CSS112 Computer Programming

ดร. พีระจักร์ วิฑูรชาติ

Contents

- Map
- Zip
- Reduce
- Filter
- Any,All
- Enumerate
- Sum,min,max

Map

- Map เป็นการทำแผนที่ one-to-one mapping ระหว่าง function และ iterable
- Map บังคับฟังก์ชันลงแต่ละค่าของ iterable
- มักจะใช้คู่กับ lambda function แต่ใช้ function ปกติก็ได้

Map

```
def add_one(n):
    return n+1

numbers = [1, 2, 3, 4, 5]
result = map(add_one, numbers)
print(result)
print(type(result))
print(list(result))
```

```
<map object at 0x10a3c7150>
<class 'map'>
[2, 3, 4, 5, 6]
```

Map

```
numbers = (1, 2, 3, 4, 5)
result = map(lambda x: x + 1, numbers)
print(tuple(result))
```

```
(2, 3, 4, 5, 6)
```

Map

```
>>> def decode(word):
...     decode_dict = {'happy':'sad','home':'school'}
...     return decode_dict.get(word,word)
...
>>> llist = ['I', 'am', 'a' , 'happy', 'child', 'at', 'home']
>>> anslist = list(map(decode,llist))
>>> print(anslist)
['I', 'am', 'a', 'sad', 'child', 'at', 'school']
```

- Notice dict.get()
- Dict.get(key, default_ans)
- ใช้บ่อยเพื่อให้เกิดคำตอบที่คาดล่วงหน้าได้หากหาคีย์ไม่เจอ

Map

In this example, corresponding items of two lists are added.

```
num1 = [4, 5, 6]
num2 = [5, 6, 7]

result = map(lambda n1, n2: n1+n2, num1, num2)
print(list(result))
```

Run Code »

Output

```
[9, 11, 13]
```

- ลิสต์สองลิสต์ใช้ map เพื่อให้ได้คำตอบ list เดียว

Map

```
>>> thai_str_list = ['ก','ข','คาใจ','เงื่องำ']
>>> d_thai_str_list = list(map(lambda x,y : x*y, thai_str_list, range(1,5)))
>>> print(d_thai_str_list)
['ก', 'ขข', 'คาใจคาใจคาใจ', 'เงื่องำเงื่องำเงื่องำเงื่องำ']
>>> d_thai_str_tuple = list(map(lambda x,y : x*y, thai_str_list, range(1,5)))
>>> print(d_thai_str_tuple)
['ก', 'ขข', 'คาใจคาใจคาใจ', 'เงื่องำเงื่องำเงื่องำเงื่องำ']
```

- 2 iterable โดยใช้ lambda function สร้าง list เดียว

ลองดู

- ใช้map สร้างลิสต์ที่มีค่าเป็น 0.5x^2 จากลิสต์เริ่มต้น
- เช่น ลิสต์เริ่มต้น = [2,3,4] หลังจาก map จะได้คำตอบ [2,4.5,8]

zip

```
>>> thai_str_list = ['ก','ข','คาใจ','เงื่องง่า']
>>> dict(zip(  thai_str_list, range(1,5)))
{'ก': 1, 'ข': 2, 'คาใจ': 3, 'เงื่องง่า': 4}
```

```
>>> dict(zip( map(lambda x,y : x*y, thai_str_list, range(1,5)), map(lambda x: x**2 , range(4))))
{'ก': 0, 'ขข': 1, 'คาใจคาใจคาใจ': 4, 'เงื่องง่าเงื่องง่าเงื่องง่าเงื่องง่า': 9}
```

```
>>> zip(['a','b','c'],[2,3,4])
<zip object at 0x7fdd5fc57f40>
>>> tuple(zip(['a','b','c'],[2,3,4]))
(('a', 2), ('b', 3), ('c', 4))
```

- ขนาดของสิ่งที่จะzipต้องเท่ากัน zipจะให้ผลคู่กัน เช่น tuple(zip([a,b,c],[2,3,4])) = (('a', 2), ('b', 3), ('c', 4))

zip

```
names = ['Apple', 'Google', 'Microsoft']
ages = ['44', '21', '44']
values = ['100', '80', '60']

mapped_values = list(zip(names, ages, values))
print(mapped_values)
```

```
[('Apple', '44', '100'), ('Google', '21', '80'), ('Microsoft', '44', '60')]
```

Zip: unpack

But how about unpack?

Simple, just similar to unpanc tuple, we add the `*` to the object that we want to unpack

```
names, ages, values = zip(*mapped_values)
print(f"The names is {names}")
print(f"The ages is {ages}")
print(f"The values is {values}")
```

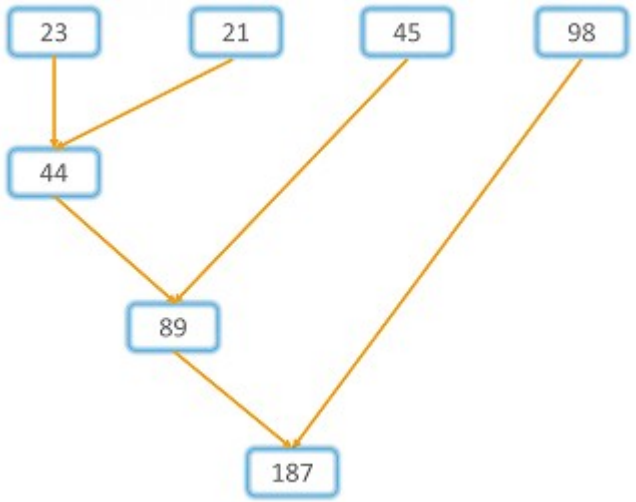
```
The names is ('Apple', 'Google', 'Microsoft')
The ages is ('44', '21', '44')
The values is ('100', '80', '60')
```

ลองดู

- สร้าง dict ซึ่งมี key คือ 0-9 ทั้งหมด 10 key และมี value คือ keyยกกำลังสอง เช่น {0:0,1:1,2:4,3:9...}

Reduce

```
>>> from functools import reduce
>>> reduce(lambda a,b: a+b,[23,21,45,98])
187
```



- Reduce คำนวนfunction บน iterable ทั้งหมด len(iterable)-1 ครั้ง โดยมีลำดับการคำนวนตามรูป

Reduce

Code:

```
from functools import reduce

# pre-defined function to calculate minimum
def mini(a, b):
    return a if a < b else b

# pre-defined function to calculate maximum
def maxi(a, b):
    return a if a > b else b


nums = [3, 5, 2, 4, 7, 1]

# passing both functions in the reduce along with nums as iterable
print('The minimum in the given list is', reduce(mini, nums))
print('The maximum in the given list is', reduce(maxi, nums))
```

Output:

```
The minimum in the given list is 1
The maximum in the given list is 7
```

- ตัวอย่างการหา max, min ของลิสต์ด้วย reduce

Reduce on empty list

Applying Reduce Function to an Empty List

You must wonder what will happen if an empty list is passed as an argument to the reduce function. Let's check it with the help of an example.

Code:

```
from functools import reduce

ans = reduce(lambda a, b: a + b, [])
print(ans)
```

Output:

```
Traceback (most recent call last):
  File "C:\Users\NAMANJEET SINGH\Documents\Scratch.py", line 24, in <module>
    ans = reduce(lambda a, b: a + b, [])
TypeError: reduce() of empty sequence with no initial value
```

Reduce with initializer

```
>>> def my_add(a, b):
...     result = a + b
...     print(f"{a} + {b} = {result}")
...     return result
```

```
>>> from functools import reduce

>>> numbers = [0, 1, 2, 3, 4]

>>> reduce(my_add, numbers, 100)
100 + 0 = 100
100 + 1 = 101
101 + 2 = 103
103 + 3 = 106
106 + 4 = 110
110
```

ลองดู

- สร้างสตริงค์ ‘0123456789’ จาก list(range(10)) โดยใช้ reduce
- str(1) = ‘1’

Filter

```
def func(variable):
    letters = ['a', 'e', 'i', 'o', 'u']
    if (variable.lower() in letters):
        return True
    else:
        return False

# given sequance
sequance = ['I', 'l', 'o', 'v', 'e', 'p', 'y', 't', 'h', 'o', 'n']

filtered = list(filter(func, sequance))
print(f"The vowel in the sequance is {filtered}")
```

```
The vowel in the sequance is ['I', 'o', 'e', 'o']
```

ลองดู

- จงสร้างลิสต์ที่มีแต่เลขคู่จากลิสต์knum
- จงสร้างลิสต์ที่มีแต่เลขบวกจากลิสต์knum

```
>>> knum=list(range(-5,5))
>>> knum
[-5, -4, -3, -2, -1, 0, 1, 2, 3, 4]
```

Filter

```python
def positive(num):
    if num > 0:
        return True
    else:
        return False

# odd or even number
def even_number(num):
    if num % 2 == 0:
        return True
    else:
        return False

numbers = [1, -3, 5, -20, 0, 9, 12]
positive_number = list(filter(positive, numbers))
even_number = list(filter(even_number, numbers))

print(f"The positive number is: {positive_number}.")
print(f"The even number is {even_number}.")
```

```
The positive number is: [1, 5, 9, 12].
The even number is [-20, 0, 12].
```

All

```
>>> all([1, 1, 1, 1, 1])
True

>>> all([1, 1, 1, 0, 1])
False

>>> all([])
True
```

```
>>> all(map(lambda x: x%2==0,map(lambda x:x*2,range(10))))
True
>>> all(map(lambda x: x%2==0,range(10)))
False
```

- Check if all elements are true

Any

```
>>> a = list(map(lambda i: 0.00005*i+i, range(10)))
>>> a
[0.0, 1.00005, 2.0001, 3.00015, 4.0002, 5.00025, 6.0003, 7.00035, 8.0004, 9.00045]
>>> list(map(lambda a,b: a-b <1e-4,a,range(10)))
[True, True, False, False, False, False, False, False, False, False]
>>> all(map(lambda a,b: a-b <1e-4,a,range(10)))
False
>>> any(map(lambda a,b: a-b <1e-4,a,range(10)))
True
>>>
```

- Check if atleast elements one is true

ลองดู

- ตรวจสอบว่าถ้า a=[1,2,3,4], b=[2,3,4,5] , ให้คิดว่า a,b เป็นเว็คเตอร์ในทางคณิตศาสตร์ a-b จะมีค่าเป็น 1 ทุกค่าหรือไม่
- ตรวจสอบว่าถ้า a=[1,2,3,4], b=[2,3,4,6] , ให้คิดว่า a,b เป็นเว็คเตอร์ในทางคณิตศาสตร์ a-b จะมีค่าเป็น 1 ทุกค่าหรือไม่
- ตรวจสอบว่าถ้า a=[1,2,3,4], b=[2,3,4,6] , ให้คิดว่า a,b เป็นเว็คเตอร์ในทางคณิตศาสตร์ a-b จะมีค่าบางค่ามากกว่า1หรือไม่

Sum,Min,Max

sum

The `sum` function takes an iterable of numbers and returns the sum of those numbers.

```
>>> sum([2, 1, 3, 4, 7])
17
```

There's not much more to it than that.

Python has lots of helper functions that **do the looping for you**, partly because they pair nicely with generator expressions:

```
>>> numbers = [2, 1, 3, 4, 7, 11, 18]
>>> sum(n**2 for n in numbers)
524
```

min and max

The `min` and `max` functions do what you'd expect: they give you the minimum and maximum items in an iterable.

```
>>> numbers = [2, 1, 3, 4, 7, 11, 18]
>>> min(numbers)
1
>>> max(numbers)
18
```

Enumerate

```
languages = ['Python', 'Java', 'JavaScript']

enumerate_prime = enumerate(languages)

# convert enumerate object to list
print(list(enumerate_prime))

# Output: [(0, 'Python'), (1, 'Java'), (2, 'JavaScript')]
```

Enumerate

```
grocery = ['bread', 'milk', 'butter']

for item in enumerate(grocery):
  print(item)

print('\n')

for count, item in enumerate(grocery):
  print(count, item)

print('\n')
# changing default start value
for count, item in enumerate(grocery, 100):
  print(count, item)
```

```
(0, 'bread')
(1, 'milk')
(2, 'butter')

0 bread
1 milk
2 butter

100 bread
101 milk
102 butter
```

ลองดู

- a = list(range(10,0,-1)
- สร้าง dict ซึ่งมี key เป็นตำแหน่งในลิสต์ของa

  {0: 10, 1: 9, 2: 8, 3: 7, 4: 6, 5: 5, 6: 4, 7: 3, 8: 2, 9: 1}