = (a < b)      # True
t = a.__lt__(b) # True
s = str(a)       # '[2,2,2]'
s = a.__str__() # '[2,2,2]'
print(a)           # [2,2,2]
print(str(a))      # [2,2,2]
print(a.__str__()) # [2,2,2]

a.increment()
print(a)           # [2,2,2,2]

a.addAll(5)
print(a)           # [7,7,7,7]
```