函數補充

Fuction

enumerate()

- enumerate()函數通常使用在for迴圈中，將傳入的序列型與一組連續整數以tuple方式配對，語法為：

```
enumerate(iterables, [start=0])
```

  - iterables：可以是tuple、list、字串等
  - start：預設為0

```python
names = ["小明", "小張", "小白"]

for i, name in enumerate(names):
    print("No.%d : %s" %(i+1, name))

"""
No.1 : 小明
No.2 : 小張
No.3 : 小白
"""
```

```python
names = ["小明", "小張", "小白"]

for i, name in enumerate(names, start=1):
    print("No.%d : %s" %(i, name))

"""
No.1 : 小明
No.2 : 小張
No.3 : 小白
"""
```

zip()

- zip()函數是將多個序列物件配對成一群tuple，並打包成一個可迭代的zip物件，語法為：

  zip(iterables)

  - iterables：可以是tuple、list、字串等

```python
country = ["Taiwan", "Japan", "U.S.A.", "U.K.", "Australia"]
country_code = (886, 81, 1 , 44, 61)

# 方法一：先用zip打包後再用for迴圈輸出結果
code = zip(country, country_code)
for i in code:
    print(i)

# 方法二：用zip打包後存入dict中
info = {}
for name, i in zip(country, country_code):
    info[name] = i

print(info)
```

```
('Taiwan', 886)
('Japan', 81)
('U.S.A.', 1)
('U.K.', 44)
('Australia', 61)
{'Taiwan': 886, 'Japan': 81, 'U.S.A.': 1, 'U.K.': 44, 'Australia': 61}
```

map()

- map()函數是將傳入的函式套用到可迭代的元素上，並將運算結果回傳，語法為：

  map(function, iterable,....)

  - fuction：fuction函式中的參數個數要與可迭代的個數相同；若傳入多個可迭代物件，但迭代物件長度不相同時，運算時則以長度最短迭代物件為主

```python
list1 = [1,2,3,4,5]

# 方法一：map+list
def cal(x):
    return x**2

result1 = list(map(cal, list1))
print(result1)
# [1, 4, 9, 16, 25]

# 方法二：map+lambda
result2 = list(map(lambda x:x**3, list1))
print(result2)
# [1, 8, 27, 64, 125]
```

```python
list1 = [1,2,3,4,5]
list2 = [10,20,30]

def cal(x, y):
    return x+y

result = list(map(cal, list1, list2))
print(result)

# [11, 22, 33]
```

```python
list1 = ["100", "200", "300", "400"]

list2 = list(map(int, list1))
print(list2)
# [100, 200, 300, 400]
```

filter()

- filter()函數是使用function對可迭代物件內的元素進行過濾，只回傳為True的元素，語法為：

```
filter(function, iterable)
```

```python
list1 = [0, 2.5, False, 30]

def cal(x):
    if x > 1:
        return x

ans1 = list(filter(cal, list1))
print(ans1)
# [2.5, 30]


ans2 = list(filter(lambda x:x>1, list1))
print(ans2)
# [2.5, 30]
```

```python
ans1 = list(filter(None, [0,1,2,3,4,5]))
print(ans1)     # [1, 2, 3, 4, 5]

list1 = [i for i in range(6)]
def cal(x):
    if x%2==0:
        return x
ans2 = list(filter(cal, list1))
print(ans2)     # [2, 4]

ans3 = list(filter(lambda x:x%2==0, list1))
print(ans3)     # [0, 2, 4]
```

lambda()

- lambda()函數為匿名函式，不需要定義名稱，只有一行運算式，語法為：

```
lambda arguments_list : expression
```

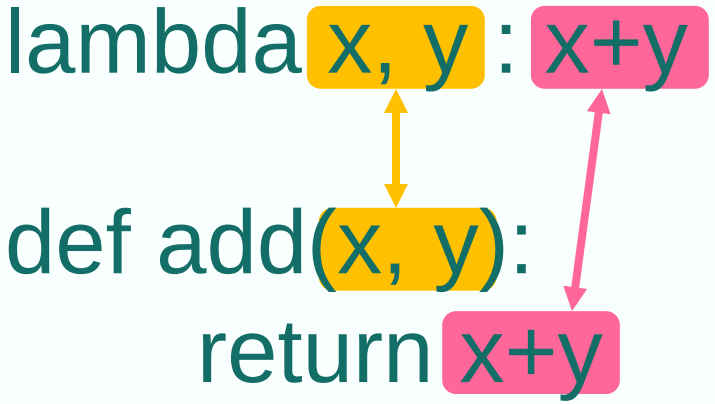
- 普遍存於多種程式語言中，例如JAVA、C++、R、JavaScript等都支援lambda

- 自訂函數跟lambda的比較

	function	lambda
是否要定義名稱	有(def add(x):)	無
運算式行數	無限制	僅能一行
自動回傳結果	不一定(有寫return才會回傳)	是

lambda()

lambda與自定函數的比較




```python
# 沒有參數
ans1 = lambda : 2020
print(ans1())      # 2020

# 一個參數
ans2 = list(map(lambda x:x*10, [i for i in range(1,6)]))
print(ans2)        # [10, 20, 30, 40, 50]

# 二個參數
ans3 = lambda x, y : x+y
print(ans3(2,9))   # 11

# 可變參數*args
ans4 = lambda *args : sum(args)
print(ans4(2, 100, 10.5))    # 112.5

# 可變參數**kwargs
ans5 = lambda **kwargs : kwargs
print(ans5(name="小智", age=15))  # {'name': '小智', 'age': 15}
```