Mahidol University
Wisdom of the Land

Mahidol University

Checkpoint 4

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

Condition

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`**

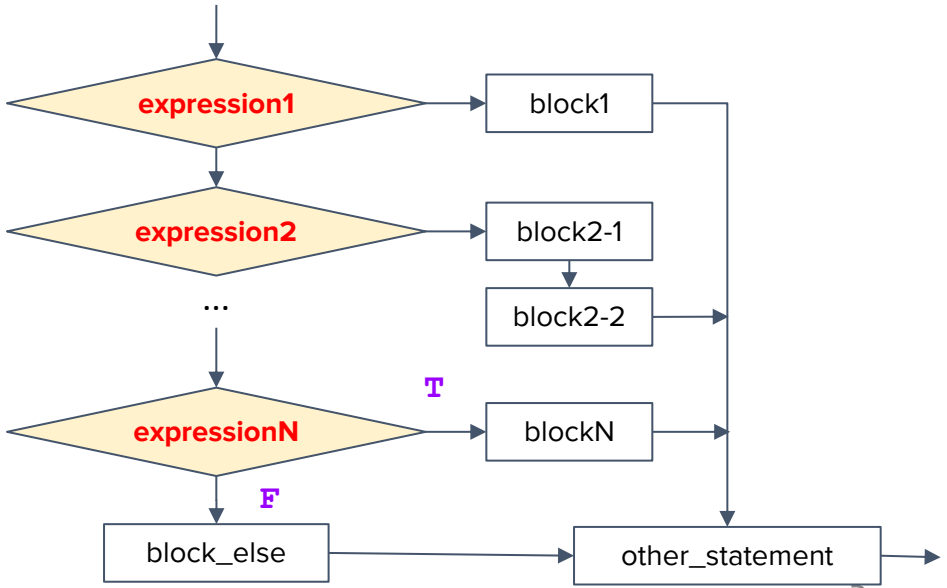
```
Hello, Aj.Tip
Have a good day!!
```

Contact: [jidapa.kra, 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
```



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

Mahidol University
Wisdom of the Land

Mahidol University

Repetitions

`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
```

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, 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**

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, 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)
```

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

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, 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, 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

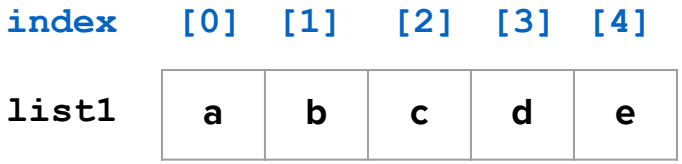
Item ใน list สามารถมีได้หลายประเภท

Contact: [jidapa.kra, 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]      #
```

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, 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, 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]
```

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']`

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]
```

Obtaining List Length

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

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

```
4
```

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

List Iteration: Example

หาผลรวมของตัวเลขใน 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

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']`

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

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

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

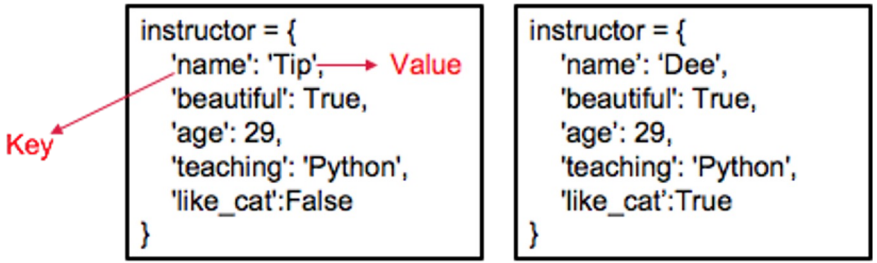
Dict

- 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, 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, 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, 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)
```

Iterating data in Dictionary

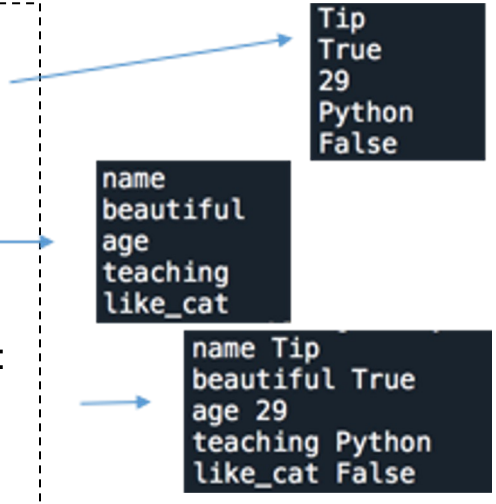
•เราสามารถใช้ for...in...loop ในการเข้าถึงข้อมูลใน dictionary ได้ร่วมกับ 3 methods (.values(), .keys(), .items()) ดังนี้

```
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)
```



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

Tuple

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, 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
```

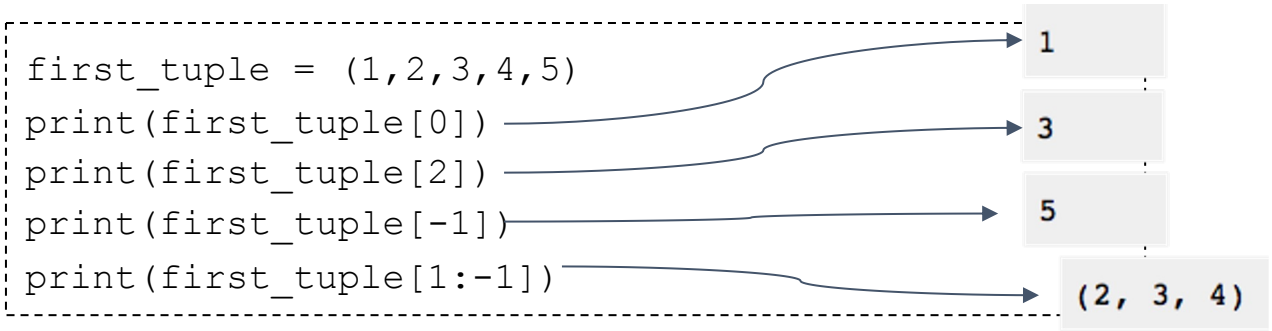
```
AttributeError: 'tuple' object has no attribute 'append'
```

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

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

Accessing a Tuple

- การเข้าถึงข้อมูลโดยใช้ index คล้าย list



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
```

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, 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])
```

Set

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

```
{1, 'b', 4, 5, 'a', 25.5542}
{1, 2, 3, 4, 5}
True
False
```

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

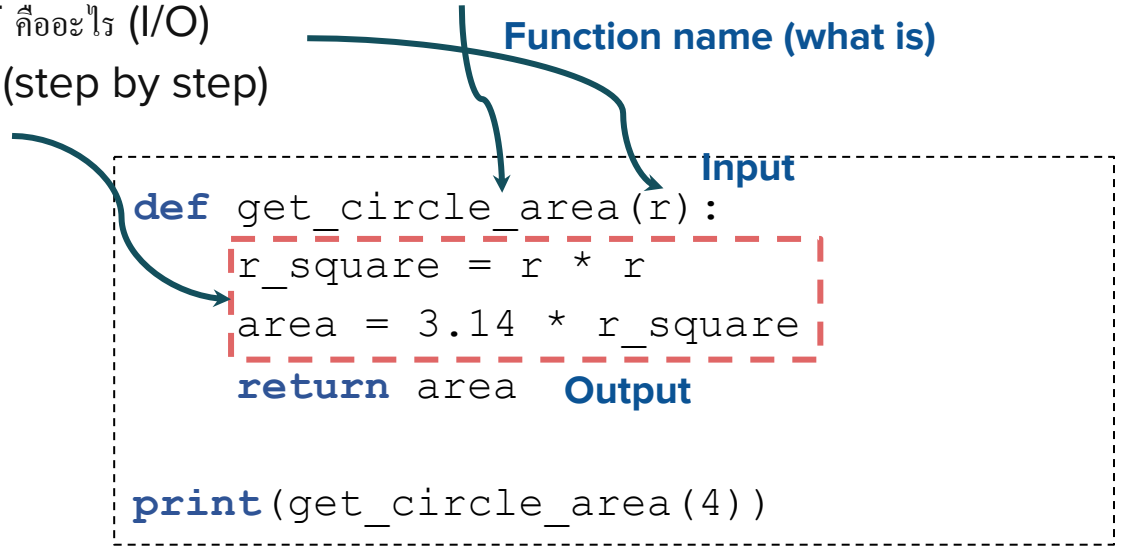
Function

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)



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
```

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)

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

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'
```

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
```

Scope

- 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
```

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

*args

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

*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

**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
```

Lambda

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

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']
```

String

สตริง (String) คือ ลำดับของตัวอักษรหลายตัวเรียงต่อกัน

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

ตัวอย่าง:

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

เราสามารถแสดง string ได้โดยใช้ฟังก์ชั่น print()

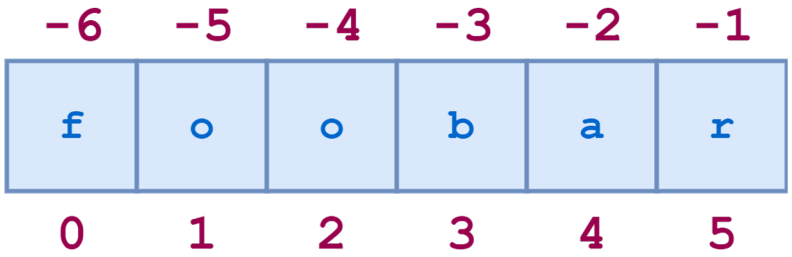
```
print('hello')
print("hello")
print('ผมจะตั้งใจเรียนครับ')
print('This is a string value')
```

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

เนื่องจาก String คือ ลำดับของตัวอักษรหลายตัวเรียงต่อกัน เราจึงสามารถใช้งานได้เหมือนว่าเป็น list ของตัวอักษร

String indexing

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




```
f
b
r
```

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 (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
```

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
```

ลบ whitespace

สามารถลบ whitespace เช่น เว้นวรรค ก่อนและหลังตัวอักษรใด ๆ ได้โดยให้ฟังก์ชั่น strip()

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

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

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!
```

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

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

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

นับจำนวนตัวอักษรใน string

สามารถนับจำนวนตัวอักษรใน string ได้โดยใช้คำสั่ง `count(<val>)`

```
txt = "Hello, World! Hello hello!"
print(txt.count("e"))
print(txt.count("Hello"))
print(txt.count("world!"))
print(txt.count(" "))
```

```
3
2
0
3
```

ฟังก์ชั่นอื่น ๆ ของ string

ใช้ตรวจสอบ string ว่ามีตัวอักษรอะไรอยู่บ้าง

- `s.isalnum()` : return True ถ้าทั้งหมดเป็น A-Z, a-z และ 0-9
- `s.isalpha()` : return True ถ้าทั้งหมดเป็น A-Z และ a-z
- `s.isdecimal()` : return True ถ้าทั้งหมดเป็น 0-9
- `s.islower()` : return True ถ้าทั้งหมดเป็นตัวพิมพ์เล็ก
- `s.isupper()` : return True ถ้าทั้งหมดเป็นตัวพิมพ์ใหญ่

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

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
```

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
```

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
```

File I/O

เปิดไฟล์

การทำงานร่วมกับไฟล์เราต้องเปิดไฟล์ก่อนทุกครั้ง โดนการเปิดไฟล์เราสามารถระบุโหมดของการเปิดได้

- เปิดเพื่ออ่านไฟล์

```
f = open('input.txt', 'r')
```

- เปิดเพื่อเขียนไฟล์

```
f = open('output.txt', 'w')
```

- เปิดเพื่อเขียนต่อไฟล์ (ถ้าไม่มีไฟล์นั้นอยู่จะทำการสร้างใหม่)

```
f = open('output.txt', 'a')
```

หลังจากจัดการกับไฟล์เสร็จแล้วอย่าลืมปิดไฟล์ด้วยทุกครั้ง

```
f.close()
```

input.txt

input.txt

```
1    Hello world1
2    Hello world2
3    Hello world3 Hello world Hello world Hello world Hello worldHello
4    Hello world4
5    Hello world5
6
```

`read()` : อ่านข้อมูลทั้งไฟล์

```
f = open('input.txt', 'r')
print(f.read())
f.close()
```

```
Hello world1
Hello world2
Hello world3 Hello world Hello world Hello world Hello worldHello
worldHello worldHello worldHello worldHello worldHello worldHello
worldHello worldHello worldHello worldHello worldHello worldHello
worldHello worldHello worldHello worldHello worldHello worldHello
worldHello worldHello worldHello worldHello worldHello worldHello
worldHello worldHello world
Hello world4
Hello world5
```

`readlines()` : อ่านข้อมูลทั้งไฟล์โดยแบ่งเป็น list ของบรรทัด

```
f = open('input.txt', 'r')
lines = f.readlines()
for l in lines:
    print(l)
f.close()
```

```
Hello world1

Hello world2

Hello world3 Hello world Hello world Hello world Hello
worldHello worldHello worldHello worldHello worldHello
worldHello worldHello worldHello worldHello worldHello
worldHello worldHello worldHello worldHello worldHello
worldHello worldHello worldHello worldHello worldHello
worldHello worldHello worldHello worldHello worldHello
worldHello worldHello world

Hello world4

Hello world5
```

การเขียนไฟล์ (Write file)

`write()` : เขียน string ลงไปบนไฟล์ (ต้องขึ้นบรรทัดใหม่เองด้วย `'\n'`)

```
f = open('output.txt', 'w')
messages = ['m1', 'm2', 'm3', 'm4']
for m in messages:
    f.write(m)
f.close()
```

output.txt

```
m1m2m3m4
```

โหมดนี้จะเขียนทับไฟล์เดิม

```
f = open('output.txt', 'w')
messages = ['m1', 'm2', 'm3', 'm4']
for m in messages:
    f.write(f'{m}\n')
f.close()
```

output.txt

```
m1
m2
m3
m4
```

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

การเขียนไฟล์ (Write file)

`write()` : เขียน string ลงไปบนไฟล์ (ต้องขึ้นบรรทัดใหม่เองด้วย `'\n'`)

```
f = open('output.txt', 'w')
messages = ['m1', 'm2', 'm3', 'm4']
for m in messages:
    f.write(m)
f.close()
```

output.txt

```
m1m2m3m4
```

โหมดนี้จะเขียนต่อจากบรรทัดสุดท้ายในไฟล์เดิม (จะสร้างใหม่ถ้าไม่เคยมีไฟล์อยู่)

```
f = open('output.txt', 'a')
messages = ['m1', 'm2', 'm3', 'm4']
for m in messages:
    f.write(f'{m}\n')
f.close()
```

output.txt

```
m1m2m3m4m1
m2
m3
m4
```

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

ใช้ with กับการอ่านและเขียนไฟล์


```
# -*- coding: utf-8 -*-
with open('output.txt', 'w') as f:
    f.write('สร้างไฟล์ใหม่\n')

with open('output.txt', 'r') as f:
    print(f.read())

with open('output.txt', 'a') as f:
    f.write('ระวังเวลาใช้โหมด  w  เพราะจะเขียนทับไฟล์ที่มีอยู่แล้ว\n')

with open('output.txt', 'r') as f:
    print(f.read())
```

output.txt

```
สร้างไฟล์ใหม่
ระวังเวลาใช้โหมด  w  เพราะจะเขียนทับไฟล์ที่มีอยู่แล้ว
```

with จะทำการปิดไฟล์ให้เองโดยอัตโนมัติ
นิยมใช้ถ้ากลัวลืมใส่ f.close()

การลบไฟล์ (Delete a file)

ตรวจสอบว่ามีไฟล์อยู่หรือเปล่าก่อนที่จะลบทิ้ง

```
import os
if os.path.exists('output.txt'):
  os.remove('output.txt')
else:
  print('The file does not exist')
```

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

NumPy

การสร้าง numpy array


```
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
print(arr)
print(type(arr))
```

```
[1 2 3 4 5]
<class 'numpy.ndarray'>
```

0-D Array

```
import numpy as np
arr = np.array(42)   # 0-D array
print(arr)
```

```
42
```

การสร้าง numpy array มิติต่างๆ

1-D Array

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5])  # 1-D array
print(type(a))
print(a.shape)
print(a[0], a[1], a[2])  # index
```

```
[1 2 3 4 5]
1 2 3
<class 'numpy.ndarray'>
(5,)
```

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

การสร้าง numpy array มิติต่างๆ

1-D Array

```
import numpy as np
a = np.array([1, 2, 3, 4, 5])  # 1-D array
print(type(a))
print(a.shape)
print(a[0], a[1], a[2])  # index
```

จะให้ค่าจำนวน element ในแต่ละมิติของ array มาในรูปแบบของ tuple

```
<class 'numpy.ndarray'>
(5,)
1 2 3
```

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

2-D Array

```
import numpy as np
arr = np.array([[1,2,3], [4,5,6]])
print(arr)
print(arr[0,0], arr[0,1], arr[1,0], arr[-1,-1])  # index
print(arr.shape)
```

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

3-D Array

```
import numpy as np
arr = np.array(
  [
    [[1, 2, 3], [4, 5, 6]],
    [[7, 8, 9], [10, 11, 12]]
  ])  # 3-D array
print(arr)
print(
  arr[0,0,0], arr[0,1,0],
  arr[0,0,1], arr[0,1,1],
  arr[1,0,0], arr[1,0,1])
print(arr.shape)
```

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

 [[ 7  8  9]
  [10 11 12]]]
1 4 2 5 7 8
(2, 2, 3)
```

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

สร้าง numpy array

สามารถใช้คำสั่ง `np.arange([start,] stop, [step])` ในการสร้าง numpy array ได้

```
import numpy as np
a = np.arange(5)          # ระบุเฉพาะ stop จะเริ่มจาก 0 เสมอ
b = np.arange(5,10)       # ระบุ start และ stop
c = np.arange(5,10,2)     # ระบุ start, stop และ step
print(a)
print(b)
print(c)
```

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

Array Slicing


```
import numpy as np
a = np.array([
  [1,2,3,4],
  [5,6,7,8],
  [9,10,11,12]
])
print(a)
# ดึง 2-D array จาก a ที่แถวที่ 0 และ 1 หลักที่ 1 และ 2
b = a[:2, 1:3]
print(b)

b[0,0] = 77     # b[0,0] จะอ้างอิงถึงข้อมูลเดียวกันกับ a[0, 1]

print(a)
print(b)
```

```
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
[[2 3]
 [6 7]]
[[ 1 77  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
[[77  3]
 [ 6  7]]
```

การใช้ slicing จะไม่ทำให้มิติ (dimension) ของ numpy array ลดลง

```
import numpy as np
# 2D array of shape (3,4)
a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])
print(a)
```

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

Array Slicing

การใช้ slicing จะไม่ทำให้มิติ (dimension) ของ numpy array ลดลง

```
import numpy as np
# 2D array of shape (3,4)
a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])
print(a)

row_r1 = a[1, :]     # มิติลดลงถ้าเราใช้ตัวเลข integer
row_r2 = a[1:2, :]   # มิติเท่าเดิม
print(row_r1, row_r1.shape)
print(row_r2, row_r2.shape)
```

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

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

Array Slicing

การใช้ slicing จะไม่ทำให้มิติ (dimension) ของ numpy array ลดลง

```
import numpy as np
# 2D array of shape (3,4)
a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])
print(a)

col_r1 = a[:,1]      # มิติลดลงถ้าเราใช้ตัวเลข integer
col_r2 = a[:,1:2]    # มิติเท่าเดิม
print(col_r1, col_r1.shape)
print(col_r2, col_r2.shape)
```

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

Integer array indexing

เราสามารถเข้าถึง element โดยใช้ list ของ index ในแต่ละมิติได้

```
import numpy as np
a = np.array([[1,2], [3, 4], [5, 6]])

# ใน 2D-array เราสามารถระบุตำแหน่งของ element
# โดนใช้ list ของ row index และ column index ได้
out = a[[0,1,2], [0,1,0]]
print(out)
print(out.shape)

# ตัวอย่างด้านบนจะให้ค่าเท่ากับ code ด้านล่างนี้
print(a[0,0],a[1,1],a[2,0])
```

```
[1 4 5]
(3,)
1 4 5
```

เราสามารถแก้ไขค่าใน column ต่าง ๆ ได้

```
import numpy as np
a = np.array([[1,2,3], [4,5,6], [7,8,9], [10, 11, 12]])
print(a)

# ระบุ array ของ column index ที่จะแก้ไข
b = np.array([0, 2, 0, 1])

# แสดงผล
print(a[np.arange(4), b])

# เพิ่มค่าให้กับ column เหล่านั้นไป 10
a[np.arange(4), b] += 10
print(a)
```

```
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
[ 1  6  7 11]
[[11  2  3]
 [ 4  5 16]
 [17  8  9]
 [10 21 12]]
```

Boolean array indexing

เราสามารถเข้าถึง element ตามเงื่อนไข (condition) ต่าง ๆ ได้

```
import numpy as np
a = np.array([[1,2], [3, 4], [5, 6]])
bool_idx = (a > 2)   # จะได้ boolean array ที่มีจำนวนมิติเท่ากัน
                     # โดยแต่ละ element เป็น True หรือ False
                     # ที่แสดงสถานะว่า element นั้นเป็นไปตาม
                     # เงื่อนไขที่กำหนดหรือไม่
print(bool_idx)

# เราสามารถเข้าถึง element ต่างโดยใช้ boolean array ได้
# โดยจะดึงมาเฉพาะ element ที่เป็น True เท่านั้น
print(a[bool_idx])
# เราสามารถทำตามด้านบนแบบย่อได้ดังนี้
print(a[a > 2])
```

```
[[False False]
 [ True  True]
 [ True  True]]
[3 4 5 6]
[3 4 5 6]
```

Data Types in NumPy

ประเภทของข้อมูลใน numpy array

- i - integer (`int, np.int32, np.int64`)
- b - boolean
- u - unsigned integer
- f - float (`float, np.float32, np.float64`)
- m - timedelta
- M - datetime
- S - string
- U - unicode string
- ...

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

Data Types in NumPy


```
import numpy as np

arr = np.array([1, 2, 3, 4])
print(arr.dtype)

arr = np.array(['apple', 'banana', 'cherry'])
print(arr.dtype)
```

```
int64
<U6
```

เปลี่ยน dtype ใน numpy

เราสามารถใช้คำสั่ง `astype` ในการเปลี่ยนประเภทของข้อมูลบใน numpy array ได้

```
import numpy as np
arr = np.array([1.1, 2.1, 3.1])
print(arr)
print(arr.dtype)
newarr = arr.astype(int)
print(newarr)
print(newarr.dtype)
```

```
[1.1 2.1 3.1]
float64
[1 2 3]
int64
```

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

เปลี่ยน dtype ใน numpy

เราสามารถใช้คำสั่ง `astype` ในการเปลี่ยนประเภทของข้อมูลใน numpy array ได้

```
import numpy as np
arr = np.array([1.1, 2.1, 3.1])
print(arr)
print(arr.dtype)
newarr = arr.astype(int)
print(newarr)
print(newarr.dtype)
newarr = newarr.astype(float)
print(newarr)
print(newarr.dtype)
```

```
[1.1 2.1 3.1]
float64
[1 2 3]
int64
[1. 2. 3.]
float64
```

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

Math functions in NumPy


```
import numpy as np
arr1 = np.array([10, 11, 12, 13, 14, 15])
arr2 = np.array([20, 21, 22, 23, 24, 25])

out = np.add(arr1, arr2)        # arr1 + arr2 (element-wise)
print(out)
out = np.subtract(arr1, arr2)   # arr1 - arr2 (element-wise)
print(out)
out = np.multiply(arr1, arr2)   # arr1 * arr2 (element-wise)
print(out)
out = np.divide(arr1, arr2)     # arr1 / arr2 (element-wise)
print(out)
```

Math functions in NumPy


```
import numpy as np
arr1 = np.array([10, 11, 12, 13, 14, 15])
arr2 = np.array([20, 21, 22, 23, 24, 25])

out = np.add(arr1, arr2)
print(out)
out = np.subtract(arr1, arr2)
print(out)
out = np.multiply(arr1, arr2)
print(out)
out = np.divide(arr1, arr2)
print(out)
```

```
[30 32 34 36 38 40]
[-10 -10 -10 -10 -10 -10]
[200 231 264 299 336 375]
[0.5        0.52380952 0.54545455 0.56521739
0.58333333 0.6        ]
```

Math functions in NumPy


```
import numpy as np
arr1 = np.array([10, 11, 12, 13, 14, 15])
arr2 = np.array([20, 21, 22, 23, 24, 25])
arr3 = np.array([[1,2,3], [4,5,6]])

out = np.sum(arr1)
print(out)
out = np.sum(arr2)
print(out)
out = np.sum(arr1 - arr2)
print(out)
out = np.sum(arr3)
print(out)
```

```
75
135
-60
21
```

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

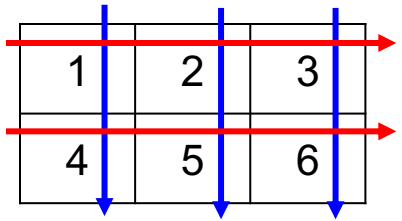
Math functions in NumPy


```
import numpy as np
arr = np.array([[1,2,3], [4,5,6]])

# หาผลรวมตามมิติแรก
out = np.sum(arr, axis=0)
print(out)

# หาผลรวมตามมิติที่สอง
out = np.sum(arr, axis=1)
print(out)
```

```
[5 7 9]
[ 6 15]
```



สามารถทำได้เหมือนกันกับ

`mean()` → **หาค่าเฉลี่ย**

`std()` → **หาค่าเบี่ยงเบนมาตรฐาน**

แสดงค่าใน 2D array ที่มีค่ามากกว่า 10

```
import numpy as np
x = np.array([[0, 10, 20], [20, 30, 40]])
print(x)

# แสดงผล
print(x[x>10])
```

```
[[ 0 10 20]
 [20 30 40]]
[20 20 30 40]
```

นับจำนวนตัวเลขใน array

```
import numpy as np

x = np.array([1,1,1,2,2,2,5,25,1,1])
unique, counts = np.unique(x, return_counts=True)
for v, c in zip(unique, counts):
  print(f'{v}: {c}')
```

```
1: 5
2: 3
5: 1
25: 1
```

Pandas

การสร้าง DataFrame

สร้างผ่าน dict

ชื่อคอลัมน์

```
import pandas as pd

data = {
  "calories": [420, 380, 390],
  "duration": [50, 40, 45]
}

df = pd.DataFrame(data)

print(df)
```

ต้องมีขนาดเท่ากัน !!

```
   calories  duration
0       420        50
1       380        40
2       390        45
```

อ่านค่าจากไฟล์ CSV

อ่านค่าจากไฟล์ CSV แล้วเปลี่ยนให้เป็น numpy array

```
import pandas as pd

df = pd.read_csv('sample.csv')
print(df)

ages = df['age'].values
print(ages)
print(type(ages))
print(ages.dtype)
```

```
   id                                comment    age gender
0   1                           what is this?  24.00      M
1   2                           poor services  27.00      F
2   3  nice food. would love to try again      40.00      F
3   4                                      -  14.35      M
[24.   27.   40.   14.35]
<class 'numpy.ndarray'>
float64
```

อ่านค่าจากไฟล์ CSV

อ่านค่าจากไฟล์ CSV แล้วเปลี่ยนให้เป็น list

```
import pandas as pd

df = pd.read_csv('sample.csv')
print(df)
```

```
   id                                comment    age gender
0   1                           what is this?  24.00      M
1   2                           poor services  27.00      F
2   3  nice food. would love to try again      40.00      F
3   4                                      -   14.35      M
what is this?
poor services
nice food. would love to try again
-
```

```
comments = list(df['comment'].values)
for c in comments:
  print(c)
```

การทำงานกับ DataFrame

แสดงข้อมูลใน DataFram

```
import pandas as pd

df = pd.read_csv('data.csv')
print(df.head())
print(df.tail())
print(df.info())
print(df.describe())
```

```
   Unnamed: 0  PassengerId  Survived  Pclass  ...            Ticket     Fare  Cabin  Embarked
0           0            1         0       3  ...         A/5 21171   7.2500    UNK         S
1           1            2         1       1  ...          PC 17599  71.2833    C85         C
2           2            3         1       3  ...  STON/O2. 3101282   7.9250    UNK         S
3           3            4         1       1  ...            113803  53.1000   C123         S
4           4            5         0       3  ...            373450   8.0500    UNK         S

[5 rows x 13 columns]
     Unnamed: 0  PassengerId  Survived  Pclass  ...      Ticket   Fare  Cabin  Embarked
886         886          887         0       2  ...      211536  13.00    UNK         S
887         887          888         1       1  ...      112053  30.00    B42         S
888         888          889         0       3  ...  W./C. 6607  23.45    UNK         S
889         889          890         1       1  ...      111369  30.00   C148         C
890         890          891         0       3  ...      370376   7.75    UNK         Q

[5 rows x 13 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 13 columns):
Unnamed: 0     891 non-null int64
PassengerId    891 non-null int64
Survived       891 non-null int64
Pclass         891 non-null int64
```

```
       Unnamed: 0  PassengerId    Survived      Pclass         Age       SibSp       Parch        Fare
count  891.000000   891.000000  891.000000  891.000000  891.000000  891.000000  891.000000  891.000000
mean   445.000000   446.000000    0.383838    2.308642   29.699118    0.523008    0.381594   32.204208
std    257.353842   257.353842    0.486592    0.836071   13.002015    1.102743    0.806057   49.693429
min      0.000000     1.000000    0.000000    1.000000    0.420000    0.000000    0.000000    0.000000
25%    222.500000   223.500000    0.000000    2.000000   22.000000    0.000000    0.000000    7.910400
50%    445.000000   446.000000    0.000000    3.000000   29.699118    0.000000    0.000000   14.454200
75%    667.500000   668.500000    1.000000    3.000000   35.000000    1.000000    0.000000   31.000000
max    890.000000   891.000000    1.000000    3.000000   80.000000    8.000000    6.000000  512.329200

None
```

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

ตัวอย่าง: แสดง PassengerId บางส่วน


```
import numpy as np
import pandas as pd

df = pd.read_csv('data.csv')
print(df.head(10)['PassengerId'].values)   # np.ndarray
print(df.iloc[:10]['PassengerId'])         # pd.Series

# NumPy กับ Pandas ถูกออกแบบมาให้ทำงานร่วมกันได้
print(np.sum(df.head(10)['PassengerId']))
print(np.sum(df.iloc[:10]['PassengerId'].values))
# สามารถใช้กับ function อื่นนอกจาก np.sum ได้เหมือนกัน
```

ใช้ระบุ index หรือทำ slicing เสมือนเป็น 2D array ได้ เช่น

```
.iloc[4,5]
.iloc[:,3:6]
```

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

แสดงเฉพาะข้อมูลคอลัมน์ PassengerId กับ Pclass

```
import pandas as pd


df = pd.read_csv('data.csv')
small_df = df.head(10)  # df.iloc[:10]
print(small_df[['PassengerId','Pclass']])
print(small_df[['PassengerId','Pclass']].values)
```

```
   PassengerId  Pclass
0            1       3
1            2       1
2            3       3
3            4       1
4            5       3
5            6       3
6            7       1
7            8       3
8            9       3
9           10       2
[[ 1  3]
 [ 2  1]
 [ 3  3]
 [ 4  1]
 [ 5  3]
 [ 6  3]
 [ 7  1]
 [ 8  3]
 [ 9  3]
 [10  2]]
```

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

แสดงเฉพาะผู้โดยสารที่มี Pclass==1

```
import pandas as pd


df = pd.read_csv('data.csv')
first_class_df = df[df['Pclass']==1]
print(first_class_df)
```

```
     Unnamed: 0  PassengerId  Survived  Pclass  ...       Ticket      Fare        Cabin  Embarked
1             1            2         1       1  ...    PC 17599   71.2833          C85         C
3             3            4         1       1  ...       113803  53.1000         C123         S
6             6            7         0       1  ...       17463   51.8625          E46         S
11           11           12         1       1  ...      113783   26.5500         C103         S
23           23           24         1       1  ...      113788   35.5000           A6         S
27           27           28         0       1  ...       19950  263.0000  C23 C25 C27         S
30           30           31         0       1  ...    PC 17601   27.7208          UNK         C
...
```

หาค่าเฉลี่ยอายุของผู้โดยสารผู้ชายและผู้หญิง

```
import pandas as pd
import numpy as np

df = pd.read_csv('data.csv')
male_passenger_df = df[df['Sex']=='male']
print(male_passenger_df['Age'].mean())
female_passenger_df = df[df['Sex']=='female']
print(female_passenger_df['Age'].mean())
```

```
30.50582424304206
28.216730048707397
```

เลือกข้อมูลผู้โดยสารที่มีชื่อบางส่วนเป็น James

```
import pandas as pd
import numpy as np

df = pd.read_csv('data.csv')
james_df = df[df['Name'].str.contains('James')]
print(james_df[['PassengerId', 'Name', 'Pclass', 'Embarked']])
```

เลือกข้อมูลผู้โดยสารที่มีชื่อบางส่วนเป็น James

มีชื่อ James เป็นส่วนหนึ่งก็ได้

```
import pandas as pd
import numpy as np

df = pd.read_csv('data.csv')
james_df = df[df['Name'].str.contai
print(james_df[['PassengerId', 'Nam
```

	PassengerId	Name	Pclass	Embarked
5	6	Moran, Mr. James	3	Q
67	68	Crease, Mr. Ernest James	3	S
134	135	Sobey, Mr. Samuel James Hayden	2	S
150	151	Bateman, Rev. Robert James	2	S
161	162	Watt, Mrs. James (Elizabeth "Bessie" Inglis Mi...	2	S
174	175	Smith, Mr. James Clinch	1	C
194	195	Brown, Mrs. James Joseph (Margaret Tobin)	1	C
221	222	Bracken, Mr. James H	2	S
250	251	Reed, Mr. James George	3	S
299	300	Baxter, Mrs. James (Helene DeLaudeniere Chaput)	1	C
416	417	Drew, Mrs. James Vivian (Lulu Thorne Christian)	2	S
428	429	Flynn, Mr. James	3	Q
468	469	Scanlan, Mr. James	3	Q
511	512	Webber, Mr. James	3	S
512	513	McGough, Mr. James Robert	1	S
525	526	Farrell, Mr. James	3	Q
551	552	Sharp, Mr. Percival James R	2	S
582	583	Downton, Mr. William James	2	S
592	593	Elsbury, Mr. William James	3	S
696	697	Kelly, Mr. James	3	S
740	741	Hawksford, Mr. Walter James	1	S
811	812	Lester, Mr. James	3	S
812	813	Slemen, Mr. Richard James	2	S
883	884	Banfield, Mr. Frederick James	2	S

เลือกข้อมูลผู้โดยสารที่ขึ้นเรือจากท่า 'Q' หรือ 'S'

```
import pandas as pd
import numpy as np

df = pd.read_csv('data.csv')
embark_df = df[df['Embarked'].isin(['Q','S'])]
print(embark_df[['PassengerId', 'Name', 'Pclass', 'Embarked']])
```

ต้องเป็น list หรือ numpy array ของค่า

หาค่าเฉลี่ยค่าโดยสาร (Fare) ของผู้โดยสารที่มีชื่อบางส่วนเป็น William

```
import pandas as pd


df = pd.read_csv('data.csv')
william_df = df[df['Name'].str.contains('William')]
print(william_df['Fare'].mean())
```

```
                                                 Name      Fare
4                            Allen, Mr. William Henry    8.0500
12                         Saundercock, Mr. William Henry    8.0500
17                           Williams, Mr. Charles Eugene   13.0000
23                           Sloper, Mr. William Thompson   35.5000
31   Spencer, Mrs. William Augustus (Marie Eugenie)  146.5208
..                                                 ...       ...
802             Carter, Master. William Thornton II  120.0000
810                            Alexander, Mr. William    7.8875
864                            Gill, Mr. John William   13.0000
880    Shelley, Mrs. William (Imanita Parrish Hall)   26.0000
885              Rice, Mrs. William (Margaret Norton)   29.1250

[69 rows x 2 columns]
33.13435797101449
```

นับจำนวนของค่าในคอลัมน์ : value_counts()

```
import pandas as pd

df = pd.read_csv('data.csv')
print(df['Embarked'].value_counts())            # pd.Series
print(df['Embarked'].value_counts().to_dict())  # dict
```

```
S      644
C      168
Q       77
UNK      2
Name: Embarked, dtype: int64
{'S': 644, 'C': 168, 'Q': 77, 'UNK': 2}
```

หาค่าที่ไม่ซ้ำกันในแต่ละคอลัมน์ : unique()

```
import pandas as pd

df = pd.read_csv('data.csv')
print(df['Pclass'].unique())     # numpy array
print(df['Embarked'].unique())   # numpy array

# ระวัง: ไม่ควรใช้กับตัวเลขจำนวนจริง (ที่จะมีค่าแตกต่างกันเยอะมาก) เช่น ค่าโดยสาร
# print(df['Fare'].unique())
```

```
[3 1 2]
['S' 'C' 'Q' 'UNK']
```

เรียงลำดับโดยดูจากค่าในคอลัมน์ต่าง ๆ : sort_values()

```
import pandas as pd

df = pd.read_csv('data.csv')
print(df.sort_values(by=['Pclass']))                    # ค่าตั้งต้นมาตรฐาน ascending=True (เรียงจากน้อยไปมาก)
print(df.sort_values(by=['Pclass'], ascending=False))   # เรียงจากมากไปน้อย

# เรียงด้วย Pclass จากมากไปน้อยตามด้วย Fare จากมากไปน้อย
print(df.sort_values(by=['Pclass', 'Fare'], ascending=False)[['Pclass', 'Fare']])

# เรียงด้วย Pclass จากน้อยไปมากตามด้วย Fare จากมากไปน้อย
print(df.sort_values(by=['Pclass', 'Fare'], ascending=[True, False])[['Pclass', 'Fare']])
```

การทำงานกับ DataFrame

เราสามารถรวมเงื่อนไขต่าง ๆ ได้โดยใช้

- `&` : and
- `|` : or
- `~` : not

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

แสดงผู้โดยสารผู้ชายที่ซื้อตั๋วโดยสารถูกกว่า 15.25

```
import pandas as pd

df = pd.read_csv('data.csv')
print(df[(df['Sex']=='male') & (df['Fare']<15.25)])
```

ต้องมีวงเล็บ () ครอบเงื่อนไขต่าง ๆ เวลาที่ถูกเชื่อมด้วย &, | หรือ ~ ด้วย

ผู้โดยสารผู้ชาย

ค่าตั๋วถูกกว่า 15.25

	Unnamed: 0	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	0	1	0	3	Braund, Mr. Owen Harris	male	22.000000	1	0	A/5 2117	7.250	UNK	S
4	4	5	0	3	Allen, Mr. William Henry	male	35.000000	0	0	37345	8.050	UNK	S
5	5	6	0	3	Moran, Mr. James	male	29.699118	0	0	33087	8.458	UNK	Q
12	12	13	0	3	Saundercock, Mr. William Henry	male	20.000000	0	0	A/5. 215	8.050	UNK	S
17	17	18	1	2	Williams, Mr. Charles Eugene	male	29.699118	0	0	24437	13.000	UNK	S
..	...	...	...	...	...	...	...	...	...	..	..	...	...
881	881	882	0	3	Markun, Mr. Johann	male	33.000000	0	0	34925	7.895	UNK	S
883	883	884	0	2	Banfield, Mr. Frederick James	male	28.000000	0	0	C.A./SOTON 3406	10.500	UNK	S
884	884	885	0	3	Sutehall, Mr. Henry Jr	male	25.000000	0	0	SOTON/OQ 39207	7.050	UNK	S
886	886	887	0	2	Montvila, Rev. Juozas	male	27.000000	0	0	21153	13.000	UNK	S
890	890	891	0	3	Dooley, Mr. Patrick	male	32.000000	0	0	37037	7.750	UNK	Q

[349 rows x 13 columns]

แสดงผู้โดยสารผู้ชายที่ซื้อตั๋วโดยสารถูกกว่า 15.25

```
import pandas as pd

df = pd.read_csv('data.csv')
sel_df = df[(df['Sex']=='male') & (df['Fare']<15.25)]

# เรียงตามราคาตั๋ว
sel_df = sel_df.sort_values(by=['Fare'], ascending=False)

# แสดงผลเฉพาะคอลัมน์ 'Name','Sex','Fare'
print(sel_df[['Name','Sex','Fare']])
```

```
                                                  Name   Sex     Fare
65                              Moubarek, Master. Gerios  male  15.2458
709  Moubarek, Master. Halim Gonios ("William George")  male  15.2458
497                 Shellard, Mr. Frederick William  male  15.1000
181                              Pernot, Mr. Rene  male  15.0500
135                            Richard, Mr. Emile  male  15.0458
..                                            ...   ...     ...
263                         Harrison, Mr. William  male   0.0000
271                  Tornquist, Mr. William Henry  male   0.0000
277                     Parkes, Mr. Francis "Frank"  male   0.0000
302                  Johnson, Mr. William Cahoone Jr  male   0.0000
732                            Knight, Mr. Robert J  male   0.0000

[349 rows x 3 columns]
```