



Mahidol University  
*Wisdom of the Land*



# Lecture 12: Function II

Lecturers:

Aj. Jidapa Kraisangka

Aj. Akara Supratak

Aj. Tipajin Thaipisutikul



Mahidol University  
*Wisdom of the Land*



# Recap

สัปดาห์หน้าเราจะมี Checkpoint 3 (Coding + Multiple Choice)

ไม่มีแล็ป สอบเสร็จกลับบ้านได้เลย



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 parameter มีไว้เพื่อกำหนดค่าเริ่มต้นใน function
- default parameter จะถูกเรียกใช้เมื่อ ไม่มีการส่งค่าไปที่ function

`def exponent(num, power=2):` default parameter: ลำดับของ default parameter ควรอยู่หลัง parameter ธรรมดา  
 `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'



- Parameters และ default parameters สามารถเป็นข้อมูลชนิดใดก็ได้

```
def add(a,b):  
    return a+b  
  
def math(a,b, fn=add): #default should go the last, order does matter  
    return fn(a,b)  
  
def subtract(a,b):  
    return a-b  
  
print(math(4,5))  
print(math(4,5, subtract))
```

→ 9  
-1



- 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
```



1. สามารถใช้ \* และ \*\* operator as parameters to a function
2. เข้าใจว่า lambda คืออะไรและใช้งานได้อย่างไร
3. สามารถใช้ built-in functions เช่น min, max, sort, reverse ได้



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





## Before \*args

```
def sum_all_nums(num1, num2, num3):  
    return num1+num2+num3  
  
print(sum_all_nums(1,2,3)) #6
```

```
def sum_all_nums(num1, num2, num3=0, num4=0, num5=0):  
    return num1+num2+num3+num4+num5  
  
print(sum_all_nums(1,2,3)) #6
```



After \*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
```

(1, 2, 3, 4, 5)

(1, 2)

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

15

3



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



```
def fav_colors(**kwargs):
```

```
    for person, color in kwargs.items():
```

```
        print(f'{person} -> {color}')
```

```
    print(kwargs)
```

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

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

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

```
fav_colors('hi') #error
```

TypeError: fav\_colors() takes 0 positional  
arguments but 1 was given



```
def special_greeting(**kwargs):  
    if 'Dee' in kwargs and kwargs['Dee'] == 'special':  
        return 'you get a special greeting'  
    elif 'Dee' in kwargs:  
        return 'Hello'  
    return 'Not sure who this is...'
```

print(special\_greeting(Dee='Hi'))

print(special\_greeting(Dee='special'))

print(special\_greeting(John='Yo'))

Hello

you get a special greeting

Not sure who this is...



ถ้า function มีการรับ parameter ทั้ง 4 แบบ ลำดับของ parameters ควรมีการจัดวางดังนี้

1) parameters 2) \*args 3) default parameters 4) \*\*kwargs



1) parameters 2) \*args 3) default parameters 4) \*\*kwargs

```
def display_info(a,b,*args, instructor='Mock', **kwargs):
```

```
    print(a)
```

```
    print(b)
```

```
    print(args)
```

```
    print(instructor)
```

```
    print(kwargs)
```

```
    return [a,b,args, instructor, kwargs]    #return a list
```

1

2

(3, 4)

Mock

{'last\_name': 'Yo', 'job': 'instructor'}

```
print(display_info(1,2,3,4,last_name='Yo', job='instructor'))
```

```
[1, 2, (3, 4), 'Mock', {'last_name': 'Yo', 'job': 'instructor'}]
```



1) parameters 2) \*args 3) default parameters 4) \*\*kwargs

```
def display_info(a,b,*args, instructor='Mock', **kwargs):  
    print(a)  
    print(b)  
    print(args)  
    print(instructor)  
    print(kwargs)  
    return [a,b,args, instructor, kwargs]    #return a list
```

What is the output?

```
print(display_info(1,2,3,4,instructor='Dee',last_name='Yo', job='instructor'))  
print(display_info(1,2,3,4,'Dee',last_name='Yo', job='instructor'))
```





**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
```



## 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]



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

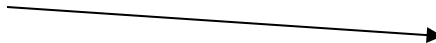
[4, 8, 12, 16, 20]

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

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



```
names = [  
    {"first": "Darcy", "last": "Rap"},  
    {"first": "Kitty", "last": "Wang"},  
    {"first": "Mark", "last": "Ton"}  
]  
  
first_name = list(map(lambda x: x['first'], names))  
print(first_name)
```



```
['Darcy', 'Kitty', 'Mark']
```



## 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
- Map 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']



```
users = [  
    {"username": "samuel", "tweets": ["I love cake", "I love pie", "hello world!"]},  
    {"username": "katie", "tweets": ["I love my cat"]},  
    {"username": "jeff", "tweets": []},  
    {"username": "bob123", "tweets": []},  
    {"username": "doggo_luvr", "tweets": ["dogs are the best", "I'm hungry"]},  
    {"username": "guitar_gal", "tweets": []}  
]  
  
#extract inactive users using filter:  
inactive_users = list(filter(lambda u: len(u['tweets'])==0, users))  
print(inactive_users) → [{'username': 'jeff', 'tweets': []},  
                          {'username': 'bob123', 'tweets': []},  
                          {'username': 'guitar_gal', 'tweets': []}]  
  
# extract usernames of inactive users with map and filter:  
usernames = list(map(lambda user: user["username"].upper(),  
                     filter(lambda u: len(u['tweets'])==0, users)))  
print(usernames) → ['JEFF', 'BOB123', 'GUITAR_GAL']
```



**Syntax** `sorted(iterable)`  
`sorted(iterable, reverse=True)`

- `sorted` เป็น built-in function ที่รับ an iterable เช่น lists, string, dictionaries, sets, tuples และ return data ที่เรียงจากน้อยไปมาก
- ถ้าต้องการเรียงลำดับข้อมูลจากมากไปน้อย ให้ระบุ `reverse=True`

```
nums = [4,6,1,30,55,23]
```

```
new_list = sorted(nums)
```

```
print(new_list)
```

[1, 4, 6, 23, 30, 55]

```
new_list = sorted(nums, reverse=True)
```

```
print(new_list)
```

[55, 30, 23, 6, 4, 1]



**Syntax** `max(iterable)`

**Syntax** `min(iterable)`

- Max, min เป็น built-in function ที่รับ an iterable เช่น lists, string, dictionaries, sets, tuples หรือตัวเลขหลายตัว หรือตัวอักษรหลายตัว และ return data ที่มีค่ามากที่สุดหรือน้อยที่สุด

```
nums = [40, 32, 6, 5, 10]
print(max(3, 67, 99))           #99
print(max('c', 'd', 'a'))       #d
print(max(nums))                 #40
print(min(3, 67, 99))           #3
print(min('c', 'd', 'a'))       #a
print(min(nums))                 #5
```





Methods	Description
<code>reversed(item)</code>	เรียงข้อมูลโดยเอาตัวท้ายขึ้นก่อน
<code>abs(item)</code>	คืนค่าจำนวนเต็มบวกของข้อมูลนั้น
<code>sum(item)</code>	คืนค่าผลรวมของข้อมูลนั้น
<code>round(item, decimal)</code>	ใช้เพื่อระบุจำนวนจุดทศนิยมที่ต้องการ ถ้าจำนวนจุดทศนิยมไม่ได้ถูกระบุ ตัวเลขจะถูกปัดให้เป็นจำนวนเต็มที่ใกล้เคียงที่สุด



**Syntax** `reversed(iterable)`

**Syntax** `sum(iterable)`

**Syntax** `abs(number)`

**Syntax** `round(number)`

```
nums = [40,32,6,5,10]
print(list(reversed(nums)))
print(list(reversed('hello')))
```

[10, 5, 6, 32, 40]

['o', 'l', 'l', 'e', 'h']

```
print(abs(-5))
print(abs(-3.444))
```

5

3.444



```
print(sum([1,2,3]))  
print(sum([1,2,3], 10))  
print(sum([1,2,3], -6))  
print(sum((1.5,1.5,3.0)))
```

6  
16  
0  
6.0

```
print(round(10.2))  
print(round(3.51234, 3))
```

10  
3.512