009

```
data=[('홍길동',23,'01099990001'),
      ('김철수',31,'01099990002'),
      ('이영희',29,'01099990003')
      ]
```

```
from collections import namedtuple
```

```
Employee = namedtuple('Employee','name,age,cellphone')

```

```
data=[Employee(emp[0], emp[1], emp[2]) for emp in data]
```

```
# [i*100 for i in[1,2,3]]
```

```
data
```

```
[Employee(name='홍길동', age=23, cellphone='01099990001'),
 Employee(name='김철수', age=31, cellphone='01099990002'),
 Employee(name='이영희', age=29, cellphone='01099990003')]
```

```
data2 = [Employee._make(emp) for emp in data]
```

```
data2
```

```
[Employee(name='홍길동', age=23, cellphone='01099990001'),
 Employee(name='김철수', age=31, cellphone='01099990002'),
 Employee(name='이영희', age=29, cellphone='01099990003')]
```

```
emp = data[0]
print(emp.name)
print(emp.age)
print(emp.cellphone)
```

```
홍길동
23
01099990001
```

```
emp[0]
```

```
'홍길동'
```

```
emp[1]
```

```
23
```

```
emp[2]
```

```
'01099990001'
```

```
emp._asdict() #튜플인데 as 마치 dict 딕셔너리처럼 만들어줘라~~
```

```
{'name': '홍길동', 'age': 23, 'cellphone': '01099990001'}
```

```
#emp.name='박길동' (x)
emp._replace(name="차희주")
  # 기존에 있던 튜플의 값에서, name에 대한 값만 바꾼 "새로운" 객체를 형성한 것임. 값이 바뀐게 아님!
```

```
Employee(name='차희주주', age=23, cellphone='01099990001')
```

```
data[0] = emp._replace(name='차희주')
data
```

```
[Employee(name='차희주', age=23, cellphone='01099990001'),
 Employee(name='김철수', age=31, cellphone='01099990002'),
 Employee(name='이영희', age=29, cellphone='01099990003')]
```

010

```
#from collections import Counter.py
from collections import Counter
import re


#따옴표 세개 입력하면 줄바꿈을 포함한 문자열을 입력할 수 있음.
#맨끝에 역슬래시 붙이면 값 달라지는 거 유의
data = """
산에는 꽃 피네.₩
꽃이 피네.₩
갈 봄 여름없이₩
꽃이 피네.₩
산에
산에
피는 꽃은
저만치 혼자서 피어있네.
산에서 우는 새여
꽃이 좋아
산에서
사노라네.
산에는 꽃지네
꽃이 지네.
갈 봄 여름 없이
꽃이 지네.
"""

```

```
data
```

```
' ₩n산에는 꽃 피네.꽃이 피네.갈 봄 여름없이꽃이 피네.산에₩n산에₩n피는 꽃은₩n저만치 혼자서 피어있네.₩n산에서 우는 새여₩n꽃이 좋아₩n산에서₩n사노라네.₩n산에는 꽃지네₩n꽃이 지네.₩n갈 봄 여름 없이₩n꽃이 지네.₩n'
```

```
from collections import Counter
import re

'''
words=re.findall('₩w{1}₩s', data) #w : [a-z A-Z 0-9_]'''

words=re.findall('₩w+', data) #w : [a-z A-Z 0-9_]
words
```

```
['산에는',
 '꽃',
 '피네',
 '꽃이',
 '피네',
 '갈',
 '봄',
 '여름없이꽃이',
 '피네',
 '산에',
 '산에',
 '피는',
 '꽃은',
 '저만치',
 '혼자서',
 '피어있네',
 '산에서',
 '우는',
 '새여',
 '꽃이',
 '좋아',
 '산에서',
 '사노라네',
 '산에는',
 '꽃지네',
 '꽃이',
 '지네',
 '갈',
 '봄',
 '여름',
 '없이',
 '꽃이',
 '지네']
```

```
counter = Counter(words)
counter
```

```
Counter({'산에는': 2,
         '꽃': 1,
         '피네': 3,
         '꽃이': 5,
         '갈': 2,
```

```
 '봄': 2,
 '여름없이': 1,
 '산에': 2,
 '피는': 1,
 '꽃은': 1,
 '저만치': 1,
 '혼자서': 1,
 '피어있네': 1,
 '산에서': 2,
 '우는': 1,
 '새여': 1,
 '좋아': 1,
 '사노라네': 1,
 '꽃지네': 1,
 '지네': 2,
 '여름': 1,
 '없이': 1})
```

```
1 words=re.findall(r'₩w+',data)
2 counter=Counter(words)
3 print(counter.most_common(1))
```

```
[('꽃이', 4)]
```

```
1 print(counter.most_common(2))
```

```
[('꽃이', 4), ('피네', 3)]
```

011

```
1 #nu (not use)
2
3
4 # d = defaultdict(int)
5 # for c in text:
6 #     d[c] += 1
7
8 # print(dict(d))
```

```
{'L': 1, 'i': 2, 'f': 1, 'e': 3, ' ': 6, 's': 2, 't': 3, 'o': 5, 'h': 2, 'r': 1, ',': 1, 'Y': 1, 'u': 1, 'n': 2, 'd': 1, 'p': 1, 'y': 1, '.': 1}
```

```
1 text = "Life is too short, You need python."
```

```
1 letters=re.findall('[a-zA-Z,.{1}]',text)
2 letters
```

```
['L',
 'i',
 'f',
 'e',
 'i',
 's',
 't',
 'o',
 'o',
 's',
 'h',
 'o',
 'r',
 't',
 ',',
 'Y',
 'o',
 'u',
 'n',
 'e',
 'e',
 'd',
 'p',
 'y',
 't',
 'h',
 'o',
 'n',
 '.']
```

```
1 counter=Counter(letters)
2 counter
```

```
Counter({'L': 1,
         'i': 2,
         'f': 1,
         'e': 3,
         's': 2,
         't': 3,
         'o': 5,
         'h': 2,
         'r': 1,
         ',': 1,
         'Y': 1,
         'u': 1,
         'n': 2,
         'd': 1,
         'p': 1,
         'y': 1,
         '.': 1})
```

```
text = "Life is too short, You need python."

d=dict()
for key in text:
  if key not in d: #key가 d에 없다면
    d[key]=0
  # d[key]=d[key]+1

d
```

```
{'L': 0,
 'i': 0,
 'f': 0,
 'e': 0,
 ' ': 0,
 's': 0,
 't': 0,
 'o': 0,
 'h': 0,
 'r': 0,
 ',': 0,
 'Y': 0,
 'u': 0,
 'n': 0,
 'd': 0,
 'p': 0,
 'y': 0,
 '.': 0}
```

```
from collections import defaultdict

text = "Life is too short, You need python."

d = defaultdict(int)
for key in text:
    d[key] += 1

print(dict(d))
```

```
{'L': 1, 'i': 2, 'f': 1, 'e': 3, ' ': 6, 's': 2, 't': 3, 'o': 5, 'h': 2, 'r': 1, ',': 1, 'Y': 1, 'u': 1, 'n': 2, 'd': 1, 'p': 1, 'y': 1, '.': 1}
```

012

```
# #heapq_sample.py (1)
# import heapq

# data = [
#     (12.23, "강보람"),
#     (12.31, "김지원"),
#     (11.98, "박시우"),
#     (11.99, "장준혁"),
#     (11.67, "차정웅"),
#     (12.02, "박중수"),
#     (11.57, "차동현"),
#     (12.04, "고미숙"),
#     (11.92, "한시우"),
#     (12.22, "이민석"),
# ]

# print(heapq.nsmallest(3, data))
```

```
[(11.57, '차동현'), (11.67, '차정웅'), (11.92, '한시우')]
```

```
data = [
    (12.23, "강보람"),
    (12.31, "김지원"),
    (11.98, "박시우"),
    (11.99, "장준혁"),
    (11.67, "차정웅"),
    (12.02, "박중수"),
    (11.57, "차동현"),
    (12.04, "고미숙"),
    (11.92, "한시우"),
    (12.22, "이민석"),
]
```

```
import heapq

h=[]
for score in data:
  heapq.heappush(h,score)

h
```

```
[(11.57, '차동현'),
 (11.92, '한시우'),
 (11.67, '차정웅'),
 (11.98, '박시우'),
 (11.99, '장준혁'),
 (12.23, '강보람'),
 (12.02, '박중수'),
 (12.31, '김지원'),
 (12.04, '고미숙'),
 (12.22, '이민석')]
```

```
heapq.heappop(h)
```

```
(11.67, '차정웅')
```

```
heapq.heappop(h)
```

```
(11.92, '한시우')
```

```
heapq.heappop(h)
```

```
(11.98, '박시우')
```

```
import heapq
h=[]
for score in data:
  heapq.heappush(h,score)

h
```

```
[(11.57, '차동현'),
 (11.92, '한시우'),
 (11.67, '차정웅'),
 (11.98, '박시우'),
 (11.99, '장준혁'),
 (12.23, '강보람'),
 (12.02, '박중수'),
 (12.31, '김지원'),
 (12.04, '고미숙'),
 (12.22, '이민석')]
```

```
#heapq_sample.py (2)
import heapq

h=[]

heapq.heapify(data)
data

# for i in range(3):
#   print(heapq.heappop(data)) #최소값부터 힙 반환

```

```
[(11.57, '차동현'),
 (11.67, '차정웅'),
 (11.98, '박시우'),
 (11.92, '한시우'),
 (12.22, '이민석'),
 (12.02, '박중수'),
```

```
    (12.23, '강보람'),
    (12.04, '고미숙'),
    (11.99, '장준혁'),
    (12.31, '김지원')]
```

```
#heapq_sample.py (3)
import heapq

data = [
    (12.23, "강보람"),
    (12.31, "김지원"),
    (11.98, "박시우"),
    (11.99, "장준혁"),
    (11.67, "차정웅"),
    (12.02, "박중수"),
    (11.57, "차동현"),
    (12.04, "고미숙"),
    (11.92, "한시우"),
    (12.22, "이민석"),
]

print(heapq.nsmallest(3,data))

```

```
[(11.57, '차동현'), (11.67, '차정웅'), (11.92, '한시우')]
```

013

```
import pprint
result={'userId':1,'id':1,'title':'sunt aun facere repella provident occaecatiexcepturi optio reprehenderit','body':'quia et suscipit₩msisco[ot rec
pprint.pprint(result)
```

```
{'body': 'quia et suscipit₩₩msisco[ot recusandae consequuntur expedita et cum₩n'
         'reprehenderit molestiae ut ut quas totam₩n'
         'nostrum rerum est autem sunt rem eveniet architexto',
 'id': 1,
 'title': 'sunt aun facere repella provident occaecatiexcepturi optio '
          'reprehenderit',
 'userId': 1}
```

014

```
# 일반적 파이썬
scores=[33, 99, 77, 70, 89, 90, 100]

result=[]
for score in scores:
  if score>=90:
    result.append('A')
  elif score>=80:
    result.append('B')
  elif score>=70:
    result.append('C')
  elif score>=60:
    result.append('D')
  else:
    result.append('F')

result
```

```
['F', 'A', 'C', 'C', 'B', 'A', 'A']
```

```
import bisect
scores=[33, 99, 77, 70, 89, 90, 100]


result=[]
for score in scores:
  pos=bisect.bisect([60,70,80,90],score)
  grade='FDCBA'[pos]
  result.append(grade)

print(result)
```

```
['F', 'A', 'C', 'C', 'B', 'A', 'A']
```

```
#bisect_sample.py (1)
import bisect

result = []
for score in [33, 99, 77, 70, 89, 90, 100]:
    pos = bisect.bisect_right([60, 70, 80, 90], score)  # 점수가 정렬되어 삽입될 수 있는 포지션을 반환
    grade = 'FDCBA'[pos]
    result.append(grade)

print(result)
```

```
['F', 'A', 'C', 'C', 'B', 'A', 'A']
```

```
#bisect_sample.py (2)
import bisect

result = []
for score in [33, 99, 77, 70, 89, 90, 100]:
    pos = bisect.bisect_left([60, 70, 80, 90], score)  # 점수가 정렬되어 삽입될 수 있는 포지션을 반환
    grade = 'FDCBA'[pos]
    result.append(grade)

print(result)
```

```
['F', 'A', 'C', 'D', 'B', 'B', 'A']
```

```
import bisect
a=[60,70,80,90]
bisect.insort(a,85) #insort도 bisect처럼 left와 right가 존재함.
a
```

```
[60, 70, 80, 85, 90]
```

015

```
from datetime import date

def get_menu(input_date):
  weekday=input_date.isoweekday()
  if weekday == 1:
    menu = "김치찌개"
  elif weekday == 2:
    menu ="비빔밥"
  elif weekday == 3:
    menu = "된장찌개"
  elif weekday == 4:
    menu = "불고기"
  elif weekday == 5:
    menu = "갈비탕"
  elif weekday == 6:
    menu = "라면"
  elif weekday == 7:
    menu = "건빵"
  return menu

print(get_menu(date (2021, 12, 6)))
print(get_menu(date (2021, 12, 18)))

```

```
김치찌개
라면면
```

```
#enum_sample.py
from datetime import date
from enum import IntEnum

#enum.IntEnum클래스를 상속하여 새로운 enum.Enum클래스(Enum) 구현

class Week(IntEnum):
    MONDAY = 1
    TUESDAY = 2
    WEDNESDAY = 3
    THURSDAY = 4
    FRIDAY = 5
    SATURDAY = 6
    SUNDAY = 7


```

```
def get_menu(input_date):
    menu = {
        Week.MONDAY: "김치찌개",
        Week.TUESDAY: "비빔밥",
        Week.WEDNESDAY: "된장찌개",
        Week.THURSDAY: "불고기",
        Week.FRIDAY: "갈비탕",
        Week.SATURDAY: "라면",
        Week.SUNDAY: "건빵",
    }
    return menu[input_date.isoweekday()]

#류타이쿠진_푸팟퐁커리 맛잇겟다

print(get_menu(date(2020, 12, 6)))
print(get_menu(date(2020, 12, 18)))
```

```
건빵
갈비탕
```

016

```
#topologicalsorter_sample.py
from graphlib import TopologicalSorter

# tree is graph's special 모양.


ts = TopologicalSorter()

# 규칙1
ts.add('영어중급', '영어초급')  # 영어중급의 선수과목은 영어초급
ts.add('영어고급', '영어중급')  # 영어고급의 선수과목은 영어중급

# 규칙2
ts.add('영어문법', '영어중급')  # 영어문법의 선수과목은 영어중급
ts.add('영어고급','영어 중급', '영어문법')  # 영어고급의 선수과목은 영어문법

# 규칙3
ts.add('영어회화', '영어문법')  # 영어회화의 선수과목은 영어문법

print(list(ts.static_order()))  # 위상정렬한 결과를 출력

```

```
['영어초급', '영어 중급', '영어중급', '영어문법', '영어고급', '영어회화']
```

+ 코드 + 텍스트