Split & Join

Pythonic Code

최성철 교수
Director of TEAMLAB

Split

Split 함수

- String Type의 값을 나눠서 List 형태로 변환

Python shell

```
>>> items = 'zero one two three'.split()  # 빈칸을 기준으로 문자열 나누기
>>> print (items)
['zero', 'one', 'two', 'three']
>>> example = 'python,jquery,javascript'  # ","을 기준으로 문자열 나누기
>>> example.split(",")
['python', 'jquery', 'javascript']
>>> a, b, c = example.split(",")
# 리스트에 있는 각 값을 a,b,c 변수로 unpacking
>>> example = 'cs50.gachon.edu'
>>> subdomain, domain, tld = example.split('.')
# "."을 기준으로 문자열 나누기 → Unpacking
```

Join 함수

- String List를 합쳐 하나의 String으로 반환할 때 사용

Python shell

```
>>> colors = ['red', 'blue', 'green', 'yellow']
>>> result = ''.join(colors)
>>> result
'redbluegreenyellow'
>>> result = ' '.join(colors) # 연결 시 빈칸 1칸으로 연결
>>> result
'red blue green yellow'
>>> result = ', '.join(colors) # 연결 시 ", "으로 연결
>>> result
'red, blue, green, yellow'
>>> result = '-'.join(colors)  # 연결 시 "-"으로 연결
>>> result
'red-blue-green-yellow'
```

Human knowledge belongs to the world.

List Comprehension

Pythonic Code

최성철 교수
Director of TEAMLAB

List comprehensions

- 기존 List 사용하여 간단히 다른 List를 만드는 기법
- 포괄적인 List, 포함되는 리스트라는 의미로 사용됨
- 파이썬에서 가장 많이 사용되는 기법 중 하나
- 일반적으로 for + append 보다 속도가 빠름

List comprehensions (1/4)

Python shell

```
>>> result = []
>>> for i in range(10):
...  result.append(i)
...
>>> result
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

General Style

Python shell

```
>>> result = [i for i in range(10)]
>>> result
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> result = [i for i in range(10) if i % 2 == 0]
>>> result
[0, 2, 4, 6, 8]
```

List Comprehension

List comprehensions (2/4)

Python shell

```
>>> word_1 = "Hello"
>>> word_2 = "World"
>>> result = [i+j for i in word_1 for j in word_2]
    # Nested For loop
>>> result
['HW', 'Ho', 'Hr', 'Hl', 'Hd', 'eW', 'eo', 'er',
'el', 'ed', 'lW', 'lo', 'lr', 'll', 'ld', 'lW',
'lo', 'lr', 'll', 'ld', 'oW', 'oo', 'or', 'ol', 'od']
```

List comprehensions (3/4)

Python shell

```
>>> case_1 = ["A","B","C"]
>>> case_2 = ["D","E","A"]
>>> result = [i+j for i in case_1 for j in case_2]
>>> result
['AD', 'AE', 'AA', 'BD', 'BE', 'BA', 'CD', 'CE', 'CA']
>>> result = [i+j for i in case_1 for j in case_2 if not(i==j)]
# Filter: i랑 j과 같다면 List에 추가하지 않음
>>> result
['AD', 'AE', 'BD', 'BE', 'BA', 'CD', 'CE', 'CA']
>>> result.sort()
>>> result
['AD', 'AE', 'BA', 'BD', 'BE', 'CA', 'CD', 'CE']
```

List comprehensions (4/4)

```
>>> words = 'The quick brown fox
jumps over the lazy dog'.split()
# 문장을 빈칸 기준으로 나눠 list로 변환

>>> print (words)
['The', 'quick', 'brown', 'fox',
'jumps', 'over', 'the', 'lazy', 'dog']
>>>
>>> stuff = [[w.upper(), w.lower(),
len(w)] for w in words]

# list의 각 elemente들을 대문자, 소문자, 길이
로 변환하여 two dimensional list로 변환
```

```
>>> for i in stuff:
...     print (i)
...
['THE', 'the', 3]
['QUICK', 'quick', 5]
['BROWN', 'brown', 5]
['FOX', 'fox', 3]
['JUMPS', 'jumps', 5]
['OVER', 'over', 4]
['THE', 'the', 3]
['LAZY', 'lazy', 4]
['DOG', 'dog', 3]
```

Two dimensional vs One dimensional

Python shell

```
>>> case_1 = ["A","B","C"]
>>> case_2 = ["D","E","A"]
['AD', 'AE', 'AA', 'BD', 'BE', 'BA', 'CD', 'CE', 'CA']
>>> result = [i+j for i in case_1 for j in case_2]
>>> result
>>> result = [ [i+j for i in case_1] for j in case_2]
>>> result
```

Human knowledge belongs to the world.

Enumerate & Zip

Pythonic Code

최성철 교수
Director of TEAMLAB

Enumerate

Enumerate

- List의 element를 추출할 때 번호를 붙여서 추출

Python shell

```
>>> for i, v in enumerate(['tic', 'tac', 'toe']):
# list의 있는 index와 값을 unpacking
...     print (i, v)
...
0 tic
1 tac
2 toe

>>> mylist = ["a","b","c","d"]
>>> list(enumerate(mylist)) # list의 있는 index와 값을 unpacking하여 list로 저장
[(0, 'a'), (1, 'b'), (2, 'c'), (3, 'd')]
>>> {i:j for i,j in enumerate('Gachon University is an academic institute
located in South Korea.'.split())}
# 문장을 list로 만들고 list의 index와 값을 unpacking하여 dict로 저장
{0: 'Gachon', 1: 'University', 2: 'is', 3: 'an', 4: 'academic', 5: 'institute',
6: 'located', 7: 'in', 8: 'South', 9: 'Korea.'}
```

Zip

Zip

두 개의 list의 값을 병렬적으로 추출함

Python shell

```
>>> alist = ['a1', 'a2', 'a3']
>>> blist = ['b1', 'b2', 'b3']
>>> for a, b in zip(alist, blist): # 병렬적으로 값을 추출
...  print (a,b)
...
a1 b1
a2 b2
a3 b3

>>> a,b,c =zip((1,2,3),(10,20,30),(100,200,300)) #각 tuple의 같은 index 끼리 묶음
(1, 10, 100) (2, 20, 200) (3, 30, 300)

>>> [sum(x) for x in zip((1,2,3), (10,20,30), (100,200,300))]
# 각 Tuple 같은 index를 묶어 합을 list로 변환
[111, 222, 333]
```

Enumerate & Zip

Python shell

```
>>> alist = ['a1', 'a2', 'a3']
>>> blist = ['b1', 'b2', 'b3']
>>>
>>> for i, (a, b) in enumerate(zip(alist, blist)):
...     print (i, a, b) # index alist[index] blist[index] 표시
...
0 a1 b1
1 a2 b2
2 a3 b3
```

Human knowledge belongs to the world.

Lambda & MapReduce

Pythonic Code

최성철 교수
Director of TEAMLAB

Lambda

Lambda

- 함수 이름 없이, 함수처럼 쓸 수 있는 익명함수
- 수학의 람다 대수에서 유래함

General function

```
def f(x, y):
    return x + y

print(f(1, 4))
```

Lambda function

```
f > lambda x-" y;" x , y
print)f)2-" 5**
```

Lambda

- Python 3부터는 권장하지는 않으나 여전히 많이 쓰임

```
f > lambda x-"y;"x , y
print)f)2-"5**
```

```
f > lambda x;"x ++ 3
print)f)4**
```

```
f > lambda x;"x 0 3
print)f)4**
```

```
print))lambda x;"x , 2*)6**
```

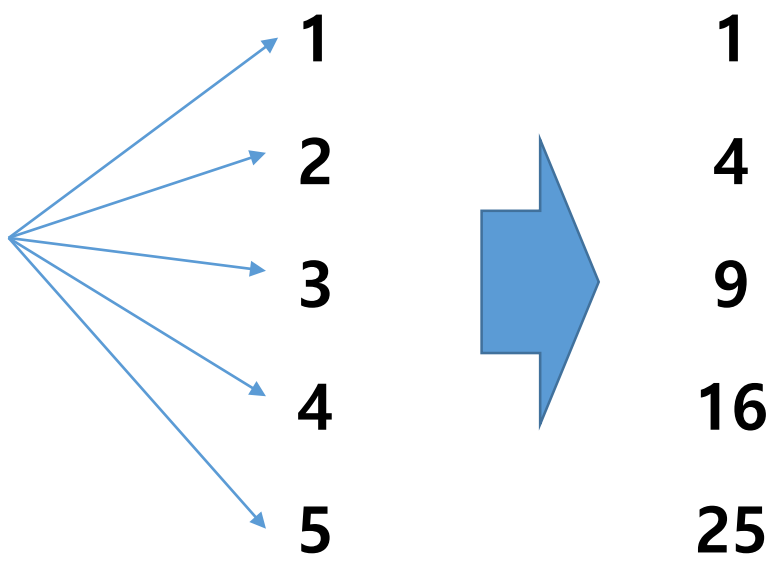
Map & Reduce

Map function

- Sequence 자료형 각 element에 동일한 function을 적용함

map(function_name, list_data)

lambda x: x ** 2




```
ex > B2-3-4-5-6D
f > lambda x;"x ++ 3
print)list)map)f-"ex***

f > lambda x-"y;"x , y
print)list)map)f-"ex-"ex***
```

Map function

- 두 개 이상의 list에도 적용 가능함, if filter도 사용가능

```
ex  >  B2-3-4-5-6D
f  >  lambda  x-"y;"x  ,   y
print)list)map)f-"ex-"ex***
```

```
list)
  map)
  lambda  x;"x  ++  3  if  x  &   3  >>  1
  else  x-"
  ex*
*
```

Map function

- python3 는 iteration을 생성 → list을 붙여줘야 list 사용가능
- 실행시점의 값을 생성, 메모리 효율적

```
ex = [1,2,3,4,5]
print(list(map(lambda x: x+x, ex)))
print((map(lambda x: x+x, ex)))

f  >  lambda  x;" x  ++  3
print)map)f-" ex**
for  i  in  map)f-" ex*;
      print)i*
```

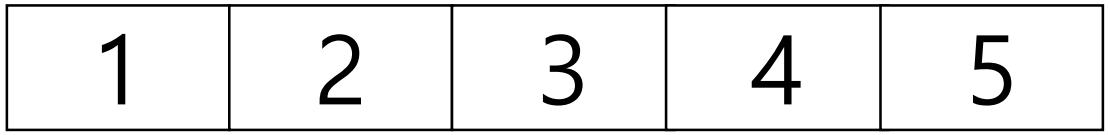
```
result = map(f, ex)
print(next(result))
```

Reduce function

- map function과 달리 list에 똑같은 함수를 적용해서 통합

```
from functools import reduce

print(reduce(lambda x, y: x+y, [1, 2, 3, 4, 5]))
```



Summary

- Lambda, map, reduce는 간단한 코드로 다양한 기능을 제공

- 그러나 코드의 직관성이 떨어져서 lambda나 reduce는 python3에서 사용을 권장하지 않음

- Legacy library나 다양한 머신러닝 코드에서 여전히 사용중

Human knowledge belongs to the world.

Asterisk

Pythonic Code

최성철 교수
Director of TEAMLAB

Asterisk

- 흔히 알고 있는 * 를 의미함

- 단순 곱셈, 제곱연산, 가변 인자 활용 등 다양한게 사용됨

***args**

```
def asterisk_test(a, *args):
    print(a, args)
    print(type(args))



asterisk_test(1,2,3,4,5,6)
```

****kargs**

```
def  asterisk_test)a-" ++kargs*;
        print)a-" kargs*
        print)type)kargs**


asterisk_test)2-" b>3-" c>4-"
d>5-" e>6-" f>7*
```

Asterisk – unpacking a container

- tuple, dict 등 자료형에 들어가 있는 값을 unpacking
- 함수의 입력값, zip 등에 유용하게 사용가능

```
def asterisk_test)a-" +args*;
    print)a-" args*
    print)type)args**


asterisk_test)2-" +)3-4-5-6-7**
```

```
def asterisk_test)a-" args*;
    print)a-" *args*
    print)type)args**


asterisk_test)2-" )3-4-5-6-7**
```

Asterisk – unpacking a container

```
a-" b-" c  >  )B2-" 3D-" B4-" 5D-" B6-" 7D*
print)a-" b-" c*


data  >  )B2-" 3D-" B4-" 5D-" B6-" 7D*
print)+data*
```

```
def  asterisk_test)a-" b-" c-" d-*;

        print)a-" b-" c-" d*

data  >  G"b";2  -" "c";3-" "d";4I

asterisk_test)21-" ++data*
```

```
for data in zip(*([1, 2], [3, 4], [5, 6])):
    print(data)
```

Human knowledge belongs to the world.

Lambda & MapReduce

Pythonic Code

최성철 교수
Director of TEAMLAB

Lambda

Lambda

- 함수 이름 없이, 함수처럼 쓸 수 있는 익명함수
- 수학의 람다 대수에서 유래함

General function

```
def f(x, y):
    return x + y

print(f(1, 4))
```

Lambda function

```
f > lambda x-" y;" x , y
print)f)2-" 5**
```

Lambda

- Python 3부터는 권장하지는 않으나 여전히 많이 쓰임

```
f > lambda x-"y;"x , y
print)f)2-"5**
```

```
f > lambda x;"x ++ 3
print)f)4**
```

```
f > lambda x;"x 0 3
print)f)4**
```

```
print))lambda x;"x , 2*)6**
```

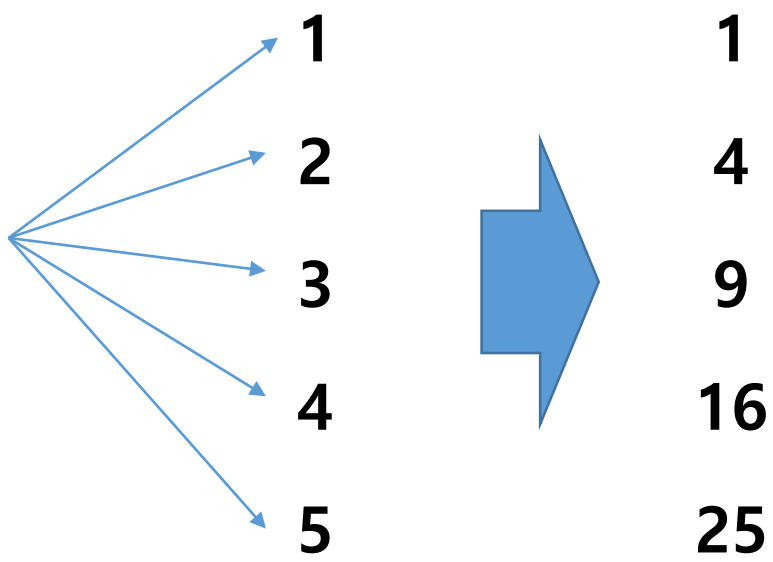
Map & Reduce

Map function

- Sequence 자료형 각 element에 동일한 function을 적용함

map(function_name, list_data)

lambda x: x ** 2




```
ex > B2-3-4-5-6D
f > lambda x;"x ++ 3
print)list)map)f-"ex***

f > lambda x-"y;"x , y
print)list)map)f-"ex-"ex***
```

Map function

- 두 개 이상의 list에도 적용 가능함, if filter도 사용가능

```
ex  >  B2-3-4-5-6D
f  >  lambda  x-"y;"x  ,   y
print)list)map)f-"ex-"ex***
```

```
list)
  map)
  lambda  x;"x  ++  3  if  x  &   3  >>  1
  else  x-"
  ex*
*
```

Map function

- python3 는 iteration을 생성 → list을 붙여줘야 list 사용가능
- 실행시점의 값을 생성, 메모리 효율적

```
ex = [1,2,3,4,5]
print(list(map(lambda x: x+x, ex)))
print((map(lambda x: x+x, ex)))

f  >  lambda  x;" x  ++  3
print)map)f-" ex**
for  i  in  map)f-" ex*;
      print)i*
```

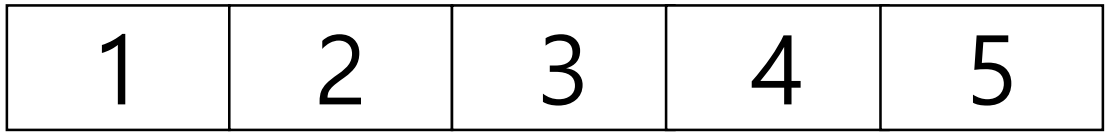
```
result = map(f, ex)
print(next(result))
```

Reduce function

- map function과 달리 list에 똑같은 함수를 적용해서 통합

```
from functools import reduce

print(reduce(lambda x, y: x+y, [1, 2, 3, 4, 5]))
```



Summary

- Lambda, map, reduce는 간단한 코드로 다양한 기능을 제공

- 그러나 코드의 직관성이 떨어져서 lambda나 reduce는 python3에서 사용을 권장하지 않음

- Legacy library나 다양한 머신러닝 코드에서 여전히 사용중

Human knowledge belongs to the world.

Linear algebra codes

Pythonic Code

최성철 교수
Director of TEAMLAB

Vector representation of python

- Vector를 파이썬으로 표시하는 다양한 방법 존재

```
vector_a = [1, 2, 10] # List로 표현했을 경우
vector_b= (1, 2, 10) # Tuple로 표현했을 경우
vector_c = {'x': 1, 'y': 1, 'z': 10} # dict 표현했을 경우

print(vector_a, vector_b, vector_c)
```

- 최선의 방법은 없음
- 값의 변경 유무, 속성값 유무에 따라 선택할 수 있음
- 본 수업에서는 기본적으로 list로 vector 연산 실시

Vector의 계산

```
u = [2, 2]
v = [2, 3]
z = [3, 5]
result = []
for i in range(len(u)):
    result.append(u[i] + v[i] + z[i])

print(result)
```

[2, 2]+[2, 3]+[3, 5]=[7, 10]

이런 코드는 쓰면 안됨
파이썬 답지 못하고
안 아름다움

Vector handling with python

- Python은 특유의 간결성이 최대의 장점
- Vector와 같은 수학 연산을 복잡하게 표현한다면 사용이 어려움
- 최대한 파이썬만의 특징을 살려서 간단하게 연산을 표시
- Comprehension과 zip 같은 pythonic technique을 적극 활용

https://goo.gl/ATmcYo

Vector의 계산

```
u = [2, 2]
v = [2, 3]
z = [3, 5]

result = [sum(t) for t in zip(u,v,z)]
print (result)
```

[2, 2]+[2, 3]+[3, 5]=[7, 10]

Vector의 계산: Scalar-Vector product

```
u = [1, 2, 3]
v = [4, 4, 4]
alpha = 2

result = [alpha*sum(t) for t in zip(u,v)]
print(result)
```

$$2([1,2,3]+[4,4,4]) = 2[5,6,7] = [10,12,14]$$

Matrix representation of python

- Matrix 역시 Python으로 표시하는 다양한 방법이 존재

```
matrix_a = [[3, 6], [4, 5]] # List로 표현했을 경우
matrix_b = [(3, 6), (4, 5)] # Tuple로 표현했을 경우
matrix_c = {(0 ,0): 3, (0 ,1): 6, (1 ,0): 4, (1 ,1): 5} # dict 표현했을 경우
```

- 특히 dict로 표현할 때는 무궁무진한 방법이 있음

- 본 수업에서는 기본적으로 two-dimensional list 형태로 표현함

- [[1번째 row], [2번째 row], [3번째 row]]

Matrix의 계산: Matrix addition

$$C = A + B = \begin{bmatrix} 3 & 6 \\ 4 & 5 \end{bmatrix} + \begin{bmatrix} 5 & 8 \\ 6 & 7 \end{bmatrix} = \begin{bmatrix} 8 & 14 \\ 10 & 12 \end{bmatrix}$$

```
matrix_a = [[3, 6], [4, 5]]
matrix_b = [[5, 8], [6, 7]]
result = [[sum(row) for row in zip(*t)] for t in zip(matrix_a, matrix_b)]

print(result)
```

Matrix의 계산: Scalar-Matrix Product

$$\alpha \times A = 4 \times \begin{bmatrix} 3 & 6 \\ 4 & 5 \end{bmatrix} = \begin{bmatrix} 12 & 24 \\ 16 & 20 \end{bmatrix}$$

```
matrix_a = [[3, 6], [4, 5]]
alpha = 4
result = [[alpha * element for element in t] for t in matrix_a]

print(result)
```

Matrix의 계산: Matrix Transpose

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}, A^T = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

```
matrix_a = [[1, 2, 3], [4, 5, 6]]
result = [ [element for element in t] for t in zip(*matrix_a) ]
print (result)
```

Matrix의 계산: Matrix Product

$$\boldsymbol{A} = \begin{bmatrix} \mathbf{1} & \mathbf{2} & \mathbf{3} \\ \mathbf{4} & \mathbf{5} & \mathbf{6} \end{bmatrix}, \boldsymbol{B} = \begin{bmatrix} \mathbf{1} & \mathbf{4} \\ \mathbf{2} & \mathbf{5} \\ \mathbf{3} & \mathbf{6} \end{bmatrix} \text{ 이면 } \quad C = A \times B = \begin{bmatrix} 1 & 1 & 2 \\ 2 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 1 & 3 \end{bmatrix} = \begin{bmatrix} 5 & 8 \\ 5 & 6 \end{bmatrix}$$

```
matrix_a = [[1, 1, 2], [2, 1, 1]]
matrix_b = [[1, 1], [2, 1], [1, 3]]
result = [[sum(a * b for a, b in zip(row_a, column_b)) ₩
               for column_b in zip(*matrix_b)] for row_a in matrix_a]

print(result)
```

두 개이상의
Argument가 존재할 때는?

Human knowledge belongs to the world.