Mahidol University
Wisdom of the Land

Mahidol University

Checkpoint 3

if-else statement

เมื่อถึงคำสั่ง **if** ตัว **condition** จะถูกประมวลผล และแสดงผลลัพธ์ตามค่า Boolean หาก **condition** เป็นจริงจะทำใน Block **if** และ **condition** เป็นเท็จจะทำใน Block **else**

greeting-by-sec2.py

```
section=int(input());
if section==1: #True
     print('Hello, Aj.Pa')
     print('Nice to meet you')
else: #False
     print('Hello, Aj.Tip')
print('Have a good day!!')
```

เมื่อ section มีค่าเป็น 1 ทำให้ section==1 เป็น **True**

```
Hello, Aj.Pa
Nice to meet you
Have a good day!!
```

เมื่อ section มีค่าเป็นอื่นที่ไม่ใช่ 1 ทำให้ section==1 เป็น **False**

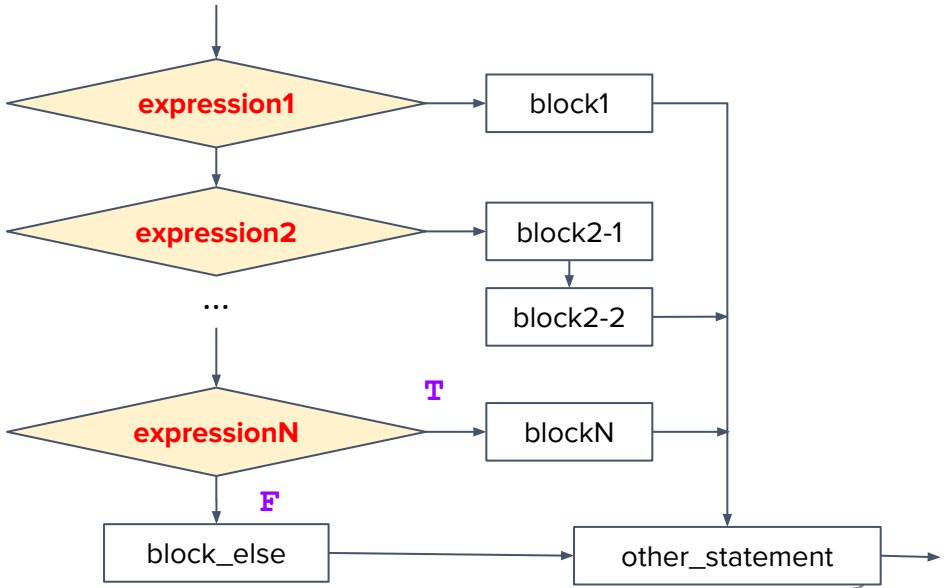
```
Hello, Aj.Tip
Have a good day!!
```

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

elif statement

- **elif** (ย่อจาก else if) เป็นคำสั่งที่ใช้ต่อเนื่องจากคำสั่ง **if** เพื่อควบคุมการทำงานของโปรแกรมให้ทำงานตามหลาย ๆ เงื่อนไขอย่างต่อเนื่อง
- ใช้ร่วมกับ **if** และ **else**

```
if expression1:
     statement_block1
elif expression2:
     statement_block2-1
     statement_block2-2
...
elif expressionN:
     statement_blockN
else:
     statement_block_else
other_statement
```



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

Example: Grading

แสดงผลลัพธ์ของเกรดตามช่วงคะแนน

Score	Grade
>= 80	A
>= 70	B
>=60	C
>=50	D
<50	F

```
score=int(input('Input a score: '))
if (score >= 80):
    print('Grade: A')
elif (score>=70):
    print('Grade: B')
elif (score>=60):
    print('Grade: C')
elif (score>=50):
    print('Grade: D')
else:
    print('Grade: F')
print('Bye bye')
```

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

While Loop: Example

เขียนโปรแกรมรับ input ตัวเลข 10 จำนวน และ print ผลรวมของตัวเลข

Output:

```
Enter a number  :10
10
Enter a number  :5
15
Enter a number  :3
18
Enter a number  :12
30
...
```

```
i = 1
total = 0
while i<=10:
    num = (int)(input("Enter a number  :"))
    total += num
    print(total)
    i+=1
```

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

For Statement

- เป็นคำสั่งวนซ้ำที่ใช้ควบคุมการทำงานซ้ำๆ มักใช้สำหรับการวนอ่านค่าใน iterable object เช่น list, string, tuple
- มักใช้เมื่อทราบจำนวนรอบในการวน loop แน่นอน

```
for var in <sequence>:
    statement(s)
```

ขั้นตอนการทำงาน `for` loop:

1. Item แรกใน sequence ถูก assign เก็บใน **`var`**
2. ชุดคำสั่งใน for loop ถูก execute
3. If item ใน sequence ยังมีอยู่
   a. item ถัดมาใน sequence จะถูก assign ให้ **`var`**
   b. กลับไป step 2

   Else ออกจาก `for` loop

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

`Range()` Function

- **`range()`** เป็น built-in function ที่มักใช้ร่วมกับ for loop
- ใช้ในการสร้าง sequence ของตัวเลข ภายใน range ที่กำหนด
- Default: เริ่มต้นที่ 0, เพิ่มขึ้นทีละ 1, และจบที่ตัวเลขสุดท้ายที่กำหนด

Syntax:

```
range(stop)
range(start, stop)
range(start, stop, step)
```

มี parameter 3 ตัว เป็น type integer ได้แก่

- `start:` ตัวเลขเริ่มต้น
- `stop:` ตัวเลขสุดท้าย (ไม่รวม)
- `step:` ค่าที่เปลี่ยนแปลง

ไม่ support float numbers

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

For/Range(): Example

ไม่รวมตัวสุดท้าย

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

Output:

```
0 1 2 3 4 5 6 7 8 9
```

```
for x in range(5, 12):
    print(x, end=" ")
```

```
5 6 7 8 9 10 11
```

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

Break Example

เขียนโปรแกรมเพื่อรับ input ตัวเลขมากที่สุด 10 จำนวน โดยถ้าตัวเลขที่รับมาเป็นจำนวนลบ ให้ออกจาก loop ทันทีและแสดงผลรวมของ input ที่เป็นจำนวนบวกทั้งหมด

Output:

```
Enter a number :10
Enter a number :5
Enter a number :3
Enter a number :0
Enter a number :-1
18
```

```
total = 0
for x in range(10):
    num = (int)(input("Enter a number :"))
    if num<0:
        break
    total = total + num
print(total)
```

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

Break Example

Revisit: เขียนโปรแกรมเพื่อรับ input ตัวเลข จนกว่าตัวเลขที่รับจะเป็นจำนวนลบ จากนั้นแสดงผลรวมของ input ที่เป็นจำนวนบวกทั้งหมด

Output:

```
Enter a number  :10
Enter a number  :5
Enter a number  :3
Enter a number  :0
Enter a number  :-1
18
```

```
total = 0
while True:
    num = (int)(input("Enter a number  :"))
    if num>=0:
        total += num
    else:
        break
print(total)
```

loop ไปเรื่อยๆ

ออกจาก loop เมื่อตัวเลขติดลบ

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

Continue Example

Print ตัวเลข 1 ถึง 10 ยกเว้น 5

```
1 2 3 4 6 7 8 9 10
```

While Statement

```
x = 0
while x<10:
    x +=1
    if x==5:
        continue
    print(x, end=" ")
```

For Statement

```
for x in range(1, 11):
    if x==5:
        continue
    print(x, end=" ")
```

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

List

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

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

Creating a 1D List

- การประกาศ list นั้นข้อมูลของ list จะอยู่ภายในเครื่องหมาย “[ ]”
- คั่นสมาชิก (item) แต่ละตัวด้วยเครื่องหมาย “ , ”

การสร้าง list เปล่าที่ยังไม่มีข้อมูล

```
# Create an empty list
list1 = []
#or
list1 = list()
```

```
# Create a list of integers
list1 = [1, 2, 3, 4, 5]

# Create a list of characters
list2 = ['a', 'b', 'c', 'd', 'a']

# Create a list with mixed type of item
list3 = [1, 2, 'b', 'c', list1]
```

Item ใน list สามารถซ้ำกันได้

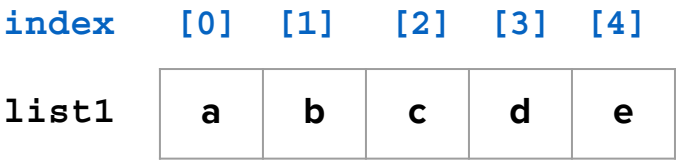
Item ใน list สามารถมีได้หลายประเภท

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

Accessing List Item

- List ใช้ index สำหรับการเข้าถึงข้อมูล แต่ละตัว
- ข้อมูลแรกใน list มี index 0
- ถ้า list มีข้อมูล n item, index ของข้อมูลลำดับสุดท้ายคือ n-1

```
list1 = ['a', 'b', 'c', 'd', 'e']
```




```
list1[0]   # 'a'
list1[1]   # 'b'
list1[4]   # 'e'
```

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

Accessing Range of List Item

- การเข้าถึง range ของข้อมูลใน list สามารถทำได้ โดยการระบุ index เริ่มต้นและ index สุดท้าย (ไม่รวม) ของ range

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

```
list1 = ['a', 'b', 'c', 'd', 'e', 1, 2, 3]
```

```
print(list1[0:2])
```
→
```
['a', 'b']
```

```
print(list1[3:6])
```
→
```
['d', 'e', 1]
```

```
print(list1[5:8])
```
→
```
[1, 2, 3]
```

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

Adding Items to List

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

Contact: [jidapa.kra, dolvara.gun, 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, dolvara.gun, 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]
```

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, dolvara.gun, akara.sup, tipajin.tha]@mahidol.edu

Updating 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]
```

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

Obtaining List Length

- function **`len()`** ใช้ในการหาความยาวของ list นั้นคือจำนวนข้อมูลใน list

```
list1 = [1, 2, 3, 4]

print(len(list1))
```

```
4
```

Contact: [jidapa.kra, dolvara.gun, 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, dolvara.gun, 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']`

Summary of List Methods

Methods	Description
`x.append(item)`	ใส่ข้อมูลหนึ่ง item ที่ท้าย list
`x.insert(index, item)`	ใส่ข้อมูลหนึ่ง item ที่ตำแหน่งที่กำหนด
`x.extend(another_list)`	ใส่ข้อมูลหลาย item พร้อมๆกันที่ท้าย list
`x.remove(item)`	ลบข้อมูลที่ระบุจาก list
`x.pop(index)`	ลบ และ return ข้อมูลตำแหน่งท้ายสุดของ list (default), หรือข้อมูลในตำแหน่งที่ระบุ
`x.clear()`	ลบข้อมูลทั้งหมดจาก list

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

Summary of List Methods

Methods	Description
`x.reverse()`	ใช้สำหรับย้อนกลับข้อมูลภายใน list จากตำแหน่งสุดท้ายไปตำแหน่งแรก
`x.count(item)`	ใช้ในการนับจำนวนของ item ที่ระบุใน list
`x.index(item, start, end)`	ใช้ในการหาตำแหน่ง (index) ของ item
`x.copy()`	ใช้ในการคัดลอก (shallow copy) list ทั้งหมด
`x.sort()`	ใช้ในการเรียงข้อมูลภาย list จากน้อยไปมาก หรือ ในทางกลับกัน

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

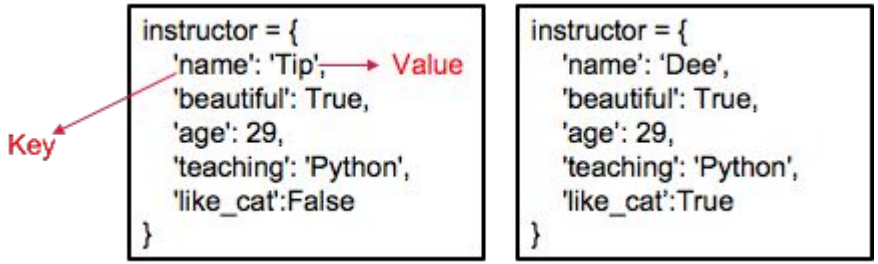
Dictionary

- โครงสร้างข้อมูลชนิดหนึ่ง (built-in data type) ที่ใช้เก็บข้อมูลแบบคู่ของ key-value
- ข้อมูลใน dictionary สามารถเข้าถึงได้โดยใช้ key ซึ่งต่างกับข้อมูลใน list ที่ใช้ index ในการระบุถึงค่าใน list
- dictionary สามารถถูกแก้ไขเปลี่ยนแปลงได้ (mutable)
- ข้อมูลใน dictionary สามารถเปลี่ยนแปลง โดยการเพิ่ม ลด เปลี่ยนค่า จาก dictionary methods

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

Why Dictionary?

Dictionary ที่เก็บข้อมูลอาจารย์:



- โครงสร้างข้อมูลที่ประกอบด้วย key และ value pairs ซึ่งผู้ใช้สามารถตั้งชื่อ key ตามความเหมาะสมได้เอง
- เราสามารถใช้ keys ในการอธิบายความหมายของข้อมูลได้ และ ค่าของข้อมูลนั้นจะถูกเก็บอยู่ใน values
- เราอ้างอิงข้อมูลใน Dictionary โดยใช้ key (ต่างกับ list ที่ใช้ index)

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

Creating a dictionary (1)

- การประกาศ dict นั้นข้อมูลของ dict จะอยู่ภายในเครื่องหมาย “{ }”
- คั่นสมาชิก (item) แต่ละตัวด้วยเครื่องหมาย “ , ”
- คั่น key และ value ด้วย “:”

การสร้าง dict เปล่าที่ยังไม่มีข้อมูล

```
# Create an empty dict
dict1 = {}
#or
dict1 = dict()
```

```
#1. declear + initilize
instructor = {
    'name': 'Tip',
    'beautiful': True,
    'age': 29,
    'teaching': 'Python',
    'like_cat':False
}
```

Item ใน dict สามารถมีได้หลายประเภท (int, float, boolean, string)

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

Creating a dictionary (2)

- การประกาศ dict นั้นข้อมูลของ dict จะอยู่ภายในเครื่องหมาย “**dict()**”
- คั่นสมาชิก (item) แต่ละตัวด้วยเครื่องหมาย “ , ”
- คั่น key และ value ด้วย “=”

การสร้าง dict เปล่าที่ยังไม่มีข้อมูล

```
# Create an empty dict
dict1 = {}
#or
dict1 = dict()
```

Item ใน dict สามารถมีได้หลายประเภท (int, float, boolean, string)

```
#2. declear + initilize
instructor2 = dict(name = 'Dee',
beautiful= True,
age =28,
teaching = 'Python',
like_cat = True)
```

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

Iterating data in Dictionary

- เราสามารถใช้ for...in...loop ในการเข้าถึงข้อมูลใน dictionary ได้ร่วมกับ 3 methods (.values(), .keys(), .items()) ดังนี้

```
instructor = {
    'name': 'Tip',
    'beautiful': True,
    'age': 29,
    'teaching': 'Python',
    'like_cat':False
}
#if we want to print all the values
print(instructor['name'])
print(instructor['beautiful'])
print(instructor['age'])
print(instructor['teaching'])
print(instructor['like_cat'])
```

```
for value in instructor.values():
    print(value)

for key in instructor.keys():
    print(key)

for key, value in instructor.items():
    print(key, value)
```

```
Tip
True
29
Python
False
```

```
name
beautiful
age
teaching
like_cat
```

```
name Tip
beautiful True
age 29
teaching Python
like_cat False
```

Contact: [jidapakra, dolvara.gun, akarasup, tipajin.tha]@mahidol.edu

Dictionary Methods : `fromkeys()`

`fromkeys()` : สร้างข้อมูล dict จาก key-value pairs โดยมี ‘,’ ใช้คั่น

Syntax `dict.fromkeys('key','value')`

```
{}.fromkeys("a","b")
```
→ `{'a':'b'}`

```
dict.fromkeys('a','b')
```
→ `{'a':'b'}`

Dictionary Methods : `fromkeys()`

`fromkeys()` : สร้างข้อมูล dict จาก key-value pairs โดยมี ‘,’ ใช้คั่น

Syntax `dict.fromkeys(['key1','key2'],'value')`

```
{}.fromkeys(['email'],'unknown')
print({}.fromkeys('a',[1,2,3,4,5]))

print({}.fromkeys(['email1','email2'],'unknown'))
print(dict.fromkeys(['email1','email2'],'unknown'))
```

```
{'email':'unknown'}
{'a': [1, 2, 3, 4, 5]}
{'email1': 'unknown', 'email2': 'unknown'}
```

Summary of Dict Methods

Methods	Description
`dict1.clear()`	ลบข้อมูลทั้งหมดจาก dict
`dict1.copy()`	ใช้ในการคัดลอก (shallow copy) dict ทั้งหมด
`dict1.fromkeys([key list], value)`	สร้างข้อมูล dict จาก key-value pairs โดยมี ',' ใช้คั่น
`dict1.get(key)`	การเข้าถึงค่า value จาก key
`dict1.pop(key)` `dict1.popitem()`	เอาข้อมูลออกจาก dict โดยใช้ key ในการค้นหาข้อมูลเพื่อลบออก, ลบข้อมูลจาก dict โดยการ random
`dict1.update(dict2)`	update ค่า keys and values in a dictionary ด้วย another set of key value pairs

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

Tuple

Syntax numbers = (1,2,3,4) #comma separated values

- การประกาศ tuple นั้นข้อมูลจะอยู่ภายในเครื่องหมาย “( )”
- คั่นสมาชิก (item) แต่ละตัวด้วยเครื่องหมาย “ , ”
- ข้อมูลใน tuple ไม่สามารถเปลี่ยนแปลงได้ (immutable) ซึ่งต่างกับ list
- ข้อมูลใน tuple มี index คล้าย list นั่นหมายถึง ข้อมูลใน tuple มีลำดับก่อนหลัง
- ส่วนใหญ่นำไปเก็บข้อมูลที่ไม่มีการเปลี่ยนแปลง เช่น Valid keys in a dictionary

บาง method เช่น .items() ให้ค่า กลับมาเป็น tuple

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

Creating a Tuple

- การประกาศ tuple นั้นข้อมูลจะอยู่ภายในเครื่องหมาย “( )”
- คั่นสมาชิก (item) แต่ละตัวด้วยเครื่องหมาย “ , ”

```
# Create a tuple of integers
x = (1,2,3)
3 in x #true
x[0] = 'change me' #typeError
alphabet = ('a','b','c','d')
alphabet.append('e')
alphabet[0] = 'A'
```

```
TypeError: 'tuple' object does not support item assignment
```

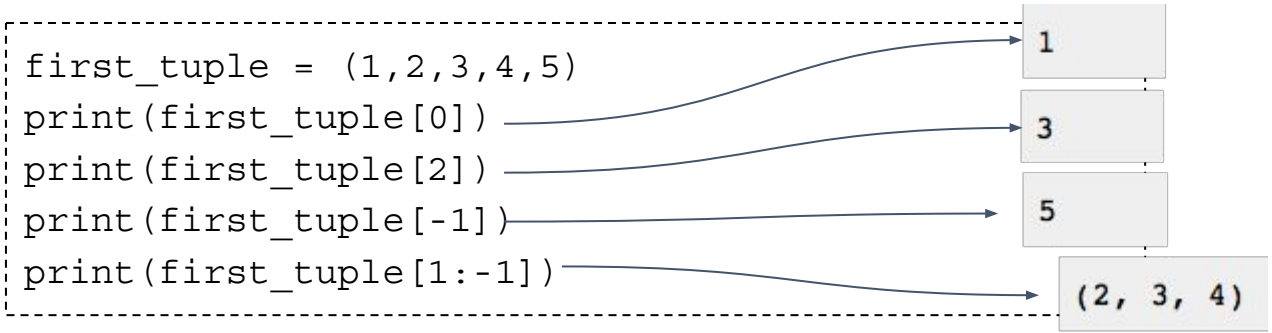
```
AttributeError: 'tuple' object has no attribute 'append'
```

```
TypeError: 'tuple' object does not support item assignment
```

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

Accessing a Tuple

- การเข้าถึงข้อมูลโดยใช้ index คล้าย list



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

Tuple as a key in Dictionary

- Tuples สามารถใช้เป็น keys in dictionaries

```
locations = {
    (35.6895, 39.6917): 'Tokyo Office',
    (40.7128, 74.0060): 'NewYork Office',
    (37.7749, 122.4194): 'San Fran Office'
}
print(locations[(35.6895, 39.6917)])
```

```
Tokyo Office
```

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

Iterate a Tuple (คล้ายกับ List)

- for, while ในการเข้าถึงข้อมูลใน tuple

```
months = ('Jan', 'Feb', 'Mar', 'Apr')
for month in months:
    print(month)

i = len(months)-1
while i>=0:
    print(months[i])
    i-=1
```

```
Jan
Feb
Mar
Apr
```

```
Apr
Mar
Feb
Jan
```

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

Tuple Methods

count(item): นับจำนวนข้อมูล

index(item): returns the index at which a value is found in a tuple

```
x = (1,2,3,3,3)
print(x.count(3))
print(x.index(1))
print(x.index(5))
print(x.index(3))
#only the first matching index is returned
```

3
0
ValueError
2

หา output ของ code ต่อไปนี้

```
#nested tuple, just like a
list
nums = (1,2,3, (4,5),6)
print(nums[3][1])
print(nums[0:])
print(nums[:4])
```

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

Set

Syntax set_data = {1,2,3,4} #comma separated values

- การประกาศ set นั้นข้อมูลจะอยู่ภายในเครื่องหมาย “{ }”
- คั่นสมาชิก (item) แต่ละตัวด้วยเครื่องหมาย “ , ”
- การประกาศ set นั้นคล้ายกันกับ dictionary แต่ข้อมูลใน set ไม่มี key-value pairs.
- ข้อมูลใน set จะไม่ซ้ำกัน (value is unique)
- ข้อมูลใน set สามารถทำ formal mathematical sets (union, intersect) ได้
- ข้อมูลใน set ไม่มีลำดับ (Elements in sets aren't ordered)
- ข้อมูลใน set ไม่สามารถเข้าถึงโดยใช้ index เพราะข้อมูลใน set ไม่มีลำดับ
- ข้อมูลที่เก็บใน set เหมาะกับข้อมูลที่ไม่มีลำดับ ไม่มีข้อมูลซ้ำซ้อน ไม่มี key-value pairs

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

Create a Set

การสร้าง set เปล่าที่ยังไม่มีข้อมูล

```
# Create a set
set1 = {data}
#or
set1 = set({data})
```

```
# Create a set
s = {1,4,5,'a','b',25.5542}
#notice that the order is changed (cannot
use order to access)
print(s)
```

```
{1, 'b', 4, 5, 'a', 25.5542}
```

```
s = set({1,4,5})
#or
s = {1,4,5}
#sets cannot have duplicates
s = set({1,2,3,4,5,5,5}) #{1,2,3,4,5}
print(s)
```

```
{1, 2, 3, 4, 5}
```

```
print(4 in s)
print(8 in s)
```

```
True
False
```

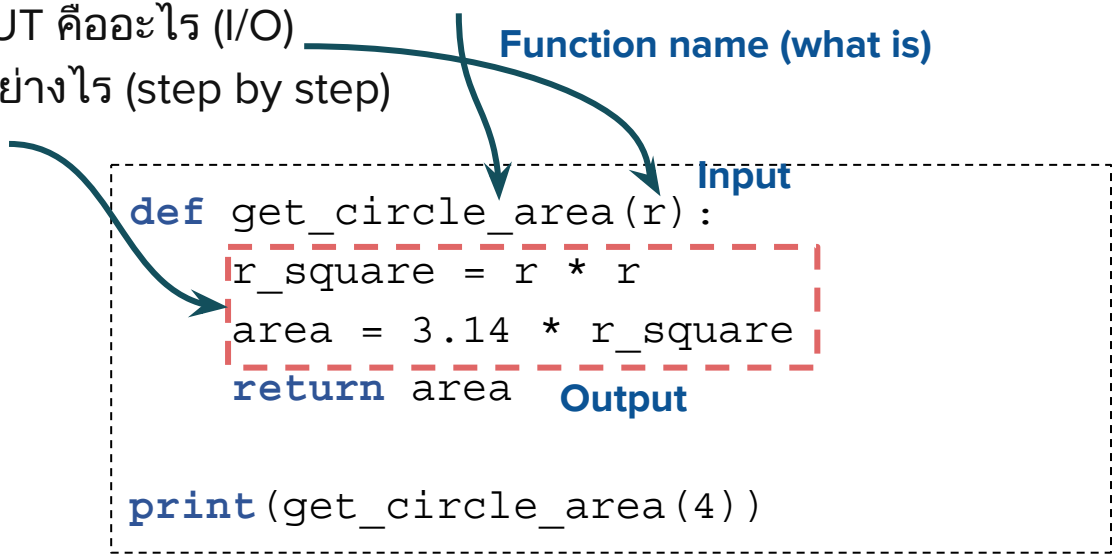
Contact: [jidapa.kra, dolvara.gun, akara.sup, tipajin.tha]@mahidol.edu

Define a function

เมื่อใดก็ตามที่เราใช้ function, เราควร

Syntax `def function_name():`

- รู้ว่า function นี้ถูกสร้างขึ้นมาเพื่อทำงานอะไร (what is)
- รู้ว่า INPUT และ OUTPUT คืออะไร (I/O)
- รู้ว่า function นี้ทำงานอย่างไร (step by step)



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

return keyword

- return ใช้เพื่อคืนค่าจาก function
- เมื่อ return ถูกเรียกใช้ เราจะถือว่าเป็นการจบการทำงานของ function นั้นๆ
- เราสามารถ return หลายค่าได้

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

```
4
```

```
def square_of_7():
    print('before fn')
    return 7**2
     print('after fn')

result= square_of_7()
print(result)
```

```
before fn
49
```

What is the output?


```
from random import random
def flip_coin(): #diff output each time
    # generate random numbers 0-1
    r = random()
    if r>0.5:
        return 'head'
    else:
        return 'tail'

print(flip_coin())
print(flip_coin())
print(flip_coin())
```

```
head
tail
head
```

```
def hello():
    print("hello\n")

hello()
hello()
```

```
hello

hello
```

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

parameters vs argument

1. parameter คือตัวแปรที่ถูกระบุไว้ใน function definition (parameter is variable in the declaration of function)
2. argument คือค่าที่ programmer ใส่เมื่อเรียกใช้ function ซึ่งค่าที่ใส่ไปนี้ จะถูกเก็บโดย parameter ตามลำดับ (argument is the actual value of this variable that gets passed to function)
3. ลำดับในการใส่ค่า parameter และ argument มีความสำคัญ (order is the matter)

default parameters

- default parameter มีไว้เพื่อกำหนดค่าเริ่มต้นใน function
- default parameter จะถูกเรียกใช้เมื่อ ไม่มีการส่งค่าไปที่ function นั้นๆ

default parameter: ลำดับของ default parameter ควรอยู่หลัง parameter ธรรมดา

```
def exponent(num, power=2):
    return num ** power

print(exponent(2,3)) #8
print(exponent(3,2)) #9
print(exponent(3)) #9 --> default value for power
print(exponent())
```

```
TypeError: exponent() missing 1
required positional argument: 'num'
```


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

Keyword Arguments

- เราสามารถระบุ ค่าของ arguments ที่ถูกส่งไปให้ parameters ใน function ได้
- ใน function เราสามารถตั้งค่า default parameter ได้
- เมื่อเรียกใช้ function เราสามารถตั้งค่า keyword argument ได้

```
def exponent(num,  power=2):
    return num ** power

print(exponent(num = 2,power = 3)) #8
print(exponent(power = 3, num = 2)) #8
print(exponent(2, 3)) #8
```

- ตัวแปร (variable) ใน python มีขอบเขตการใช้ภายในและภายนอก function (global/local scope)
- ตัวแปร (variable) ที่ถูกสร้างขึ้นใน functions จะมีขอบเขตอยู่ภายใน function นั้นๆ

```
instructor = 'Mock'    #Global variable
def say_hello():
    return 'Hello '+ instructor
print(say_hello()) ————→ Hello Mock
```

*args

- เราสามารถใช้ *args เป็น function parameter เพื่อทำการรวม arguments ที่ถูกส่งเข้ามาใน function เป็นข้อมูลแบบ **tuple**
- เราสามารถใช้ชื่อตัวแปรอื่นแทน args ได้ เช่น *tmp, *vars แต่โดยทั่วไป programmer จะใช้ชื่อ *args เป็นมาตรฐาน

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

*args

```
def sum_all_nums(*args):
    print(args)  #tuple of all parameters we pass in
    total = 0
    for num in args:
        total += num
    return total

print(sum_all_nums(1,2,3,4,5)) #how about 10 nums
print(sum_all_nums(1,2))
```

```
(1, 2, 3, 4, 5)
(1, 2)
15
3
```

**kwargs

- เราสามารถใช้ **kwargs เป็น function parameter เพื่อทำการรวม arguments ที่ถูกส่งเข้ามาใน function เป็นข้อมูลแบบ **dictionary**
- เราสามารถใช้ชื่อตัวแปรอื่นแทน kwargs ได้ เช่น **tmp, **vars แต่โดยทั่วไป programmer จะใช้ชื่อ **kwargs เป็นมาตรฐาน

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

**kwargs

```
def fav_colors(**kwargs):
    for person, color in kwargs.items():
        print(f'{person} -> {color}')
    print(kwargs)


fav_colors(dee = 'pink', tip='green', mock = 'purple', pa='yellow')
fav_colors('hi') #error
```

```
dee -> pink
tip -> green
mock -> purple
pa -> yellow
```

```
{'dee': 'pink', 'tip': 'green', 'mock':
'purple', 'pa': 'yellow'}
```

```
TypeError: fav_colors() takes 0 positional
arguments but 1 was given
```

Syntax `lambda parameters: expression`

- Lambda เป็น function อีกรูปแบบหนึ่งที่ไม่ต้องตั้งชื่อ (anonymous function)
- Lambda function จะ return ผลลัพธ์โดยอัตโนมัติ
- Lambda function มักถูกใช้ร่วมกับ built-in functions

```
def square(num):
    return num*num
print(square(9)) #81
```

```
square2= lambda num: num * num
print(square2(7)) #49
```

```
add = lambda a,b: a+b
print(add(3,10)) #13
```

```
cube = lambda num: num ** 3
print(cube(2))   #8
print(cube(3))   #27
```


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

Map

Syntax `map(function, iterable)`

- map เป็น built-in function ที่รับ 2 arguments คือ 1) a function และ 2) an iterable เช่น lists, string, dictionaries, sets, tuples
- map ทำงานโดยการเรียก function กับ element ทุกตัวใน an iterable และ return data structure ใหม่

```
nums = [2,4,6,8,10]
def double(x):
    return x*2
doubles = map(double, nums)
print(doubles)
print(list(doubles))
```

```
<map object at 0x118013a58>
```

```
[4, 8, 12, 16, 20]
```

Map

```
nums = [2,4,6,8,10]
doubles = map(lambda x: x*2,nums)
print(list(doubles))
```

```
[4, 8, 12, 16, 20]
```

```
people = ["Darcy", "Kitty", "Mark"]
peeps = map(lambda name: name.upper(), people)
print(list(peeps))
```

```
['DARCY', 'KITTY', 'MARK']
```

Filter

Syntax `filter(function, iterable)`

- filter เป็น built-in function ที่รับ 2 arguments คือ 1) a function และ 2) an iterable เช่น lists, string, dictionaries, sets, tuples
- filter ทำงานโดยการเรียก function กับ element ทุกตัวใน an iterable และ return data ที่ให้ค่า True

```
list1 = [1,2,3,4]
evens =  list(filter(lambda x: x %2== 0, list1))
print(evens)
```

```
[2, 4]
```

```
names = ['austin', 'penny', 'anthony', 'angel', 'billy']
a_names = list(filter(lambda n: n[0]=='a', names))
print(a_names)
```

```
['austin', 'anthony', 'angel']
```

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

สตริง (String)

สตริง (String) คือ ลำดับของตัวอักษรหลายตัวเรียงต่อกัน

การที่จะประกาศ string ค่าของมันจะอยู่ระหว่างเครื่องหมาย double quote (") หรือ single quote (') เท่านั้น

ตัวอย่าง:

- `'hello'`
- `"hello"`
- `'ผมจะตั้งใจเรียนครับ'`
- `'This is a string value'`

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

เราสามารถแสดง string ได้โดยใช้ฟังก์ชั่น print()

```
print('hello')
print("hello")
print('ผมจะตั้งใจเรียนครับ')
print('This is a string value')
```

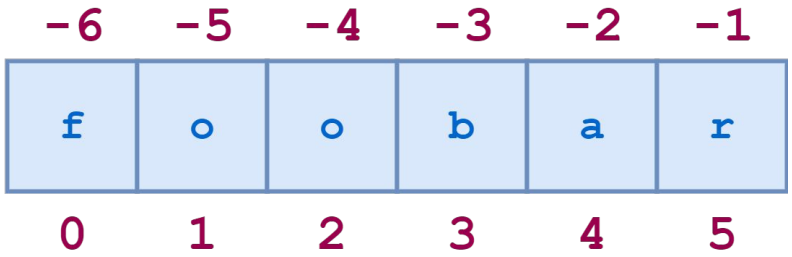
```
hello
hello
ผมจะตั้งใจเรียนครับ
This is a string value
```

สตริง (String)

เนื่องจาก String คือ ลำดับของตัวอักษรหลายตัวเรียงต่อกัน เราจึงสามารถใช้งานได้เหมือนว่าเป็น list ของตัวอักษร

String indexing

```
x = 'foobar'
print(x[0])
print(x[3])
print(x[-1])
```




```
f
b
r
```

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

สตริง (String)

เนื่องจาก String คือ ลำดับของตัวอักษรหลายตัวเรียงต่อกัน เราจึงสามารถใช้งานได้เหมือนว่าเป็น list ของตัวอักษร

```
for x in 'ITDS120':
  print(x)

a = 'อะไรนะ'
for x in a:
  print(x)
```

```
I
T
D
S
1
2
0
```

```
อ
ะ
ไ
ร
น
ะ
```

เราสามารถใช้คำสั่ง `in` ในการตรวจสอบว่ามีลำดับของตัวอักษรอยู่กับ string ได้

```
txt = 'The best things in life are free!'
print('free' in txt)     # True
print('thing' in txt)    # True
print('thins' in txt)    # False

print('h' in 'hello')    # True
print('i' in 'hello')    # False
```

การเข้าถึงบางส่วน string

การเข้าถึงบางส่วน string (slicing string) สามารถทำได้เช่นเดียวกันกับ list

```
txt = 'ITDS120 is fun'
print(txt[1:6])
print(txt[1:4])
print(txt[4:6])
print(txt[:2])
print(txt[5:-1])
print(txt[-5:])
```

```
TDS12
TDS
12
IT
20 is fu
s fun
```

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

การแก้ไข string

เปลี่ยนเป็น ตัวพิมพ์เล็ก (lowercase) (เฉพาะภาษาอังกฤษ)

```
a = "Hello, World! สวัสดีชาวโลก"
print(a.lower())
print("AEiuo".lower())
```

```
hello, world! สวัสดีชาวโลก
aeiuo
```

เปลี่ยนเป็น ตัวพิมพ์ใหญ่ (uppercase) (เฉพาะภาษาอังกฤษ)

```
a = "Hello, World! สวัสดีชาวโลก"
print(a.upper())
print("AEiuo".upper())
```

```
HELLO, WORLD! สวัสดีชาวโลก
AEIUO
```

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

ลบ whitespace

สามารถลบ whitespace เช่น เว้นวรรค ก่อนและหลังตัวอักษรใด ๆ ได้โดยให้ฟังก์ชั่น `strip()`

```
a = "  Hello, World! "
print(a)
print(a.strip())
```

```
  Hello, World!
Hello, World!
```

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

แทนที่ตัวอักษรใน string

สามารถแทนที่ตัวอักษรที่กำหนด (ถ้ามี) ได้ด้วยฟังก์ชั่น `replace(<old>, <new>)`

```
a = "Hello, World!"
print(a.replace("H", "J"))
print(a.replace("Hello", "สวัสดี"))
print(a.replace("a", "ไม่มี"))
```

```
Jello, World!
สวัสดี, World!
Hello, World!
```

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

แยก string ออกเป็นส่วน ๆ

สามารถแยก string ออกเป็นส่วน ๆ ได้โดยใช้คำสั่ง `split(<val>)`

```
a = "Hello, World!"
print(a.split(","))
print(a.split("lo"))
```

```
['Hello', ' World!']
['Hel', ', World!']
```

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

การต่อ string

สามารถใช้เครื่องหมาย + เพื่อรวม string หลายตัวได้เป็น string ตัวเดียวได้ (string concatenation)

```
a = "Hello"
b = "World"
c = a + b
print(c)
d = a + " " + b
print(d)
```

```
HelloWorld
Hello World
```

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

Format string (f-string)

สามารถแทรกค่าของตัวแปรไปใน string ได้ 2 วิธี

วิธีที่ 1 ใช้ `{}` เพื่อระบุตำแหน่งที่จะใส่ค่า และใส่ค่าของตัวแปรผ่านฟังก์ชั่น `format()`

```
age = 36
name = 'John'
txt = "My name is {}, I am {} years old"
print(txt.format(name, age))
```

```
My name is John, I am 36 years old
```

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

Format string (f-string)

สามารถแทรกค่าของตัวแปรไปใน string ได้ 2 วิธี

วิธีที่ 2 เติม f ไว้ด้านหน้า string และใช้ `{var}` ที่ระบุชื่อของตัวแปรที่จะใส่ค่าลงไประหว่าง `{}` ได้เลย

```
age = 36
name = 'John'
txt = f"My name is {name}, I am {age} years old"
print(txt)
```

```
My name is John, I am 36 years old
```