1. ใครคือผู้ออกแบบภาษา `Python`

A)  Guido van Rossum

B)  Larry Wall

C)  Donald Knuth

D)  Jessi Knudsen Castaneda

E) ไม่มีข้อใดถูก

2. Larry Wall บอกว่า Computer Programming คล้าย ๆ กับกิจกรรมใด

A) Writing a recipe B) Having dinner C) Speaking English D) Playing Sudoku E) ไม่มีข้อใดถูก

3. ฮาร์ดแวร์ของเครื่องคอมพิวเตอร์ในปัจจุบันทำงานกับข้อมูลและคำสั่งที่ถูกเข้ารหัสแบบใด

A) รหัสเลขฐานสอง B) ตัวอักษร C) ภาษาไพตอน D) ภาษาซี E) ไม่มีข้อใดถูก

4. ข้อใดไม่จัดเป็นภาษาโปรแกรมระดับสูง (high-level programming language)

A) Java B) Assembly language C) C++ D) Python E) ไม่มีข้อใดถูก

5. 1 byte มีขนาดเท่าใด

A) 8 tokens B) 8 lines C) 8 words D) 8 bits E) ไม่มีข้อใดถูก

6. CPU ที่เป็นองค์ประกอบหนึ่งของเครื่องคอมพิวเตอร์เป็นคำย่อของข้อใด

A) Computer Processor Universal B) Communication Protocol Utilities
C) Cute Python Union D) Central Processing Unit E) ไม่มีข้อใดถูก

7. Compiler รับ computer program source code มาประมวลผลเพื่อให้ได้อะไรเป็นผลลัพธ์

A) รหัสภาษาเครื่อง B) รหัสภาษาซี C) รหัสแอพ D) รหัสลับ E) ไม่มีข้อใดถูก

8. ในเครื่องคอมพิวเตอร์ RAM คืออะไร

A) หน่วยความจำ B) หน่วยถอดรหัส C) หน่วยรับข้อมูล D) หน่วยสื่อสาร E) ไม่มีข้อใดถูก

9. ในปัจจุบันเครื่องคอมพิวเตอร์โนตบุ๊คทั่วไปสามารถทำคำสั่งรหัสเครื่องได้ด้วยความเร็วประมาณเท่าใด

A) พันคำสั่งต่อวินาที B) แสนคำสั่งต่อวินาที
C) สิบล้านคำสั่งต่อวินาที D) พันล้านคำสั่งต่อวินาที E) ล้านล้านคำสั่งต่อวินาที

10. ข้อใด**<u>ไม่ใช่</u>**ข้อดีของการเขียนโปรแกรมคอมพิวเตอร์ด้วยภาษาระดับสูง (แทนที่จะเขียนด้วยภาษาระดับต่ำ)

A) ทำความเข้าใจโปรแกรมที่คนอื่นเขียนได้ง่ายกว่า
B) นำโปรแกรมไปใช้งานได้กับเครื่องหลายยี่ห้อ ถ้าต้องปรับโปรแกรมก็ไม่มาก
C) แก้ไขโปรแกรมที่มีอยู่ให้ทำงานตามความต้องการใหม่ได้ง่ายกว่า
D) เขียนโปรแกรมขึ้นมาใหม่ตามข้อกำหนดก็ง่ายกว่า
E) เป็นข้อดีของการใช้ภาษาระดับสูงทุกข้อ

11. ข้อใดคือผลลัพธ์ของ `a = [[1.5,2.5],[3.5,4.5],[2.5,3.5]]; print(a[0][1]+a[1+1][0])`

A) `4.0` B) `5.0` C) `6.0` D) `Error` E) ไม่มีข้อใดถูก

12. คำสั่งในข้อใดผิด

A) `list("Hello")` B) `list(["Hello"])` C) `list("Hello","Bye")`
D) `list(["Hello","Bye"])` E) ไม่มีข้อใดผิด

13. ข้อใดคือผลลัพธ์ของ `print(["Mike", "Tonm", "Anna"][-1][-1])`

A) `Error` B) `Anna` C) `a` D) `[]` E) ไม่มีข้อใดถูก

14. ให้ `s = [20,19,17,14,10,5,0]` ข้อใดให้ผลลัพธ์ต่างจากพวก

A) `print(s[0:-1])`
B) `print(s[-7:-1])`
C) `print(s[0:].pop(-1))`
D) `print(s[:6])`
E) เหมือนกันทุกข้อ

15. เมื่อสั่งให้โปรแกรมทางขวานี้ทำงาน ข้อใดถูก

```
mygroup = ["Tae","Kao","Pete","Kin","Kam"]
print(mygroup[3:])
mygroup[:] = ["Momo", "Ton", "An"]
print(mygroup[0])
print(mygroup.pop(1))
print(len(mygroup))
```

A) โปรแกรมพิมพ์บรรทัดแรก `Kin Kam`
B) โปรแกรมพิมพ์บรรทัดที่สอง `Tae`
C) โปรแกรมพิมพ์บรรทัดที่สาม `Kao`
D) โปรแกรมพิมพ์บรรทัดที่สี่ `7`
E) ผิดทุกข้อ

16. ให้ `a = [1,2,3,4,5]` ข้อใดให้ผลลัพธ์ต่างจากพวก

A) `a[5:5]=[6]` B) `a[6:6]=[6]` C) `a[5:]=[6]` D) `a[len(a):]=[6]` E) เหมือนกันทุกข้อ

17. ข้อใดคือผลลัพธ์ของ `n = [1,2,3]; n.append([4,5,6]*2); print(len(n))`

A) `4` B) `5` C) `6` D) `9` E) ไม่มีข้อใดถูก

18. คำสั่ง `[[x, x + 1] for x in range(0, 2)]` ได้ผลเหมือนคำสั่งในข้อใด

A) `[0,1,1,2]`
B) `[0,1,1,2,2,3]`
C) `[[0,1],[1,2]]`
D) `[[0,1],[1,2],[2,3]]`
E) ไม่มีข้อใดถูก

19. ข้อใดคือผลลัพธ์ของโปรแกรมด้านขวา

```
id = [ [ 1 if col_idx == row_idx else 0 \
           for col_idx in range(0, 5) ] \
       for row_idx in range(0, 5) ]
print(len(id[5]))
```

A) `4` B) `5` C) `16` D) `25`
E) ไม่มีข้อใดถูก

20. ข้อใดคือผลลัพธ์ของ `a = "a#b#c#d"; print(len(a.split(a[1])))`

A) `Error` B) `4` C) `5` D) `8` E) ไม่มีข้อใดถูก

21. ต้องการเขียนฟังก์ชัน `rightRotate(x,n)` รับพารามิเตอร์ 2 ตัว คือ ลิสต์นำเข้า (`x`) และ จำนวนตำแหน่งที่ต้องการขยับ (`n`) ที่ทำหน้าที่ "หมุน" ข้อมูลในลิสต์ไปทางขวา `n` ตำแหน่ง (`-len(n)` ≤ `n` ≤ `len(x)`) โดยเอาค่าในตำแหน่งขวาสุดมาต่อในตำแหน่งแรกทางซ้าย แล้วคืนสตริงที่เป็นผลจากการหมุน เช่น `rightRotate([1,2,3,4,5],2)` จะได้ผลลัพธ์ `[4,5,1,2,3]` คำสั่งที่หายไป `(1)` คือข้อใด

```
def rightRotate(x, n):
    _______(1)_________
```

A) `return x[-n:]+x[:-n]`
B) `return x[n:]+x[:n]`
C) `return x[:n]+x[n:]`
D) `return x[-n:]+x[n:]`
E) ไม่มีข้อใดถูก

22. ให้ `x = {1,3,5,9,10,2,0}` ข้อใดเป็นผลของ `x[1:2:3]`

A) `{1,3,5}` B) `{1,2,3}` C) `{1}` D) `{3}` E) ไม่มีข้อใดถูก

23. ให้ `x = {1:4, 5:8, 2:6, 8:2}` ข้อใดเป็นผลของ `sum([e for e in x if x[e]>5])`

A) `6` B) `7` C) `8` D) `14` E) ไม่มีข้อใดถูก

24. ให้ `x = {'B':'C', 'A':'B', 'D':'A', 'C':'D'}`
ข้อใดเป็นผลของ `','.join([x[e] for e in sorted(x.keys())])`

A) `'A,B,C,D'` B) `'C,B,A,D'` C) `'B,A,D,C'` D) `'B,C,D,A'` E) ไม่มีข้อใดถูก

25. ให้ `s1 = {1,2,3,4}` และ `s2 = {2,3,4,5}` คำสั่ง `s1.union(s2)` ทำให้ `s1` เป็นดังข้อใด

A) `{1,2,3,4,5}` B) `{2,3,4}` C) `{1,5}` D) `{1,2,3,4}` E) ไม่มีข้อใดถูก

26. ข้อใดได้ผลเป็น **dict**

A) **{}** B) **dict(1)** C) **empty_dict()** D) **{1,2,3,4}** E) ไม่มีข้อใดถูก

27. ผู้เขียนตั้งใจให้โปรแกรมทางขวานี้รับสตริงทาง keyboard แล้วนับว่าในสตริงนั้นมีตัวเลขแต่ละตัว อยู่กี่ตัว เมื่อนำโปรแกรมนี้ไปทำงานจะทำงานผิดพลาดที่คำสั่งในบรรทัดที่เท่าใร (เมื่อผู้ใช้ป้อน **ho1ho2ho1** เป็น input)

```
c = dict()
for e in input().strip():
  if e in '0123456789' :
    c[e] += 1
```

A) บรรทัดที่ **1** B) บรรทัดที่ **2** C) บรรทัดที่ **3**

D) บรรทัดที่ **4** E) ไม่มีข้อใดถูก

28. พิจารณาฟังก์ชัน **t** ทางขวานี้ คำสั่งในข้อใดทำงานแล้วไม่มีข้อผิดพลาดใด ๆ

```
def t(x):
  for k in range(len(x)):
    if x[k] < 2 : x[k] = 0
```

A) **t([1,2])** B) **t((1,2))** C) **t('12')**

D) **t({1,2})** E) มีมากกว่าหนึ่งข้อที่ทำงานแล้วไม่ผิดพลาด

29. ให้ **x = {1:[2,3,1], 2:[0,1,3], 4:[3,2]}** ข้อใดเป็นผลของคำสั่ง **x[4][1]**

A) **0** B) **1** C) **2** D) **3** E) ไม่มีข้อใดถูก

30. ถ้าเราต้องการให้บริการตอบคำถามว่า เลขท้ายสองตัวของสลากกินแบ่งที่กำหนดให้ ออกรางวัลในงวดใดบ้าง การเก็บข้อมูลในข้อใดทำให้ทำงานได้ตามที่ต้องการ

A) **list** จำนวน **100** ช่อง ใช้ **index** แทนเลขท้ายสองตัว แต่ละช่องเก็บ **set** ของวันเดือนปีที่ออกรางวัล

B) **list** จำนวน **100** ช่อง ใช้ **index** แทนเลขท้ายสองตัว แต่ละช่องเก็บ **list** ของวันเดือนปีที่ออกรางวัล

C) **dict** ที่มี **key** เป็นเลขท้ายสองตัว และ **value** เป็น **set** ของวันเดือนปีที่ออกรางวัล

D) **dict** ที่มี **key** เป็นเลขท้ายสองตัว และ **value** เป็น **list** ของวันเดือนปีที่ออกรางวัล

E) ทุกข้อให้บริการได้ตามโจทย์

31. พิจารณาฟังก์ชัน **c** ทางขวานี้ คำสั่ง **c("123") + c("12341")** จะได้ค่าเท่าไร

```
def c(t):
  s = set()
  for e in t:
    s.add(e)
  return len(s)
```

A) **4** B) **5** C) **6**

D) **7** E) ไม่มีข้อใดถูก

32. ให้ **t** เป็น **dict** ที่มี **key** เป็นชื่อย่อจังหวัด และ **value** แทนจำนวนประชากรในจังหวัด
คำสั่งในข้อใดหาจังหวัดที่มีประชากรมากสุด (กำหนดให้จังหวัดที่มีประชากรมากสุดมีเพียงจังหวัดเดียว)

A) **max(t.values())** B) **max([(t[k],k) for k in t])[-1]**

C) **sorted([(k,t[k]) for k in t])[-1][0]** D) **max([(v,t[v]) for v in t.values()])[1]**

E) ไม่มีข้อใดถูก

33. ให้ **t** เป็น **dict** ที่มี **key** เป็นชื่อคณะ และ **value** แทน **list** ของชื่อเล่นนิสิตในคณะ คำสั่งในข้อใดหาจำนวนชื่อเล่นที่แตกต่างกันทั้งหมดในมหาวิทยาลัย

A) **len(set([e for v in t.values() for e in v]))**
B) **len(set([t[k] for k in t]))**
C) **len([e for k in t.keys() for e in t[k]])**
D) **len(union([t[k] for k in t.keys()]))**
E) ไม่มีข้อใดถูก

34. ต้องการเก็บข้อมูลความแรงของสัญญาณ **WiFi** (เป็นจำนวนเต็ม) ณ ตำแหน่ง ต่าง ๆ ในมหาวิทยาลัย (เป็นจำนวนจริงแลตติจูดและลองติจูด) ควรใช้ที่เก็บข้อมูลแบบใด เพื่อให้สามารถหา ตำแหน่งที่มีสัญญาณ **WiFi** แรงที่สุด **5** ตำแหน่งแรก และที่มีสัญญาณ **WiFi** อ่อนที่สุด **5** ตำแหน่งแรก ได้ง่ายสุด ๆ (ทุกข้อเก็บแต่ละตำแหน่งด้วย **list 2** ช่อง ช่องหนึ่งแลตติจูด อีกช่องลองติจูด)

A) ใช้ **list** เก็บ **tuple (**ความแรง**,**ตำแหน่ง**)** B) ใช้ **dict** เก็บ **key** คือความแรง และ **value** คือตำแหน่ง

C) ใช้ **dict** เก็บ **key** คือตำแหน่ง และ **value** คือความแรง D) ใช้ **set** เก็บ **list [**ความแรง**,** ตำแหน่ง**]**

E) ทำได้ง่ายพอๆ กันมากกว่าหนึ่งข้อ

35. ต้องการเก็บเลขประจำตัวประชาชนจำนวนมาก (เป็นแสนหมายเลข) เพื่อค้น ควรเก็บแบบใด จึงค้นว่ามีเลขประจำตัวประชาชนที่ต้องการอยู่หรือเปล่า ได้เร็วสุดๆ

A) เก็บใน **list** เช่น เก็บในลิสต์ **x** เวลาค้นว่ามีเลข **e** อยู่ใน **x** ไหม ก็ใช้ **if e in x:**

B) เก็บใน **set** เช่น เก็บในเซต **x** เวลาค้นว่ามีเลข **e** อยู่ใน **x** ไหม ก็ใช้ **if e in x:**

C) เก็บใน **value** ของ **dict** เช่น เก็บในดิก **x** เวลาค้นว่ามีเลข **e** อยู่ใน **x** ไหม ก็ใช้ **if e in x.values():**

D) เก็บใน **numpy array** เช่น เก็บในอาเรย์ **x** เวลาค้นว่ามีเลข **e** อยู่ใน **x** ไหม ก็ใช้ **if sum(x[x==e])>0:**

E) ทุกข้อใช้เวลาในการค้นพอๆ กัน

36. ฟังก์ชันทางขวานี้ควรมีชื่อว่าอะไร

```
def _______(x):
  out = []
  for e in x:
    if e <= 0 : out.append(e)
  return out
```

A) **get_non_positive** B) **remove_negative**
C) **min_in_x** D) **do_something**
E) **create_list_with_element_in_x**

37. พิจารณาฟังก์ชัน **f** ทางขวานี้ ข้อใดเป็นคำสั่งที่สั่งทำงานแล้วเกิดความผิดพลาด (ให้ถือว่าได้ทำคำสั่ง **import numpy as np** แล้ว)

```
def f(x):
  for k in range(len(x)):
    if int(x[k])%2 == 0 :
      x[k] = 0
  return x
```

A) **f("1153")** B) **f((5,4,3,2))**
C) **f(['1',2,"2",3])** D) **f(np.array([1,2,4]))**
E) เกิดความผิดพลาดมากกว่าหนึ่งข้อ

38. พิจารณาโปรแกรมทางขวานี้ ผลที่ได้จากคำสั่ง **print(b(4,2))** คือข้อใด

```
def a(g):
  return b(g,g+2)

def b(x,g):
  return 2*x + g

g = 3
print(b(4,2))
print(b(g,2))
print(a(2))
```

A) **2** B) **4** C) **6**
D) **10** E) ไม่มีข้อใดถูก

39. พิจารณาโปรแกรมทางขวานี้ ผลที่ได้จากคำสั่ง **print(b(g,2))** คือข้อใด

A) **2** B) **4** C) **6**
D) **10** E) ไม่มีข้อใดถูก

40. พิจารณาโปรแกรมทางขวานี้ ผลที่ได้จากคำสั่ง **print(a(2))** คือข้อใด

A) **2** B) **4** C) **6**
D) **10** E) ไม่มีข้อใดถูก

recursive

41. ผู้เขียนฟังก์ชัน **e** ทางขวานี้ ตั้งใจให้ **e** คืนสตริงที่กลับลำดับกับสตริงในตัวแปร **x** ข้อใดเมื่อทำงานแล้ว จะไม่ทำงานตามที่ตั้งใจไว้

```
def e(x):
  if len(x)==0 : return x
  return x[-1]+e(x[1:-1])+x[0]
```

A) **e("ABCD")** B) **e("AB")**
C) **e("A")** D) **e("")** E) ทุกข้อกลับลำดับสตริงได้ถูกต้อง

recursive

42. ผู้เขียนฟังก์ชัน **f** ทางขวานี้ ตั้งใจให้ **f** คืนจำนวนฟิโบนักชี ข้อใดเมื่อทำงานแล้ว จะไม่ทำงานตามที่ตั้งใจไว้ (นิยามของ Fibonacci number คือ $f_n = f_{n-1} + f_{n-2}$ if $n \geq 2, f_0 = 0, f_1 = 1$)

```
def f(n):
  if n==1 : return 1
  if n>=2 : return f(n-1)+f(n-2)
  if n==0 : return 0
```

A) **f(0)** B) **f(1)**
C) **f(2)** D) **f(3)** E) ทุกข้อหาจำนวนฟิโบนักชีได้ถูกต้อง

recursive

43. พิจารณาฟังก์ชัน **g** ทางขวานี้ คำสั่งในข้อใด ได้ผลเป็น **7**

```
def g(x):
  if x == 1 : return 0
  return 1 + g(x//2)
```

A) **g(7)** B) **g(8)**
C) **g(100)** D) **g(200)** E) ไม่มีข้อใดถูก

recursive

44. พิจารณาฟังก์ชัน **mm** ทางขวานี้ ผลที่ได้จากคำสั่ง **mm([1, 4, 0, 8, 1, 9, 1, 0])** คือข้อใด

```
def mm(x):
  if len(x)==1: return (x[0],x[0])
  (x1,x2) = mm(x[:len(x)//2])
  (x3,x4) = mm(x[len(x)//2:])
  return (min(x1,x3),max(x2,x4))
```

A) **(0, 9)** B) **(0, 1)**
C) **(8, 9)** D) **(4, 9)**
E) ไม่มีข้อใดถูก

recursive

45. พิจารณาฟังก์ชัน **h** ทางขวานี้ ผลที่ได้จากคำสั่ง **h(3,4)** คือข้อใด

```
def h(a,b):
  if b==0 : return 1
  p = h(a,b//2)
  p *= p
  if b%2==1: p *= a
  return p
```

A) **12** B) **24** C) **27**
D) **90** E) ไม่มีข้อใดถูก

คำถาม 11 ข้อต่อไปนี้ให้ถือว่าได้ทำคำสั่ง **import numpy as np** แล้ว

ให้ **x = np.array([[4,3,2,1], [2,5,0,-4], [4,3,1,3]])** จงตอบ **3** คำถามข้างล่างนี้

46. ข้อใดเป็นผลของ **np.max(x)**

A) **1** B) **2** C) **3** D) **4** E) **5**

47. ข้อใดเป็นผลของ **np.argmax(x[1])**

A) **1** B) **2** C) **3** D) **4** E) **5**

48. ข้อใดเป็นผลของ **np.sum(np.min(x,axis=0))**

A) **1** B) **2** C) **3** D) **4** E) **-2**

49. ข้อใดข้างล่างนี้ได้ผลเหมือนคำสั่งสร้างอาเรย์ **d = np.array([[1,0,0],[0,1,0],[0,0,1]])**

A) **d = np.zero_one((3,3),dtype=int)** B) **d = np.identity(3, dtype=int)**
C) **d = np.diag_one(3, dtype=int)** D) **d = np.arrayone((3,3), dtype=int)**
E) ไม่มีข้อใดถูก

50. ข้อใดข้างล่างนี้ได้ผลเหมือนคำสั่งสร้างอาเรย์ **d = np.array([1.0, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9])**

A) **d = np.range(1.0, 2.0, 0.1)** B) **d = np.xrange(1.0, 2.0, 0.1)**
C) **d = np.arange(1.0, 2.0, 0.1)** D) **d = np.step_range(1.0, 2.0, 0.1)**
E) ไม่มีข้อใดถูก

51. ให้ **d = np.array([[1,2,3,4],[5,6,7,8]])** ข้อใดเป็นผลของ **d.shape**

A) **[4,2]** B) **[2,4]** C) **(4,2)** D) **(2,4)** E) ไม่มีข้อใดถูก

52. ให้ **d** เป็น **numpy array** ขนาด **100×100** คำสั่งในข้อใดทำให้ข้อมูลในแถวแนวนอนหมายเลขคู่ทั้งหมดมีค่าเป็น **0** หมดทั้งแถว

A) **d[::2] = 0** B) **d[:2] = 0**
C) **d[,::2] = 0** D) **d[,:2] = 0** E) ไม่มีข้อใดถูก

53. ให้ **a** และ **b** เป็น **numpy array** ของเลขจำนวนจริงหนึ่งมิติที่มีขนาดเท่ากัน หากเราให้ **a** และ **b** แทนเวกเตอร์ **2** ตัว คำสั่งในข้อใด<u>**ไม่ใช่**</u>การหา **dot product** ของ **vectors** ทั้งสอง

A) **np.sum(a*b)** B) **sum([a[i]*b[i] for i in range(len(a))])**
C) **a.dot(b)** D) **np.dot(b,a)** E) หาได้เหมือนกันหมด

54. ให้ **d = np.array([[9,8,7,6]])** คำสั่งในข้อใด <u>**ไม่ได้**</u> ผลเป็น **numpy array** ขนาด **4×4**

A) **d * d.T** B) **d.T * d**
C) **np.dot(d.T, d)** D) **np.dot(d, d.T)** E) ได้หมดทุกข้อ

55. ให้ **a = np.array([[3,3],[4,4]]; b = np.array([[1,1,1],[2,2,2]])** แทนเมทริกซ์ **2** ตัว คำสั่งในข้อใดคือการคูณเมทริกซ์ **a × b**

A) **a.multiply(b)** B) **np.multiply(a,b)**
C) **a.dot(b)** D) **a.dot(b.T)** E) ได้หมดทุกข้อ

56. ให้ **x = np.array([[1,3,10,5,2,10,8,-8]])** เก็บพิกัด **x** ของจุดต่าง ๆ บนแกน **x** คำสั่งในข้อใดหาระยะทางของคู่จุดที่ห่างกันที่สุด

A) **np.max(x - x.T)** B) **np.max(np.all_pair_distance(x))**
C) **np.argmax(x) - np.argmin(x)** D) **np.max(np.dot(x,x.T))**
E) ไม่มีข้อใดถูก

57. จากคลาส **A** ทางขวานี้ ให้ **x = [A(5), A(8), A(3)]**

ถ้าทำคำสั่ง **x.sort()** ตามด้วย **print(x[0])** จะแสดงค่าอะไร

A) **3** B) **5**

C) **8** D) เขียน **class A** ผิด จึง **sort** ไม่ได้

E) ไม่มีข้อใดถูก

```
class A :
  def __init__(self,a):
    self.a = a
  def __lt__(self,x):
    return x.a < self.a
  def __str__(self):
    return str(self.a)
  def get_a(self):
    return self.a
  def triple(self):
    self.a *= 3
```

58. จากคลาส **A** ในข้อที่แล้ว (โดยตัดเมท็อด **__lt__** และ **__str__** ออก)

ให้ **x = A(99)** คำสั่งในข้อใดได้ผลต่างจากข้ออื่น

A) **x.get_a() *= 3** B) **x.triple()**

C) **x.a *= 3** D) **A.triple(x)** E) ได้ผลเหมือนกันหมดทุกข้อ

59. หากสั่งโปรแกรมข้างล่างนี้ (ทางซ้าย) ให้ทำงาน จะได้ผลดังแสดงทางขวา

```
class A :
  def __init__(self,x):
    self.x = x
for e in [1,2.2,[1],"a",A(1)]:
  print(type(e))
```

```
<class 'int'>
<class 'float'>
<class 'list'>
<class 'str'>
<class '__main__.A'>
```

สรุปได้ว่า ข้อใดข้างล่างนี้**ไม่ใช่**ชื่อคลาสใน **Python**

A) **int** B) **float** C) **str** D) **list** E) เป็นชื่อคลาสทุกข้อ

60. ข้อใดข้างล่างนี้**ไม่ใช่**ชื่อเมท็อด/ฟังก์ชัน ในคลาสที่เรียนในวิชานี้

A) **sort** B) **append** C) **dot** D) **shape** E) เป็นชื่อเมท็อด/ฟังก์ชันทุกข้อ

61. โปรแกรมหนึ่งทำงานได้อย่างถูกต้อง หากเราพบคำสั่ง **a = t.x(y)** ในโปรแกรมนี้ คำว่า **x** ของคำสั่งคืออะไร

A) ชื่อคลาส B) ชื่อเมท็อด C) ชื่อข้อมูลภายใน D) ชื่อตัวแปร E) ไม่มีข้อใดถูก

ศึกษาคลาส **Order** ฟังก์ชัน **get_total** และการทำงานของโปรแกรมข้างล่างนี้ จงตอบคำถาม 5 ข้อต่อไปนี้

```
class Order:
    def __init__(self):
        self.orderlines = []

    def add(self,name,price):
        self.orderlines.append( (name,price) )   # list of tuples

    def total(self):
        return sum([p for (n,p) in self.orderlines])

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

def get_total(orders):
    total = 0
    for od in orders:
        total += od.total()
    return total

o1 = Order(); o1.add("Congee",30); o1.add("Fried Rice",45); o1.add("Water",7)
o2 = Order(); o2.add("Papaya Salad",40); o2.add("Congee",30)
orders = [o1,o2,o1,o1]
```

62. หลังจากโปรแกรมข้างบนนี้ทำงาน ถ้าให้ทำคำสั่ง **o1.orderlines[0][1]** จะคืนผลอะไร

A) 7 B) 30 C) 40 D) 45 E) ไม่มีข้อใดถูก

63. หลังจากโปรแกรมข้างบนนี้ทำงาน ถ้าให้ทำคำสั่ง **o2.orderlines[1][0]** จะคืนผลอะไร

A) "Papaya Salad" B) "Fried Rice" C) "Water" D) "Congee" E) ไม่มีข้อใดถูก

64. หลังจากโปรแกรมข้างบนนี้ทำงาน ถ้าให้ทำคำสั่ง **`o1.total()`** จะคืนผลอะไร

A) 70 B) 75 C) 82 D) 152 E) ไม่มีข้อใดถูก

65. หลังจากโปรแกรมข้างบนนี้ทำงาน ถ้าให้ทำคำสั่ง **`o1 < o2`** จะคืนผลอะไร

A) true B) false C) True D) False E) ไม่มีข้อใดถูก

66. หลังจากโปรแกรมข้างบนนี้ทำงาน ถ้าให้ทำคำสั่ง **`get_total(orders[2:3])`** จะคืนผลอะไร

A) 70 B) 82 C) 152 D) 232 E) ไม่มีข้อใดถูก

ศึกษาคลาส **`Menu`** คลาส **`Order`** ฟังก์ชัน **`get_total`** และการทำงานของโปรแกรมข้างล่างนี้ จงตอบคำถาม 5 ข้อต่อไปนี้

```
class Menu:
    def __init__(self, name, price):
        self.name = name
        self.price = price

class Order:
    def __init__(self, date):
        self.date = date
        self.orderlines = []

    def add(self, menu, n):
        for i in range(n):
            self.orderlines.append(menu)

    def total(self):
        return sum([menu.price for menu in self.orderlines])

def get_total(orders, date):
    return sum([od.total() for od in orders if od.date == date])

m = [ Menu("fried rice", 45), Menu("phat thai", 50),
      Menu("Congee", 30), Menu("papaya salad", 40) ]
o1 = Order("1/03/2016"); o1.add(m[0],2); o1.add(m[3],1)
o2 = Order("1/03/2016"); o2.add(m[1],2); o2.add(m[0],1)
o3 = Order("2/03/2016"); o3.add(m[1],1); o3.add(m[2],1)
o4 = Order("2/03/2016"); o4.add(m[2],5)
orders = []
orders.append(o1); orders.append(o2);
orders.append(o3); orders.append(o4)
```

67. หลังจากโปรแกรมข้างบนนี้ทำงาน ถ้าให้ทำคำสั่ง **`o3.orderlines[1].price`** จะคืนผลอะไร

A) 30 B) 40 C) 45 D) 50 E) ไม่มีข้อใดถูก

68. หลังจากโปรแกรมข้างบนนี้ทำงาน ถ้าให้ทำคำสั่ง **`o2.orderlines[1].price`** จะคืนผลอะไร

A) 30 B) 40 C) 45 D) 50 E) ไม่มีข้อใดถูก

69. หลังจากโปรแกรมข้างบนนี้ทำงาน ถ้าให้ทำคำสั่ง **`o1.total()`** จะคืนผลอะไร

A) 85 B) 95 C) 130 D) 140 E) ไม่มีข้อใดถูก

70. หลังจากโปรแกรมข้างบนนี้ทำงาน ถ้าให้ทำคำสั่ง **`m[0].price=40`** แล้วคำสั่ง **`o1.total()`** จะคืนผลอะไร

A) 80 B) 90 C) 120 D) 130 E) ไม่มีข้อใดถูก

71. หลังจากโปรแกรมข้างบนนี้ทำงาน ถ้าให้ทำคำสั่ง **`get_total(orders, "2/03/2016")`** จะคืนผลอะไร

A) 85 B) 165 C) 230 D) 275 E) ไม่มีข้อใดถูก

ศึกษาคลาส **Food** และการทำงานของโปรแกรมข้างล่างนี้

```
class Food:
    def __init__(self, name, c):
        self.name = name
        self.calories = c

    def __lt__(self, rhs):
        return False

    def cal(self):
        return self.calories

    def mix_with(self, other) :
        return Food(self.name+"+"+other.name, self.calories+other.calories)

    def __str__(self):
        return self.name+":"+str(self.calories)

def bubble_sort(x):
    for k in range(len(x)-1):
        for i in range(len(x)-1):
           if x[i+1] < x[i] : x[i],x[i+1] = x[i+1],x[i]

f = [Food('A',30), Food('B',10), Food('C',10)]
```

72. ถ้าทำคำสั่ง **print(sum([e.calories for e in f]))** จะแสดงอะไรทางจอภาพ _____

73. แล้วก็ทำคำสั่ง **print(f[0] < f[2])** จะแสดงอะไรทางจอภาพ _____

74. แล้วก็ทำคำสั่ง **print(f[0].mix_with(f[2]))** จะแสดงอะไรทางจอภาพ _____

75. แล้วก็ทำคำสั่ง **print(f[0].cal())** จะแสดงอะไรทางจอภาพ _____

76. แล้วก็ทำคำสั่ง **bubble_sort(f)** ตามด้วย **print(f[0],f[1],f[2])** จะแสดงอะไรทางจอภาพ

_____

(25 คะแนน) ธนาคารออมสินจำหน่ายสลากออมสิน ซึ่งไม่เหมือนสลากกินแบ่งรัฐบาล สลากกินแบ่งรัฐบาลหนึ่งใบมีหนึ่งหมายเลข ในขณะที่สลากออมสินหนึ่งใบประกอบด้วย *ประเภทสลาก หมายเลขเริ่มต้น* และ*หมายเลขสิ้นสุด* สลากแต่ละประเภท (เช่น ประเภท **3** ปี ประเภท **5** ปี พิเศษ...) ออกเลขรางวัลและเงินรางวัลไม่เหมือนกัน

ข้างล่างนี้แสดงคลาส **Lottery_Banker** และคลาส **Lottery_Ticket** (อ่านคำอธิบายของแต่ละคลาสจาก **comment** ที่เขียน) ธนาคารออมสินก็เป็นออบเจ็กต์ของคลาส **Lottery_Banker** และสลากออมสินแต่ละใบก็เป็นออบเจ็กต์ของคลาส **Lottery_Ticket** **Lottery_Banker** มีหน้าที่เก็บและคืนผลการออกรางวัลในแต่ละประเภท ส่วน **Lottery_Ticket** มีเมท็อดให้บริการเพื่อตอบว่าสลากใบหนึ่งถูกรางวัลอะไรบ้าง (**winning_results**) ได้เงินรางวัลรวมเท่าไร (**total_price**) เป็นต้น

จงเติมคำสั่งในเมท็อด **get_results** ของคลาส **Lottery_Banker** และในเมท็อด **winning_results, total_price** และ **__lt__** ของคลาส **Lottery_Ticket** ให้ทำงานตาม **comment** ที่เขียน (เมท็อดอื่นที่ได้เขียนคำสั่งไว้ ทำงานถูกต้องแล้ว)
ดูตัวอย่างการใช้งานข้างล่างนี้ประกอบ

```
gsb = Lottery_Banker()
t1  = Lottery_Ticket(gsb, '5yrs', '0000000', '0000000')
t2  = Lottery_Ticket(gsb, '5yrs', '0311100', '0322999')
gsb.add_result('5yrs',('11187',300)) # เลขท้ายห้าตัว 11187 รางวัล 300 บาท
gsb.add_result('5yrs',('2189',150))  # เลขท้ายสี่ตัว 2189 รางวัล 150 บาท
r2 = gsb.get_results('5yrs')         # ได้ [('11187',300), ('2189',150)]
r1 = gsb.get_results('100yrs')       # ได้ [ ]
w2 = t2.winning_results()            # ได้ [('11187',300,1),('2189',150,2)]
w1 = t1.winning_results()            # ได้ [ ]
p2 = t2.total_price()                # ได้ 600 บาท
p1 = t1.total_price()                # ได้ 0 บาท
```

ให้สังเกตว่า ตอนที่สร้างสลาก เราต้องส่งออบเจ็กต์ของ **Lottery_Banker** ไปด้วย

```
class Lottery_Banker :                  # คลาสสำหรับผลิตผู้รับผิดชอบการเก็บผลการออกเลขสลากที่ถูกรางวัล
    def __init__(self):
        self.results = dict()    # key คือ ประเภทสลาก, value คือ list เก็บผลการออกสลากที่ถูกรางวัลของประเภทนั้น
                                 # ผลการออกสลากที่ถูกรางวัลเป็น tuple (เลขที่ออก, เงินรางวัล)
    def add_result(self, ltype, result):    # เพิ่มผลรางวัลการออกสลาก result ของประเภท ltype
                                            # result คือ tuple (เลขที่ออก,เงินรางวัล)
        if ltype not in self.results:
            self.results[ltype] = list()
        self.results[ltype].append(result)
    def get_results(self, ltype) :          # คืนรายการของผลการออกสลากที่ถูกรางวัลทั้งหมดของประเภท ltype
                                            # (คำเตือน : สลากประเภท ltype อาจยังไม่เคยออกรางวัลเลยก็ได้)
                                            # ดูตัวอย่างข้างบนประกอบ
#-----------------------------------------------------------------------
class Lottery_Ticket :                  # คลาสสำหรับผลิตสลากหนึ่งใบ ภายในประกอบด้วย
    def __init__(self, bk, ltype, b, e):
        self.banker = bk                # ผู้รับผิดชอบการเก็บผลการออกเลขสลากที่ถูกรางวัล
        self.ltype = ltype              # ประเภทสลาก
        self.begin = b                  # เลขสลากเริ่มต้น
        self.end = e                    # เลขสลากสุดท้าย
    def number_of_winnings(self, x):    # คืนจำนวนหมายเลขที่ถูกรางวัลของสลากใบนี้ ถ้าเลขท้ายที่ออกคือ x
        a = self.begin; b = self.end    # x เป็นเลขกี่หลัก ก็ตรวจเลขท้ายเท่านั้นหลัก
        nwins = 0                       # เช่น สลากใบนี้มีเลข 001000 - 002999 ทั้งหมด 2000 หมายเลข
        n = len(x)                      # ถ้าเลขท้ายสองตัวที่ออกคือ 77 ก็ย่อมถูกรางวัล 20 หมายเลข
        if a[-n-1:-n] == b[-n-1:-n] :
            if a[-n:] <= x <= b[-n:] : nwins += 1
        else:
            if a[-n:] <= x : nwins += 1
            if x <= b[-n:] : nwins += 1
            nwins += int(b[:-n]) - int(a[:-n]) - 1
        return nwins
```

```
    def winning_results(self):          # คืนรายการของผลการออกรางวัลที่สลากใบนี้ได้รางวัล
        out = []                        # คืน list ของ tuple (เลขท้าย,เงินรางวัล,จำนวนหมายเลขที่ถูกรางวัล)
                                        # ดูตัวอย่างในหน้าที่แล้วประกอบ
```



```
        return out
    #-------------------------------------------------------------------------

    def total_price(self):              # คืนเงินรางวัลทั้งหมดที่สลากใบนี้ได้รางวัล
        total = 0                       # ดูตัวอย่างในหน้าที่แล้วประกอบ
```



```
        return total
    #-------------------------------------------------------------------------

    def __lt__(self, rhs):              # มีไว้ใช้เปรียบเทียบสลาก โดยใช้เงินรางวัลรวมที่ได้ของสลากเป็นตัวเปรียบเทียบ
```

โปรแกรมต่าง ๆ ในหน้านี้ ถือว่าได้เขียน **import numpy as np** ไว้แล้ว

(**5** คะแนน) ให้ **g** เป็น **numpy array** ที่ **g[i,j]** เก็บเกรด (ซึ่งเป็นเลข **0,1,2,3,4)** ของนิสิตคนที่ **i** ที่เรียนวิชา **j** นั่นหมายความว่า **g.shape[0]** คือจำนวนนิสิต และ **g.shape[1]** ก็คือจำนวนวิชา (ให้ถือว่านิสิตทุกคนใน **array** เรียนทุกวิชา ใน **array)** และให้ **c** เป็น **numpy array** ที่ **c[j]** เก็บจำนวนหน่วยกิตของวิชาที่ **j** จงเขียนฟังก์ชัน **get_student_GPA(g,c)** เพื่อคำนวณหาเกรดเฉลี่ยของนิสิตแต่ละคน คืนกลับมาเป็น **numpy array** มิติเดียว โดยช่อง **i** เก็บเกรดเฉลี่ยของนิสิตคนที่ **i**
ตัวอย่างเช่น ถ้า **g = np.array([[1,1,1,1,1],[4,0,0,0,4],[4,4,4,4,4]])** และ **c = np.array([3,3,3,4,1])** จะได้ว่า **get_student_GPA(g,c)** คืนผลลัพธ์เป็น [1.0**, 1.14285714,** 4.0]
(**1.14285714 = (12+0+0+0+4)/(3+3+3+4+1))** อนุญาตให้เขียนลงในช่องว่างที่ขีดเส้นใต้เท่านั้น และห้ามใช้คำสั่ง **dot**

```
def get_student_GPA(g,c):
    return _____________________________________
```

(**5** คะแนน) จงเขียนฟังก์ชัน **get_student_GPA(g,c)** ที่ทำงานเหมือนข้อที่แล้ว แต่ในข้อนี้ อนุญาตให้เขียนลงในช่องว่างที่ขีดเส้นใต้เท่านั้น และให้ใช้คำสั่ง **dot** เป็นส่วนหนึ่งของการคำนวณ

```
def get_student_GPA(g,c):
    return _____________________________________
```

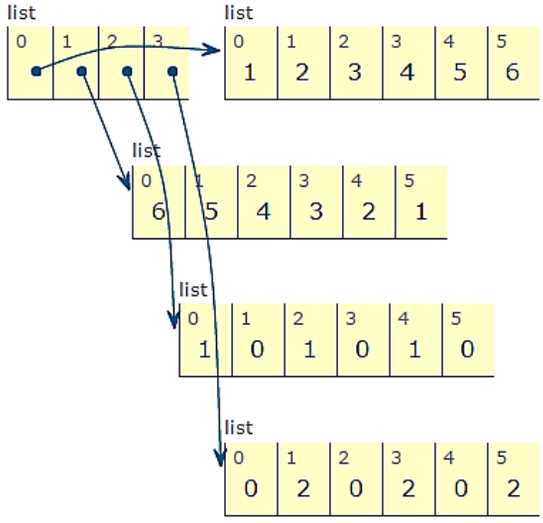
(**5** คะแนน) จาก **g** และ **c** ในข้อที่แล้ว จงเขียนฟังก์ชัน **get_course_GPA(g,c)** เพื่อคำนวณหาเกรดเฉลี่ยของแต่ละวิชา คืนกลับมาเป็น **numpy array** มิติเดียว โดยช่อง **j** เก็บเกรดเฉลี่ยของวิชาที่ **j** ตัวอย่างเช่น ถ้า **g = np.array([[1,1,1,1,1], [4,0,0,0,4],[4,4,4,4,4]])** และ **c = np.array([3,3,3,4,1])** จะได้ว่า **get_course_GPA(g,c)** คืนผลลัพธ์เป็น [**3.0, 1.66666667**, **1.66666667**, **1.66666667**, **3.0**] (เกรดเฉลี่ยของวิชาหนึ่ง คำนวณจากผลรวมเกรดที่ให้นิสิตทั้งหมดในวิชานั้น หารด้วยจำนวนนิสิต) อนุญาตให้เขียนลงในช่องว่างที่ขีดเส้นใต้เท่านั้น

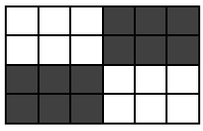
```
def get_course_GPA(g,c):
    return _____________________________________
```

(**10** คะแนน) เราแทนตารางสองมิติได้ด้วย **list of lists** ดังแสดงตัวอย่างในรูปข้างล่างนี้

1	2	3	4	5	6
6	5	4	3	2	1
1	0	1	0	1	0
0	2	0	2	0	2

```
a = \
 [[1,2,3,4,5,6],\
  [6,5,4,3,2,1],\
  [1,0,1,0,1,0],\
  [0,2,0,2,0,2] ]
```





เช่น อาเรย์ขนาด **4×6**

เราสามารถแบ่งตาราง **2** มิติ ที่มีจำนวนแถวและจำนวนหลักเป็นจำนวนคู่ ออกเป็นสี่ส่วนเท่าๆ กัน ดังรูปข้างบนนี้
จงเขียนฟังก์ชันที่รับตารางสองมิติ ที่มีจำนวนแถวและจำนวนหลักเป็นจำนวนคู่ (เช่น **4×6**, **10×200** เป็นต้น) เพื่อหาผลรวมของจำนวนที่เก็บตามช่องในส่วนจตุภาคที่ **1** กับ **3** (ที่แสดงด้วยสีดำในรูปข้างบนนี้)

```
def sum_Q1_and_Q3_list( a ): # a is a list of lists
```


(**10** คะแนน) เราสามารถแทนตารางสองมิติด้วยอาเรย์ของ **numpy** ได้เช่นกัน จงเขียนฟังก์ชัน **sum_Q1_and_Q3** ที่มีการทำงานตามที่อธิบายในข้อที่แล้ว แต่คราวนี้ รับตาราง **a** ที่เก็บด้วยอาเรย์สองมิติของ **numpy** (โดยไม่ใช้ลูป และมีขั้นตอนการทำงานที่ใช้คุณสมบัติของ **numpy** ที่ทำงานได้รวดเร็วให้มากที่สุด ด้วยจำนวนคำสั่งน้อยสุด ๆ)

```
import numpy as np
def sum_Q1_and_Q3_numpy( a ): # a is a numpy array
```

คำสั่งพื้นฐาน

`x = int(input())` รับข้อมูลจำนวนเต็ม 1 จำนวน

`x,y,z = [float(e) for e in input().split()]`
รับข้อมูลจำนวนจริง 3 จำนวนในบรรทัดเดียวกัน คั่นด้วยช่องว่าง

`int(x), float(x), str(x):`
คืนค่า x ที่ถูกเปลี่ยนประเภทข้อมูลเป็นจำนวนเต็ม จำนวนจริง และสตริง

`abs(n):` คืนค่าสัมบูรณ์ของ n

`round(f):` คืนค่าจำนวนเต็มที่เกิดจากการปัดเศษจำนวนจริง f
โดยถ้าเศษของ f มีค่าตั้งแต่ 0.5 จะปัดขึ้น ถ้าน้อยกว่า 0.5 จะปัดลง

`round(f,d):` คืนค่าจำนวนจริงที่เกิดจากการปัดเศษจำนวนจริง f
โดยปัดให้มีจำนวนตัวเลขหลังจุดทศนิยม d หลัก

`range(start,stop [,step])` หรือ `range(stop):`
คืนค่าเป็นลิสต์ของตัวเลขตามลำดับตั้งแต่ start ถึง stop-step และเพิ่มขึ้นทีละ step (ถ้าไม่ระบุ start จะมีค่า 0 และ step จะมีค่า 1)

`enumerate(L):` คืนลิสต์ของ tuple (index, element) ของแต่ละข้อมูลในลิสต์ L

`len(a):` คืนค่าเป็นจำนวนข้อมูลใน a ซึ่ง a อาจเป็นลิสต์ ดิกชันนารี เซต ทูเปิล สตริง หรือ numpy array ก็ได้

`max(a), min(a):` คืนค่าที่มาก/น้อยที่สุดของข้อมูลใน a ซึ่ง a อาจเป็นลิสต์ ดิกชันนารี เซต ทูเปิล หรือสตริงก็ได้
(numpy array ใช้ np.max(a), np.min(a))
ถ้า a เป็นดิกชันนารี จะคืนค่ามาก/น้อยที่สุดของ key ของดิกชันนารี

`type(a):` คืนค่าประเภทของ a เช่น type([1,2]) ได้ <class 'list'>

`list(), dict(), tuple(), set():`
สร้างลิสต์ว่าง ดิกชันนารีว่าง ทูเปิลว่าง เซตว่าง

list L

`L.append(e):` เพิ่ม e ไปที่ท้ายลิสต์ L

`L.insert(index,e):` เพิ่ม e ไปที่ตำแหน่ง index ในลิสต์ L

`L.pop(index):` ลบข้อมูลที่ตำแหน่ง index และคืนค่าข้อมูลที่ถูกลบ

`L.count(e):` คืนจำนวนครั้งที่ e ปรากฏในลิสต์ L

dict D

`D.items():` คืนลิสต์ของ tuple (key, value) ของดิกชันนารี D

`D.keys():` คืนลิสต์ของ key ทั้งหมดของดิกชันนารี D

`D.values():` คืนลิสต์ของ value ทั้งหมดของดิกชันนารี D

`D.pop(k):` ลบข้อมูลใน D ที่มี key เป็น k และคืนค่า value ของ key นั้น

`D.update(D1):` เพิ่มข้อมูลจากดิกชันนารี D1 เข้าไปใน D

set S

`S.add(e):` เพิ่ม e ในเซต S

`S.difference(T):` คืนเซตใหม่ที่เท่ากับ S-T

`S.discard(e):` ลบ e ออกจากเซต S ถ้าไม่มี e ใน S ก็ไม่ทำอะไร

`S.intersection(T):` คืนเซตใหม่ที่เท่ากับ S ∩ T

`S.union(T):` คืนเซตใหม่ที่เท่ากับ S U T

`S.issubset(T):` ทดสอบว่า S ⊆ T หรือไม่

`S.issuperset(T):` ทดสอบว่า S ⊇ T หรือไม่

`S.pop():` ลบข้อมูลหนึ่งตัวออกจากเซต S และคืนข้อมูลที่ถูกลบ

`S.update(T):` ให้ S = S U T

import math

`math.exp(x):` คืนค่า e ยกกำลัง x

`math.cos(x):` คืนค่า cosine ของ x เรเดียน

`math.sin(x):` คืนค่า sine ของ x เรเดียน

`math.sqrt(x):` คืนค่ารากที่สองของ x

`math.log(x,base):` คืนค่าลอการิทึมของ x ฐาน base

`math.degrees(x):` แปลงมุม x จากเรเดียนเป็นองศา

`math.radians(x):` แปลงมุม x จากองศาเป็นเรเดียน

`math.pi, math.e:` ค่าคงที่ pi และ e

string s

`s.lower():` คืนสตริงใหม่ที่มีค่าเหมือน s แต่เป็นตัวพิมพ์เล็กทั้งหมด

`s.upper():` คืนสตริงใหม่ที่มีค่าเหมือน s แต่เป็นตัวพิมพ์ใหญ่ทั้งหมด

`s.find(sub):` คืน index แรกสุดที่พบ sub ใน s ถ้าไม่พบคืนค่า -1

`s.find(sub,i):` คืน index แรกสุดที่พบ sub ใน s โดยเริ่มค้นที่ index i

`s.count(sub):` คืนจำนวนครั้งที่ sub ปรากฏในสตริง s

`s.split(sep):` คืนลิสต์ของสตริงที่แยกด้วย sep (หรือ space ถ้าไม่ระบุ)

`s.strip():` คืนสตริงใหม่ที่มีค่าเหมือน s แต่ตัด spaces หัวท้ายออก

`s.join(L):` คืนสตริงที่สร้างจากการนำแต่ละ element ในลิสต์ L มาต่อกัน โดยมี s เป็นตัวคั่นระหว่างข้อมูลที่ต่อกัน (L ต้องเป็นลิสต์ของสตริง)

import numpy as np

`np.array(L):` คืนค่า numpy array ที่สร้างจากลิสต์ L

`np.arange(start,stop,step):` คืนอาเรย์ 1 มิติของจำนวนที่มีค่าตาม start,stop,step

`np.ones(shape):` คืนอาเรย์ที่มีค่า 1 ทั้งหมด มีขนาดตาม tuple shape

`np.zeros(shape):` คืนอาเรย์ที่มีค่า 0 ทั้งหมด มีขนาดตาม tuple shape

`np.identity(size):` คืนอาเรย์ขนาด size x size ซึ่งมีข้อมูลในแนวเส้นทแยงมุมเป็น 1 และค่าในตำแหน่งอื่น ๆ เป็น 0

`np.empty_like(a):` คืนอาเรย์ใหม่ที่มีขนาดเหมือน a แต่ไม่มีการกำหนดค่าข้อมูลในอาเรย์ใหม่นี้

`np.add(a,b), np.subtract(a,b), np.multiply(a,b), np.divide(a,b):` คืนค่าอาเรย์ใหม่ที่เป็นผลบวกลบคูณหารแบบช่องต่อช่องของ a และ b

`np.dot(a,b):` คืนอาเรย์ที่เป็นผลคูณแบบเมทริกซ์ของ a และ b

`np.sin(a), np.cos(a), np.sqrt(a), np.abs(a):` คืนอาเรย์ที่มีค่าของข้อมูลในแต่ละตำแหน่งเป็นผลจากการเรียกฟังก์ชัน sine, cosine, sqrt, abs กับข้อมูลในอาเรย์ a ที่ตำแหน่งเดียวกัน

`np.max(a,axis), np.min(a,axis):` คืนอาเรย์ของค่ามาก/น้อยที่สุดใน a ตาม axis ที่กำหนด

`np.argmax(a,axis), np.argmin(a,axis):` คืนอาเรย์ของ index ที่มีค่ามาก/น้อยที่สุดใน a ตาม axis ที่กำหนด ตัวอย่างเช่น

```
a = np.array([ [2, 4, 6], [8, 10, 12] ])
np.max(a) คืนค่า 12, np.argmax(a) คืนค่า 5
np.max(a,axis=0)     คืนค่า array([8,10,12])
np.argmax(a,axis=0) คืนค่า array([1,1,1])
np.argmax(a,axis=1) คืนค่า array([2,2])
```

`np.sum(), np.std(), np.mean():` มีการใช้งานเหมือน np.max()

`np.ndenumerate(a):` คืนลิสต์ของ tuple (position,element) ของแต่ละข้อมูลใน a โดย position เป็น tuple ที่เก็บตำแหน่งของข้อมูล