Python data structure

TEAMLAB director
최성철

특징이 있는 정보는

어떻게 저장하면 좋을까?

데이터 구조 생각해보기

- 전화번호부 정보는 어떻게 저장하면 좋을까?
- 은행 번호표 정보는 어떻게 처리하면 좋을까?
- 서적 정보는 어떻게 관리하면 좋을까?
- 창고에 쌓인 수화물의 위치를 역순으로 찾을 때?

파이썬 기본 데이터 구조

- 스택과 큐(stack & queue with list)
- 튜플과 집합(tuple & set)
- 사전(dictionary)
- Collection 모듈

문자열 (string)

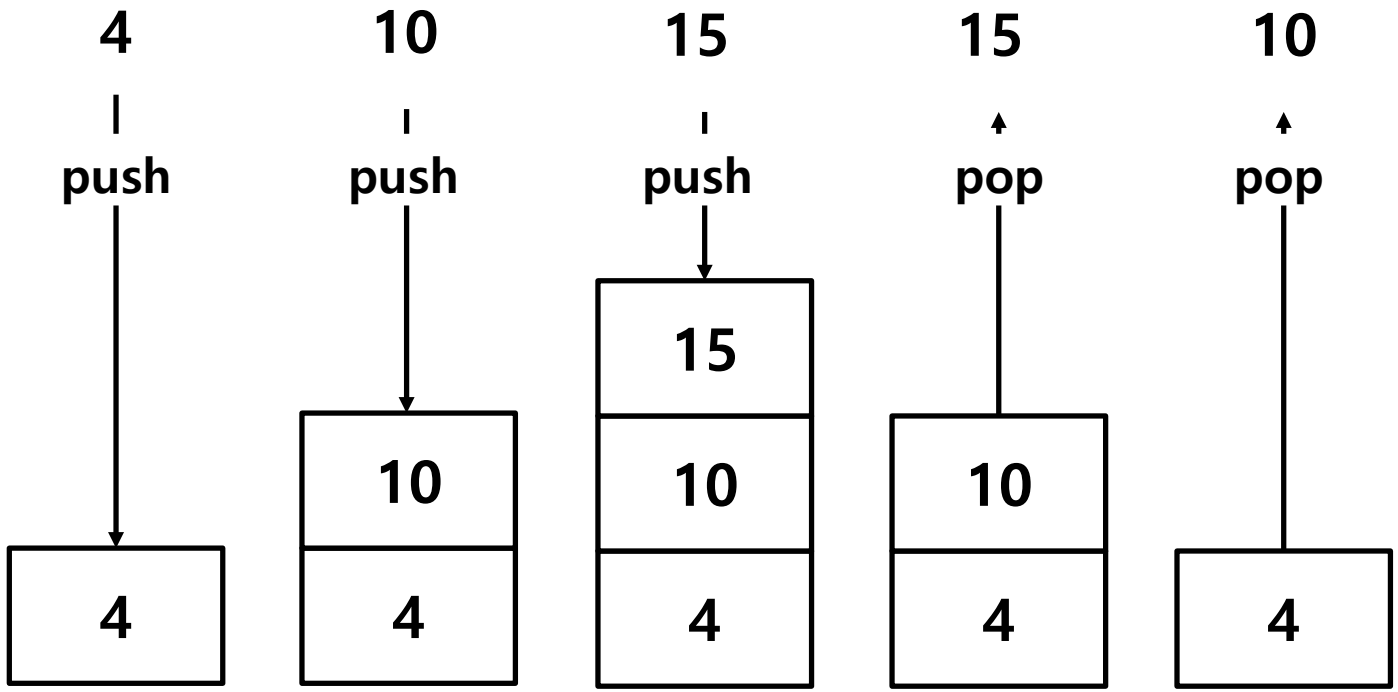
- 시퀀스 자료형으로 문자형 data를 메모리에 저장
- 영문자 한 글자는 1byte의 메모리공간을 사용

```
>>> import sys          # sys 모듈을 호출
>>> print (sys.getsizeof("a"), sys.getsizeof("ab"), sys.getsizeof("abc"))
50 51 52 # “a”, “ab”, “abc”의 각 메모리 사이즈 출력
```

stack

스택 (Stack)

- 나중에 넣은 데이터를 먼저 반환하도록 설계된 메모리 구조
- Last In First Out (LIFO)
- Data의 입력을 Push, 출력을 Pop이라고 함



스택 (Stack) with list object

- 리스트를 사용하여 스택 구조를 구현 가능
- push를 append(), pop을 pop()를 사용

```
>>> a = [1,2,3,4,5]
>>> a.append(10)
>>> a.append(20)
>>> a.pop()          # 20 출력
20
>>> a.pop()          # 10 출력
10
>>>
```

stack example

- 스택 구조를 활용, 입력된 글자를 역순으로 출력

```
word = input("Input a word : ")      # Input Word
word_list = list(word)               # String to List
for i in range(len(word_list)):
    print (word_list.pop())          # 하나씩 빼면서 출력
```

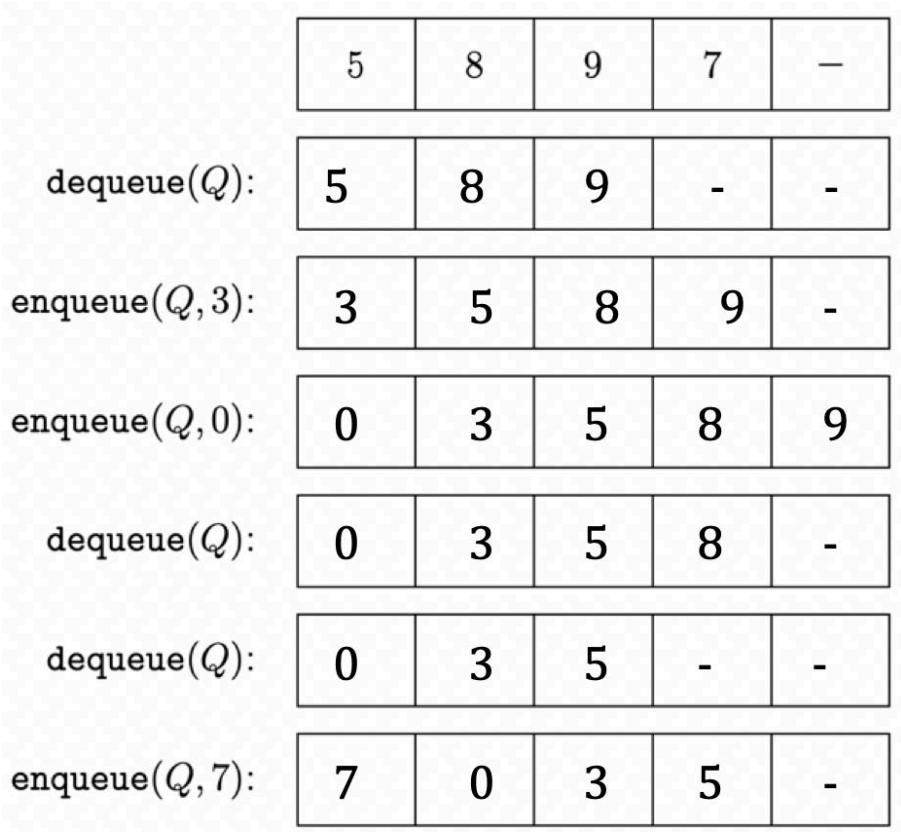
queue

큐 (Queue)

- 먼저 넣은 데이터를 먼저 반환하도록 설계된 메모리 구조

- First In First Out (FIFO)

- Stack과 반대되는 개념



http://bit.ly/38B3Xo3

큐 (Queue) with list object

-파이썬은 리스트를 사용하여 큐 구조를 활용

-put를 append(), get을 pop(0)를 사용

```
>>> a = [1,2,3,4,5]
>>> a.append(10)
>>> a.append(20)
>>> a.pop(0) # 1 출력
1
>>> a.pop(0) # 2 출력
2
>>>
```

tuple

튜플 (tuple)

- 값의 변경이 불가능한 리스트
- 선언 시 "[ ]" 가 아닌 "( )"를 사용
- 리스트의 연산, 인덱싱, 슬라이싱 등을 동일하게 사용

```
>>> t = (1,2,3)
>>> print (t + t , t * 2) # (1, 2, 3, 1, 2, 3) (1, 2, 3, 1, 2, 3)
(1, 2, 3, 1, 2, 3) (1, 2, 3, 1, 2, 3)
>>> len(t)                  # 3
3
>>> t[1] = 5       # Error 발생
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
>>>
```

튜플 (tuple)

왜 쓸까?

- 프로그램을 작동하는 동안 변경되지 않은 데이터의 저장
  Ex) 학번, 이름, 우편번호 등등
- 함수의 반환 값등 사용자의 실수에 의한 에러를 사전에 방지

```
>>> t = (1)    # 일반정수로 인식
1
>>> t = (1, ) # 값이 하나인 Tuple은 반드시 "," 를 붙여야 함
(1,)
```

set

집합 (set)

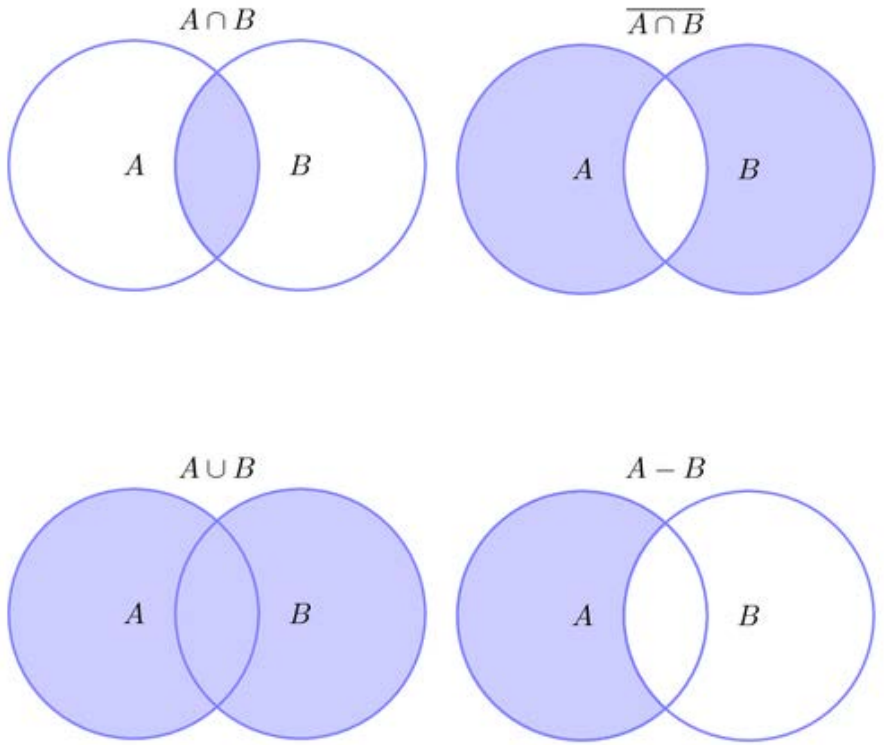
- 값을 순서없이 저장, 중복 불허 하는 자료형
- set 객체 선언을 이용하여 객체 생성

```
>>> s = set([1,2,3,1,2,3])      #  set 함수를 사용 1,2,3을 집합 객체 생성 , a = {1,2,3,4,5} 도 가능
>>> s
{1, 2, 3}
>>> s.add(1)                   #  한 원소 1만 추가,  추가, 중복불허로 추가 되지 않음
>>> s
{1, 2, 3}
>>> s.remove(1)                #  1 삭제
>>> s
{2, 3}
>>> s.update([1,4,5,6,7])      #  [1,4,5,6,7] 추가
>>> s
{1, 2, 3, 4, 5, 6, 7}
>>> s.discard(3)               #  3 삭제
>>> s
{1, 2, 4, 5, 6, 7}
>>> s.clear()                  # 모든 원소 삭제
```

집합의 연산

- 수학에서 활용하는 다양한 집합연산 가능

```
>>> s1 = set([1,2,3,4,5])
>>> s2 = set([3,4,5,6,7])
>>> s1.union(s2)       #  s1 과 s2의 합집합
{1, 2, 3, 4, 5, 6, 7}
>>> s1 | s2              # set([1, 2, 3, 4, 5, 6, 7])
{1, 2, 3, 4, 5, 6, 7}
>>> s1.intersection(s2)    #  s1 과 s2의 교집합
{3, 4, 5}
>>> s1 & s2                # set([3, 4, 5])
{3, 4, 5}
>>> s1.difference(s2)      # s1 과 s2의 차집합
{1, 2}
>>> s1 - s2                # set([1, 2])
{1, 2}
```



dict

사전 (dictionary)

- 데이터를 저장 할 때는 구분 지을 수 있는 값을 함께 저장
  예) 주민등록 번호, 제품 모델 번호
- 구분을 위한 데이터 고유 값을 Identifier 또는 Key 라고함
- Key 값을 활용하여, 데이터 값(Value)를 관리함

학번	이름	생년월일	주소
20150230	홍길동	1995-04-03	서울시 동대문구
20150233	김은정	1995-04-20	성남시 분당구
20150234	오영심	1996-01-03	성남시 중원구
20150236	최성철	1995-12-27	인천시 계양구

사전 (dictionary)

- key와 value를 매칭하여 key로 value를 검색
- 다른 언어에서는 Hash Table 이라는 용어를 사용
- {Key1:Value1, Key2:Value2, Key3:Value3 ...} 형태

```
student_info = {20140012:'Sungchul',
20140059:'Jiyong',20140058:'JaeHong'}

student_info[20140012]
student_info[20140012] = 'Janhyeok'
student_info[20140012]
student_info[20140039] = 'wonchul'
student_info
```

Key	Value
20140012	Janhyeok
20140059	Jiyong
20140058	JaeHong
20140039	Wonchul

사전 (dictionary) 다루기


```
>>> country_code = {}   # Dict 생성, country_code = dict() 도 가능
>>> country_code = { " America " : 1,  " Korea " : 82,  " China " : 86,  " Japan " : 81}
>>> country_code
{ ' America ' : 1,  ' China ' : 86,  ' Korea ' : 82,  ' Japan ' : 81}
>>> country_code.items() # Dict 데이터 출력
Dict_items([( ' America ' , 1), ( ' China ' , 86), ( ' Korea ' , 82), ( ' Japan ' , 81)])
>>> country_code.keys()  # Dict 키 값만 출력
Dict_keys(["America", "China", "Korea", "Japan"])
>>> country_code["German"]= 49  # Dict 추가
>>> country_code
{'America': 1, 'German': 49, 'China': 86, 'Korea': 82, 'Japan': 81}
>>> country_code.values()       # Dict Value만 출력
dict_values([1, 49, 86, 82, 81])
```

사전 (dictionary) 다루기


```
>>> for k,v in country_code.items():
...     print ("Key : ", k)
...     print ("Value : ", v)
...
Key :  America
Value :  1
Key :  Gernman
Value :  49
Key :  China
Value :  86
Key :  Korea
Value :  82
Key :  Japan
Value :  81
>>> "Korea" in country_code.keys()  # Key값에 "Korea"가 있는지 확인
True
>>> 82 in country_code.values()     # Value값에 82가 있는지 확인
True
```

Lab - dict

Command Analyzer

- Command: 사용자가 서버에 명령어를 입력한 명령어

어떤 사용자가
얼마나 많이
명령어를 입력하였을 까?

Data source:
https://bit.ly/3nR3qFa

Command Analyzer


```
import csv

def getKey(item):                    # 정렬을 위한 함수
    return item[1]                   # 신경 쓸 필요 없음

command_data = []                    # 파일 읽어오기
with open("command_data.csv", "r") as csvfile:
     spamreader = csv.reader(csvfile, delimiter=',', quotechar='"')
     for row in spamreader:
        command_data.append(row)

command_counter = {}                             # dict 생성, 아이디를 key값, 입력줄수를 value값
for data in command_data:                        # list 데이터를 dict로 변경
    if data[1] in command_counter.keys():        # 아이디가 이미 Key값으로 변경되었을 때
        command_counter[data[1]] += 1                        # 기존 출현한 아이디
    else:
        command_counter[data[1]] = 1                         # 처음 나온 아이디

dictlist = []                                                # dict를 list로 변경
for key, value in command_counter.items():
    temp = [key,value]
    dictlist.append(temp)

sorted_dict= sorted(dictlist, key=getKey, reverse=True) # list를 입력 줄 수로  정렬
print (sorted_dict[:100])                                    # List의 상위 10객값만 출력
```

collections

collections

- List, Tuple, Dict에 대한 Python Built-in 확장 자료 구조(모듈)
- 편의성, 실행 효율 등을 사용자에게 제공함
- 아래의 모듈이 존재함

```
from collections import deque
from collections import Counter
from collections import OrderedDict
from collections import defaultdict
from collections import namedtuple
```

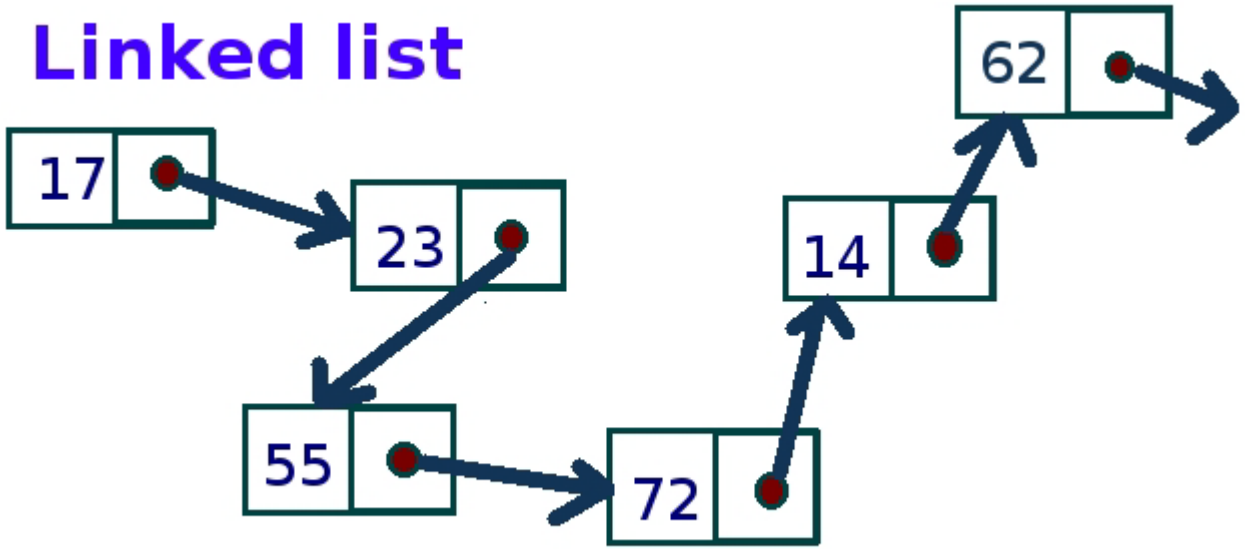
deque

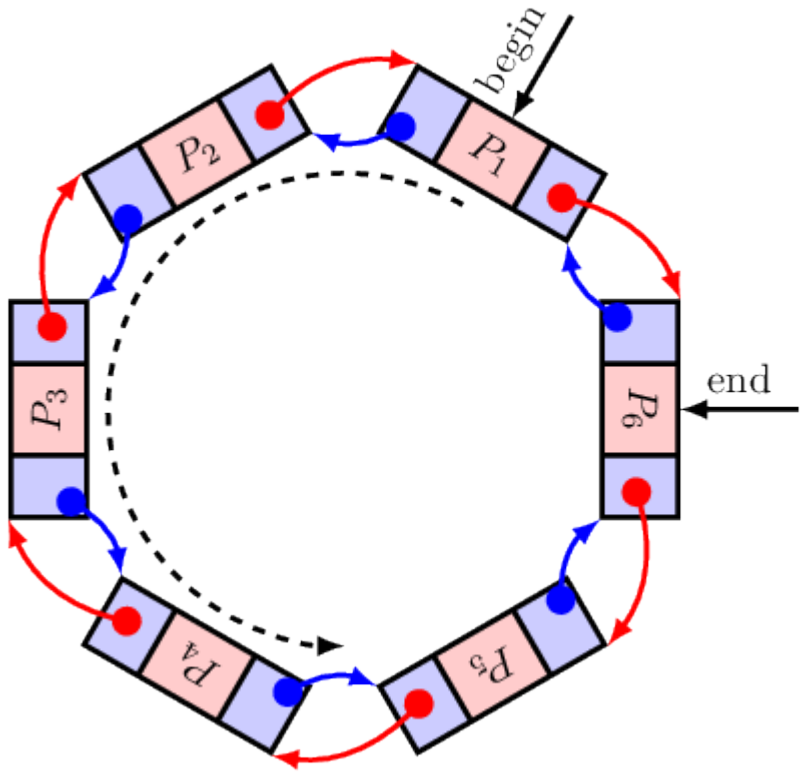
- Stack과 Queue 를 지원하는 모듈
- List에 비해 효율적인=빠른 자료 저장 방식을 지원함

```
from collections import deque

deque_list = deque()
for i in range(5):
    deque_list.append(i)
print(deque_list)
deque_list.appendleft(10)
print(deque_list)
```

deque

- rotate, reverse등 Linked List의 특성을 지원함
- 기존 list 형태의 함수를 모두 지원함





deque

- rotate, reverse등 Linked List의 특성을 지원함
- 기존 list 형태의 함수를 모두 지원함

```
deque_list.rotate(2)
print(deque_list)
deque_list.rotate(2)
print(deque_list)
print(deque(reversed(deque_list)))

deque_list.extend([5, 6, 7])
print(deque_list)
deque_list.extendleft([5, 6, 7])
print(deque_list)
```

deque

- deque 는 기존 list보다 효율적인 자료구조를 제공
- 효율적 메모리 구조로 처리 속도 향상

deque

```
from collections import deque
import time

start_time = time.clock()
deque_list = deque()
# Stack
for i in range(10000):
    for i in range(10000):
        deque_list.append(i)
        deque_list.pop()
print(time.clock() - start_time, "seconds")
```

general list

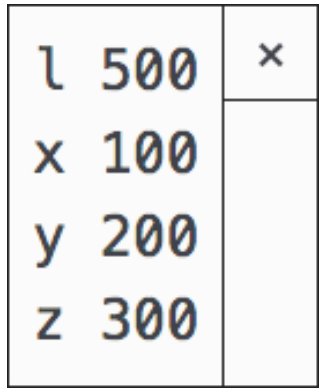
```
import time


start_time = time.clock()
just_list = []
for i in range(10000):
    for i in range(10000):
        just_list.append(i)
        just_list.pop()
print(time.clock() - start_time, "seconds")
```

OrderedDict

OrderedDict

- Dict와 달리, 데이터를 입력한 순서대로 dict를 반환함
- 그러나 dict도 python 3.6 부터 입력한 순서를 보장하여 출력함

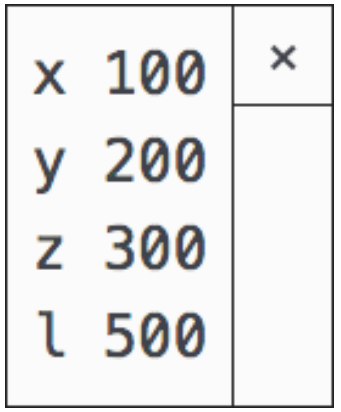
```
d = {}
d['x'] = 100
d['y'] = 200
d['z'] = 300
d['l'] = 500
```




```
for k, v in d.items():
    print(k, v)
```

×

```
from collections import OrderedDict

d = OrderedDict()
d['x'] = 100
d['y'] = 200
d['z'] = 300
d['l'] = 500
```




```
for k, v in d.items():
    print(k, v)
```

OrderedDict

- Dict type의 값을, value 또는 key 값으로 정렬할 때 사용 가능

```
for k, v in OrderedDict(sorted(d.items(), key=lambda t: t[0])).items():
    print(k, v)
```

```
l 500
x 100
y 200
z 300
```

```
for k, v in OrderedDict(sorted(d.items(), key=lambda t: t[1])).items():
    print(k, v)
```

```
x 100
y 200
z 300
l 500
```

defaultdict

defaultdict

- Dict type의 값에 기본 값을 지정, 신규값 생성시 사용하는 방법

```
d = dict()
print(d["first"])
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call l
<ipython-input-1-3b4ee8f5b4c3> in <module>()
      1 d = dict()
----> 2 print(d["first"])

KeyError: 'first'
```

defaultdict

- Dict type의 값에 기본 값을 지정, 신규값 생성시 사용하는 방법

```
from collections import defaultdict
d = defaultdict(object)      # Default dictionary를 생성
d = defaultdict(lambda: 0)  # Default 값을 0으로 설정합
print(d["first"])
```

defaultdict

- 하나의 지문에 각 단어들이 몇 개나 있는지 세고 싶을경우?
- Text-mining 접근법 - Vector Space Model

```
text = """A press release is the quickest and easiest way to get free publicity. If well written, a press release can result in multiple published articles about your firm and its products. And that can mean new prospects contacting you asking you to sell to them. ….""".lower().split()
```

```
print(text)
```

```
['a', 'press', 'release', 'is', 'the', 'quickest', 'and', 'easiest', 'way', 'to', 'get', 'f
```

defaultdict


```
from collections import OrderedDict
word_count = defaultdict(lambda: 0)  # Default 값을 0으로 설정합
for word in text:
    word_count[word] += 1
for i, v in OrderedDict(sorted(
        word_count.items(), key=lambda t: t[1],
        reverse=True)).items():
    print(i, v)
```

```
a 12
to 10
and 9
the 9
press 8
release 8
that 7
of 5
your 4
```

Counter

Counter

- Sequence type의 data element들의 갯수를 dict 형태로 반환

```
from collections import Counter

c = Counter()                           # a new, empty counter
c = Counter('gallahad')                 # a new counter from an iterable
print(c) Counter({'a': 3, 'l': 2, 'g': 1, 'd': 1, 'h': 1})
```

Counter

- Dict type, keyword parameter 등도 모두 처리 가능

```
c = Counter({'red': 4, 'blue': 2})      # a new counter from a mapping
print(c)
print(list(c.elements()))
```

```
Counter({'red': 4, 'blue': 2})
['blue', 'blue', 'red', 'red', 'red', 'red']
```

```
c = Counter(cats=4, dogs=8)             # a new counter from keyword args
print(c)
print(list(c.elements()))
```

```
Counter({'dogs': 8, 'cats': 4})
['dogs', 'dogs', 'dogs', 'dogs', 'dogs', 'dogs', 'dogs', 'dogs', 'cats', 'cats', 'cats', 'cats']
```

Counter

- Set의 연산들을 지원함

```
c = Counter(a=4, b=2, c=0, d=-2)
d = Counter(a=1, b=2, c=3, d=4)
c.subtract(d)  # c- d
print(c) Counter({'a': 3, 'b': 0, 'c': -3, 'd': -6})
```

Counter

- Set의 연산들을 지원함

```
c = Counter(a=4, b=2, c=0, d=-2)
d = Counter(a=1, b=2, c=3, d=4)
print(c + d)
print(c & d)
print(c | d)
```

```
Counter({'a': 5, 'b': 4, 'c': 3, 'd': 2})
Counter({'b': 2, 'a': 1})
Counter({'a': 4, 'd': 4, 'c': 3, 'b': 2})
```

×

×

Counter

- word counter의 기능도 손쉽게 제공함

```
text = """A press release is the quickest and easiest way to get free publicity. If well written, a press release can result in multiple published articles about your firm and its products. And that can mean new prospects contacting you asking you to sell to them. …""".lower().split()

print(Counter(text))
print(Counter(text)["a"])
```

```
Counter({'a': 12, 'to': 10, 'the': 9, 'and': 9, 'release': 8, 'press': 8, 'that': 7, 'of': 5,
12
```

namedtuple

namedtuple

- Tuple 형태로 Data 구조체를 저장하는 방법
- 저장되는 data의 variable을 사전에 지정해서 저장함

```
from collections import namedtuple
Point = namedtuple('Point', ['x', 'y'])
p = Point(11, y=22)
print(p[0] + p[1])
```

```
x, y = p
print(x, y)
print(p.x + p.y)
print(Point(x=11, y=22))
```

namedtuple


```
from collections import namedtuple
import csv
f = open("users.csv", "r")
next(f)
reader = csv.reader(f)
student_list = []
for row in reader:
    student_list.append(row)
    print(row)


coloumns = ["user_id", "integration_id", "login_id", "password", "first_name",
            "last_name", "full_name", "sortable_name", "short_name",
            "email", "status"]
Student = namedtuple('Student', " ".join(coloumns))
student_namedtupe_list = []
for row in student_list:
    student = Student(*row)
    student_namedtupe_list.append(student)
print(student_namedtupe_list)
print(student_namedtupe_list[0].full_name)
```

End of Document
Thank You.