Mahidol University

Wisdom of the Land

Mahidol University

Lecture 07: List

Lecturers:

Aj. Jidapa Kraisangka

Aj. Akara Supratak

Aj. Tipajin Thaipisutikul

RECAP: Repetition/ Loop

- **`for:`** ใช้เมื่อทราบจำนวนรอบในการ loop อย่างแน่นอน
- **`while:`** ใช้เมื่อจำนวนรอบในการ loop ไม่ทราบล่วงหน้า หรือทราบล่วงหน้าก็ได้ แล้วแต่ style การเขียน code

```
i = 1    initialization
while i<=10:    expression
    print(i, end=' ')
    i +=1 alteration
```

Generate a sequence of number from 0 to 9

```
for x in range(10):
    print(x, end=' ')
```

RECAP: Control Statement

- **`break()`** และ **`continue()`**
- ใช้ภายใน **`while`** และ **`for`** loop

```
while expression:
        if expression:
                break
        statement1
        statement2
        …
statement n
```

หยุดการทำงานของ loop ทันที

```
while expression:
        if expression:
                continue
        statement1
        statement2
        …
statement n
```

ข้าม statement ถัดๆมาใน loop และไปทำงานในรอบใหม่ทันที

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

Nested Loop

Loop สามารถใส่ซ้อนกันได้ (loop ซ้อน loop)

```
for/while:
    for/while:
        statement(s)
        for/while:
            statement(s)
```

```
for x in range(...):
    for y in range(...):
        statement(s)
```

```
while expression1:
    while expression2:
        statement(s)
```

```
while expression1:
    for x in range(...):
        statement(s)
```

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

- List และคำสั่งพื้นฐาน
  - การสร้าง List
  - การเข้าถึงข้อมูล/ช่วงข้อมูลใน List
  - การเพิ่ม/ลบ/แก้ไข ข้อมูลใน List
- List Iteration
- Multi-dimensional List (2D)

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

Mahidol University
Wisdom of the Land

Mahidol University

List และคำสั่งพื้นฐาน

```
mylist = ['pen', 'pineapple', 'apple', 'pen']
```

- โครงสร้างข้อมูลชนิดหนึ่ง (built-in data type) ที่ใช้เก็บข้อมูลแบบลำดับ (sequence)
- สามารถใช้ List เพื่อเก็บข้อมูลจำนวนมาก และหลากหลายประเภท (เช่น integer, string, object) ในเวลาเดียวกัน
- ข้อมูลใน list สามารถถูกแก้ไขเปลี่ยนแปลงได้ (mutable)
- สามารถเก็บข้อมูลซ้ำ (duplicate value) ได้
- ความยาว (length) ของ list สามารถเปลี่ยนแปลง ได้โดยไม่ต้องมีการประกาศล่วงหน้า

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

การสร้าง List

การสร้าง list นั้นข้อมูลของ list จะอยู่ภายในเครื่องหมาย **[ ]** และสมาชิก (item) แต่ละตัวด้วยเครื่องหมาย **,**

การสร้าง list เปล่าที่ยังไม่มีข้อมูล

```
# การสร้าง List โดย [ ]
list1 = []
# หรือ ใช้คำสั่ง list()
list1 = list()
```

```
# List ของ integers
list1 = [1, 2, 3, 4, 5]

# List ของ string
list2 = ['a', 'b', 'c', 'd', 'a']

# List ของ integers, string และ list
list3 = [1, 2, 'b', 'c', list1]
```

Item ใน list สามารถซ้ำกันได้

Item ใน list สามารถมีได้หลายประเภท

การเข้าถึงข้อมูลใน List

- List ใช้ index สำหรับการเข้าถึงข้อมูลแต่ละตัว
- หากมีข้อมูล n จำนวน เริ่มจากลำดับแรกด้วย index 0 จนถึงข้อมูลลำดับสุดท้ายคือ n-1

```
          [0]       [1]          [2]       [3]      index
mylist = ['pen', 'pineapple', 'apple', 'pen']
```

list1[0] มีค่าเท่ากับ 'pen'
list1[1] มีค่าเท่ากับ _____
list1[4] มีค่าเท่ากับ _____

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

การเข้าถึงข้อมูลใน List

- List ใช้ index สำหรับการเข้าถึงข้อมูลแต่ละตัว
- หากมีข้อมูล n จำนวน เริ่มจากลำดับแรกด้วย index 0 จนถึงข้อมูลลำดับสุดท้ายคือ n-1

```
          [0]        [1]          [2]         [3]      index
  list1 = ['pen', 'pineapple', 'apple', 'pen']
```

list1[0] มีค่าเท่ากับ 'pen'
list1[1] มีค่าเท่ากับ _____
list1[4] มีค่าเท่ากับ _____

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

การเข้าถึงข้อมูลใน List

- Index สามารถติดลบได้ โดยเริ่มนับจากข้อมูลสุดท้ายของ list
  index -1 คือ สมาชิกตัวสุดท้ายของ list
  index -n คือ สมาชิกตัวแรกของ list

```
          [0]        [1]          [2]        [3]      index
  list1 = ['pen', 'pineapple', 'apple', 'pen']
          [-4]       [-3]         [-2]       [-1]     index
```

```
list1[-3] มีค่าเท่ากับ 'pineapple'          #list1[1]
list1[-2] มีค่าเท่ากับ _____________        #list1[____]
list1[-1] มีค่าเท่ากับ _____________        #list1[____]
```

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

การเข้าถึงข้อมูลใน List

- Expression ใดๆที่ค่าของผลลัพธ์เป็นจำนวนเต็ม สามารถใช้เป็น index ในการเข้าถึงข้อมูลใน list ได้ โดยผลลัพธ์ของ Expression จะต้องอยู่ภายในขอบเขตของ index ของ list (ห้ามเกิน n)

```
list1 = ['pen', 'pineapple', 'apple', 'pen']
```

```
n=4
list1[n%2]      มีค่าเท่ากับ 'pen'
list1[n-8]      มีค่าเท่ากับ _____
list1[-n]       มีค่าเท่ากับ _____
```

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

การเข้าถึงช่วงข้อมูลใน List

- การเข้าถึงช่วงของข้อมูลใน list สามารถทำได้ โดยการระบุ index เริ่มต้นและ index สุดท้าย (ไม่รวม) ของช่วงนั้น ๆ

```
list[start:end]
```

```
list2 = ['a', 'b', 'c', 'd', 1, 2, 3, 4]
```

`print(list2[0:2])`	ผลลัพธ์ คือ	`['a', 'b']`
`print(list2[2:6])`	ผลลัพธ์ คือ	_____
`print(list2[7:8])`	ผลลัพธ์ คือ	_____

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

การเข้าถึงช่วงข้อมูลใน List

- ถ้า **`start`** ไม่ถูกระบุ ข้อมูลจะเริ่มต้นที่ข้อมูลแรกใน list
- ถ้า **`end`** ไม่ถูกระบุ, range จะจบที่ข้อมูลสุดท้ายของ list

```
list2 = ['a', 'b', 'c', 'd', 1, 2, 3, 4]
```

```
print(list2[:2])
```
ผลลัพธ์ คือ `['a', 'b']`

```
print(list2[2:])
```
ผลลัพธ์ คือ `['c', 'd', 1, 2, 3, 4]`

```
print(list2[:4])
```
ผลลัพธ์ คือ _____

```
print(list2[4:])
```
ผลลัพธ์ คือ _____

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

การเพิ่มข้อมูลใส่ List

- 3 วิธีในการเพิ่มข้อมูลใส่ list
  - **`append()`:** ใส่ข้อมูลหนึ่ง item ที่ท้าย list
  - **`insert()`:** ใส่ข้อมูลหนึ่ง item ที่ตำแหน่งที่กำหนด
  - **`extend()`:** ใส่ข้อมูลหลาย item พร้อมๆกันที่ท้าย list

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

Adding Items: **append()**

- **append() :** ใส่ข้อมูลหนึ่ง item ที่ท้าย list

Syntax `list.append(item)`

```
list1 = []
list1.append(5)
list1.append(1)
list1.append(3)
print(list1)
```

```
[5, 1, 3]
```

```
list1 = [2, 3]
list1.append(4)
list1.append('x')
list1.append(1.5)
print(list1)
```

```
[2, 3, 4, 'x', 1.5]
```

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

Adding Items: `insert()`

- **`insert()`**: ใส่ข้อมูลหนึ่ง item ที่ตำแหน่งที่กำหนด

Syntax `list.insert(index, item)`

```
list1 = [1, 2, 3, 4]
list1.insert(3, 5)
list1.insert(0, 6)
list1.insert(1, 7)
print(list1)
```

```
[1, 2, 3, 5, 4]
[6, 1, 2, 3, 5, 4]
[6, 7, 1, 2, 3, 5, 4]
```

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

Adding Items: `extend()`

- **`extend()`**: ใส่ข้อมูลหลาย item พร้อมๆกันที่ท้าย list

Syntax `list.extend(another_list)`

```
list1 = [1, 2, 3]
list1.extend(['a', 'b', 'c'])
list2 = [4, 5]
list2.extend(list1)
print(list1)
print(list2)
```

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

```
[4, 5, 1, 2, 3, 'a', 'b', 'c']
```

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

การลบข้อมูลใน List

- 3 วิธีในการลบข้อมูลจาก list
  - **`remove()`**: ลบข้อมูลที่ระบุจาก list
  - **`pop()`**: ลบและส่งข้อมูลตำแหน่งท้ายสุดของ list หรือข้อมูลในตำแหน่งที่ระบุ
  - **`clear()`**: ลบข้อมูลทั้งหมดจาก list

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

Removing Items: `remove()`

Syntax `list.remove(item)`

- **`remove()`**: ลบข้อมูลที่ระบุจาก list
  - จะเกิด error ถ้าข้อมูลที่ระบุไม่มีใน list
  - ถ้ามีข้อมูลซ้ำใน list ข้อมูล item แรกที่ถูกพบใน list จะถูกลบ

```
list1 = ['A', 'B', 99, 'd', 'e', 6, 10, 'k', 6]
list1.remove('k')
list1.remove(3)
list1.remove(6)
```

```
['A', 'B', 99, 'd', 'e', 6, 10, 6]
ValueError: list.remove(x): x not in list
['A', 'B', 99, 'd', 'e', 10, 6]
```

Removing Items: **pop()**

- **pop()**: ลบ และส่งค่า return ข้อมูลในตำแหน่งที่ระบุ
  - ถ้าไม่ได้ระบุตำแหน่ง จะลบและ (return) ข้อมูลตำแหน่งสุดท้ายใน list

Syntax `list.pop(index)`

```
list1 = ['A', 'B', 99, 'd', 'e', 6, 10, 'k', 6]
a=list1.pop()
b=list1.pop(0)
c=list1.pop(-1)
d=list1.pop(3)
print(a, b, c, d)
```

```
['A', 'B', 99, 'd', 'e', 6, 10, 'k']
['B', 99, 'd', 'e', 6, 10, 'k']
['B', 99, 'd', 'e', 6, 10]
['B', 99, 'd', 6, 10]
```

```
6 A k e
```

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

Removing Items: `clear()`

- **`clear()`**: ลบข้อมูลทั้งหมดจาก list

```
list1 = ['A', 'B', 99, 'd', 'e', 6, 10, 'k', 6]
list1.clear()
print(list1)  ──→  [ ]
```

เปลี่ยนแปลงค่าใน List

ข้อมูลใน list สามารถถูกเปลี่ยนแปลงและแก้ไขได้ (mutable)

```
list1 = [1, 2, 3, 4]
list1[2] = 10
list1[-1] = 5
print(list1)
list1[0:2] = ['a','b']
print(list1)
```

```
[1, 2, 10, 5]
```

แก้ไขหลายข้อมูลพร้อมกัน

```
['a', 'b', 10, 5]
```

การหาขนาดของ List

ใช้ Function **`len()`** ในการหาขนาดของ list หรือ จำนวนข้อมูลใน list

```
list1 = [1, 2, 3, 4]
print(len(list1))
```

```
4
```

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

การเรียงลำดับข้อมูลใน List

ใช้ในการเรียงข้อมูลภาย list จากน้อยไปมาก หรือ ในทางกลับกัน

```
list1 = [5, 2, 1.1, 4]
list2 = ['c','a','d','b']
list3 = ['a', 2, 1]
list1.sort()
list2.sort(reverse=True)
print(list1)
print(list2)
list3.sort()
```

Syntax `list.sort()` # เรียงจากน้อยไปมาก
`list.sort(reverse=True)`#เรียงจากมากไปน้อย

```
[1.1, 2, 4, 5]
```

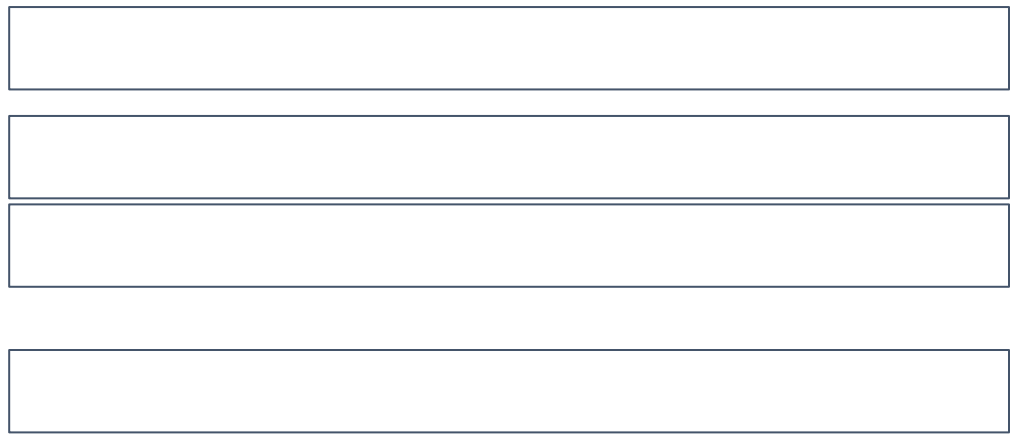
```
['d', 'c', 'b', 'a']
```

```
TypeError: '<' not supported between
instances of 'int' and 'str'
```

Exercise: List

จงหาผลลัพธ์ของชุดคำสั่งต่อไปนี้

```
list1 = [1, 100, 'A', 'x', 5, 'a', 3]
list2 = ['x', 1]
list2.append('y')
print(list2)
list1.extend(list2)
print(list1)
print(list1[5:])
list1.remove('x')
list1.remove('a')
print(list1)
```

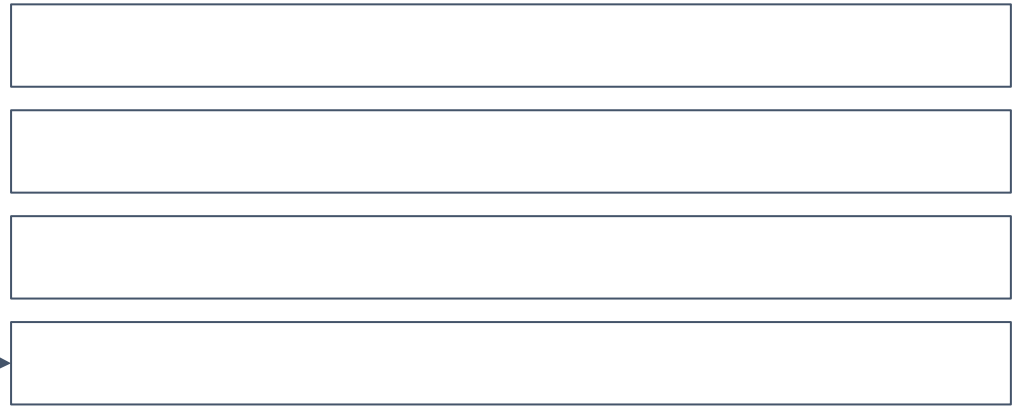


Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

Exercise: List

จงหาผลลัพธ์ของชุดคำสั่งต่อไปนี้

```
list1 = [1, 100, 'A', 'x', 5, 'a', 3]
list1.insert(2, 99)
print(list1)
list1.pop(3)
print(list1)
list1[4] = 11
print(list1)
print(len(list1))
```



Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

Other List Operations

Operation	Description	Example	Output

Concatenation	ใช้ในการเชื่อม list สอง list เข้าด้วยกัน	`x = [1, 2, 3]`
		`y = [5, 4]`
		`print(x+y)`
			`[1, 2, 3, 4, 5]`

Membership	ใช้ในการเช็คว่า item ที่ระบุอยู่ใน list หรือไม่	`x = [1, 2, 3, 4]`
		`print(2 in x)`
		`print (10 in x)`
			`True`
			`False`

Replication	ใช้ในการสร้าง list ที่ประกอบไปด้วยข้อมูลซ้ำๆ	`x = 5*[0]`
		`Y = 3*['a']`
		`print(x)`
		`print(y)`
			`[0,0,0,0,0]`
			`['a', 'a', 'a']`

List Iteration

List Iteration

- สามารถใช้ loop ในการเข้าถึงและแก้ไขข้อมูลใน list ช่วยให้การจัดการ list รวดเร็วและง่ายขึ้น
- สามารถใช้ได้ทั้ง **`for`** and **`while`** statements
- โดยทั่วไปมักจะใช้ **`for`** loop เนื่องจากขนาดของ list นั้นทราบล่วงหน้าอยู่แล้ว

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

List Iteration: `For`

- การวนอ่านค่าภายใน List ด้วยการใช้คำสั่ง **`for`** loop
- ในแต่ละรอบ `x` จะเก็บค่าเป็น item ใน List

```
list1 = [1, 2, 100, 12]
for x in list1:
        print(x, end=' ')
```

ผลลัพธ์

```
1 2 100 12
```

รอบที่	x
1	1
2	2
3	100
4	12

List Iteration: `for`

- การเข้าถึงข้อมูลใน list โดยอาศัย loop และการใช้ index
- ใช้ function **`range()`** และ **`len()`** มาช่วย

```
1
2
100
12
```

```
list1 = [1, 2, 100, 12]
for i in range(len(list1)):
        print(list1[i])
```

Print item ใน list ตามลำดับ

```
12
100
2
1
```

```
list1 = [1, 2, 100, 12]
x = len(list1)-1
for i in range(len(list1)):
        print(list1[x-i])
```

Print item ใน list แบบย้อนลำดับ

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

List Iteration: `while`

- ใช้ **`len()`** ในการช่วยระบุจำนวนรอบในการ loop
- จะต้องกำหนดค่าเริ่มต้นและเปลี่ยนแปลงตัวแปรที่ใช้เป็นเงื่อนไขของ loop เอง

```
list1 = [1, 2, 100, 12]
i = 0
while i<len(list1):
        print(list1[i])
        i+=1
```

```
1
2
100
12
```

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

List Iteration: Example

หาผลลรวมของตัวเลขใน list โดยใช้ `for` และ `while` loop

```
list1 = [3, 5, -1, 9, 2]
total = 0
i = 0
while i< len(list1):
        total = total+list1[i]
        i+=1
print(total)
```

```
list1 = [3, 5, -1, 9, 2]
total = 0
for x in list1:
        total = total +x
print(total)
```

Output: 18

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

List Iteration: Example

for Loop

เขียนโปรแกรมเพื่อหาจำนวนที่มากที่สุดใน list ดังต่อไปนี้

```
list1 = [3, 50, 12, 78, 34, 99, 1, 21]
```

```
list1 = [3, 50, 12, 78, 34, 99, 1, 21]
max_num = list1[0] # ตั้งค่าตัวแปรเป็นตัวแรกของ List
for x in list1:
        # หากมีค่า x ที่สูงกว่า ให้เปลี่ยนค่า max เป็น x
        if x>max_num:
                max_num = x
print(max_num)
```

หลักการคิด:

- ตั้งตัวแปรเพื่อเก็บค่า max
- วนเปรียบเทียบจำนวนในแต่ละคู่กับค่า max
- หากมีค่าใด ๆ มากกว่า ให้เปลี่ยนค่า max เป็นค่านั้น

Output: 99

List Iteration: Example

while Loop

เขียนโปรแกรมเพื่อหาจำนวนที่มากที่สุดใน list ดังต่อไปนี้

```
list1 = [3, 50, 12, 78, 34, 99, 1, 21]
```

```
list1 = [3, 50, 12, 78, 34, 99, 1, 21]
max_num = list1[0] # ตั้งค่าตัวแปรเป็นตัวแรกของ List
i=1
while i<len(list1):
        # หากมีค่าที่สูงกว่า ให้เปลี่ยนค่า max
        if(list1[i]>max_num):
                max_num = list1[i]
        i+=1
print(max_num)
```

Output:

```
99
```

List Iteration: Exercise

เขียนโปรแกรมเพื่อหาจำนวนที่น้อยที่สุดใน list ดังต่อไปนี้

```
list1 = [3, 50, 12, 78, 34, 99, 1, 21]
```

จุดประสงค์ของโปรแกรมนี้คืออะไร และ output คืออะไร

```
grade   = [4, 3, 3, 2, 3]
credit  = [3, 1, 3, 3, 1]
i = 0
total_grade = 0
total_credit = 0
while i<len(grade):
        total_grade = total_grade + (grade[i]*credit[i])
        total_credit = total_credit + credit[i]
        i+=1
print(total_grade/total_credit)
```

Output:

Multi-dimensional List

Multi-Dimensional List

- List ที่มี 2 มิติ หรือมากกว่า
- List สามารถเก็บข้อมูลที่เป็น list ได้
- ตัวอย่างของ multi-dimensional list:
  - `x = [[1, 2, 3],[4, 5, 6],[7, 8, 9]]` `#2D`
  - `y = [[1, 2, 3],[1, 2],[1, 2, 3]]` `#2D`
  - `z = [[[1,2],['a','b']],[[3,4],['c','d']]]` `#3D`

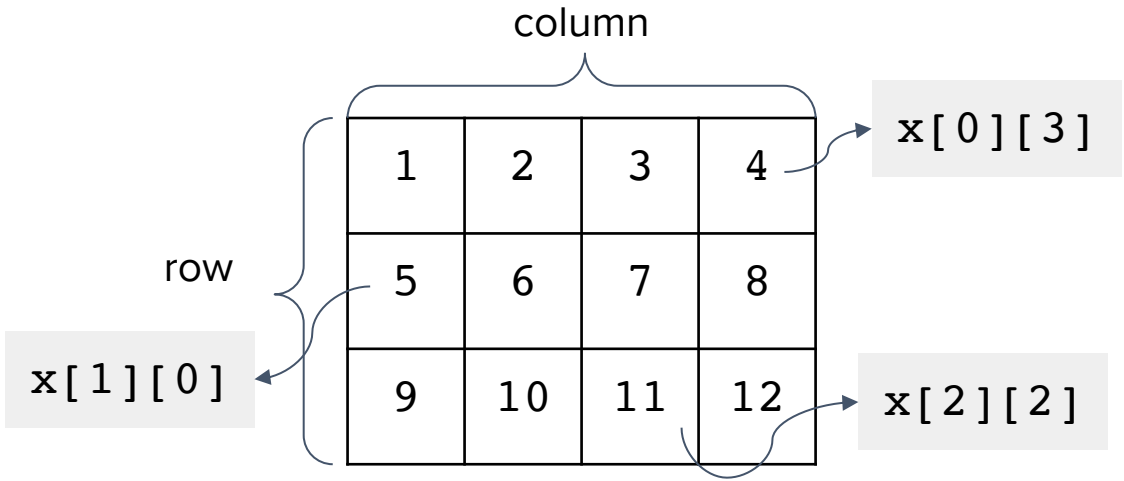
ข้อมูลใน list สามารถมีได้หลายประเภท และจำนวนข้อมูลใน sublist ไม่จำเป็นต้องเท่ากัน

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

2D List Index

- แต่ละข้อมูลใน 2D list สามารถเข้าถึงโดยการระบุตำแหน่ง **row** และ **col**
- เช่น **`x[row][column]`**

```
x = [[1, 2, 3, 4],[5, 6, 7, 8],[9, 10, 11, 12]]
```



Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

Multi-Dimensional List: 2D


```
                [0]             [1]             [2]
          [0]  [1]  [2]   [0]  [1]  [2]   [0]  [1]  [2]      index
x = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]                        #2D
```

column

row	[0]	[1]	[2]
[0]	1	2	3
[1]	4	5	6
[2]	7	8	9

index

[0][0]	[0][1]	[0][2]
[1][0]	[1][1]	[1][2]
[2][0]	[2][1]	[2][2]

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

Exercise

จงหาผลลัพธ์ของชุดคำสั่งต่อไปนี้

```
x = [[1,1,3,10], [5,4,5,8]]
y = [[4,3,2],[6,6,7],[2,1,1],[5,3,9]]
print(x[1][3])
print(y[3][0])
print(x[0][1]+y[1][0])
print(x[1][1]+y[3][2])
```

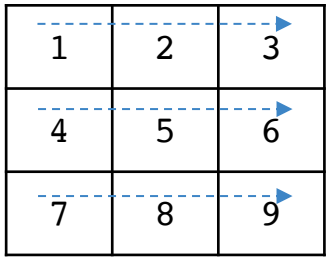
Loop with 2D List: Example

การใช้ loop กับ 2D list -- print ข้อมูลแต่ละตัว

```
x = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
row = len(x)
for i in range(row):
        col = len(x[i])

        for j in range(col):
                print(x[i][j], end=' ')

        print()
```




```
1 2 3
4 5 6
7 8 9
```

Loop with 2D List: Exercise

การ loop 2D list โดยไม่ใช้ index

```
x = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
for r in x:
        for q in r:
                print(q, end=' ')
        print()
```

```
1 2 3
4 5 6
7 8 9
```

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

Loop with 2D List: Exercise

Output ของ code นี้คืออะไร

```
x = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
row = len(x)
for i in range(row):
        col = len(x[i])
        for j in range(col):
                print(x[j][i], end=" ")
        print()
```

1	2	3
4	5	6
7	8	9

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

การสร้าง 2D List

การสร้าง 1D list ประกอบไปด้วย 0s ทั้งหมด n ตัว โดยใช้ *

```
n = 5
y = [0]* n
print(y)
```

output

```
[0, 0, 0, 0, 0]
```

การสร้าง 2D list ขนาด rows x cols ประกอบไปด้วย 0s

```
rows = 3
cols = 5
x = [[0]*cols]*rows
print(x)
```

output

```
[[0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0]]
```

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

การใส่ข้อมูล 2D List ด้วย Loop

การใส่ข้อมูลใน 2D list โดยใช้ **for** loop และ method **append()**

```
x = []
rows = 3
cols = 5
for i in range(rows):
    y = []
    for j in range(cols):
        y.append(0)
    x.append(y)
print(x)
```

ต้องใช้ loop ซ้อน loop

- Outer loop: ระบุจำนวน sub list ภายใน list x
- Inner loop: ใช้ในการเพิ่มข้อมูลใส่ sub list แต่ละอัน

```
[[0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0]]
```

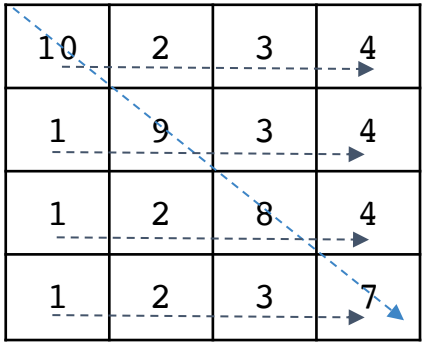
Output ของ code นี้คืออะไร

```
x = []
for i in range(4):
        y = []
        for j in range(4):
                y.append(j+1)
        x.append(y)
print(x)
```

Exercise

หาผลรวมของตัวเลขในแนวทแยง และในแต่ละแถว ของ list x ที่มีขนาด nxn

เช่น `x= [[10, 2, 3, 4], [1, 9, 3, 4], [1, 2, 8, 4], [1, 2, 3, 7]]`




```
[19, 17, 15, 13] 34
```

การเตรียมตัวสำหรับ Mock Exam

- การสอบ Mock Exam เป็นการสอบเตรียมตัวสำหรับ Check Point 2
- สอบปฏิบัติ 3 ข้อใน replit 1 ชั่วโมง 30 นาที
- สอบเวลาประมาณ 14:30 - 16:00 น. (ไม่มี Lab)
- Closed book แต่สามารถนำ Cheated sheet ขนาด A4 หน้า-หลัง เข้าห้องสอบ Mock Exam ได้
- สามารถอ่านคำชี้แจงการสอบได้ใน MyCourses

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

Thanks to: https://keep-calm.net/keep-calm-and-code-on.html

Happy Coding :)