025

```
data =[
    {'name': '이민서','blood':'O'},
    {'name': '이영순','blood':'B'},
    {'name': '이상호','blood':'AB'},
    {'name': '김지민','blood':'B'},
    {'name': '최상현','blood':'AB'},
    {'name': '김지아','blood':'A'},
    {'name': '손우진','blood':'A'},
    {'name': '박은주','blood':'A'},
]


# import itertools
# data=itertools.groupby(data)

# import pprint
# pprint.pprint(list(data))
```

```
import itertools
data=itertools.groupby(data)

import pprint
pprint.pprint(list(data))
```

```
[({'blood': 'O', 'name': '이민서'}, <itertools._grouper object at 0x7f4083f93af0>),
 ({'blood': 'B', 'name': '이영순'}, <itertools._grouper object at 0x7f4083f93820>),
 ({'blood': 'AB', 'name': '이상호'},
  <itertools._grouper object at 0x7f4083f93b50>),
 ({'blood': 'B', 'name': '김지민'}, <itertools._grouper object at 0x7f4083f937c0>),
 ({'blood': 'AB', 'name': '최상현'},
  <itertools._grouper object at 0x7f4083f93c10>),
 ({'blood': 'A', 'name': '김지아'}, <itertools._grouper object at 0x7f4083f93520>),
 ({'blood': 'A', 'name': '손우진'}, <itertools._grouper object at 0x7f4083f93940>),
 ({'blood': 'A', 'name': '박은주'}, <itertools._grouper object at 0x7f4083f93460>)]
```

```
# import operator
# data=sorted(data,key=operator.itemgetter('blood'))
```

```
# data
```

```
[{'name': '김지아', 'blood': 'A'},
 {'name': '손우진', 'blood': 'A'},
 {'name': '박은주', 'blood': 'A'},
 {'name': '이상호', 'blood': 'AB'},
 {'name': '최상현', 'blood': 'AB'},
 {'name': '이영순', 'blood': 'B'},
 {'name': '김지민', 'blood': 'B'},
 {'name': '이민서', 'blood': 'O'}]
```

```
import itertools

grouped_data=itertools.groupby(data, key=operator.itemgetter('blood'))
```

```
list(grouped_data)
```

```
[('O', <itertools._grouper at 0x7f4083fc7190>),
 ('B', <itertools._grouper at 0x7f4083fc71f0>),
 ('AB', <itertools._grouper at 0x7f4083fc7dc0>),
 ('B', <itertools._grouper at 0x7f4083fc7d00>),
 ('AB', <itertools._grouper at 0x7f4083fc7130>),
 ('A', <itertools._grouper at 0x7f4083fc7160>)]
```

```
data =[
    {'name': '이민서','blood':'O'},
    {'name': '이영순','blood':'B'},
    {'name': '이상호','blood':'AB'},
    {'name': '김지민','blood':'B'},
    {'name': '최상현','blood':'AB'},
    {'name': '김지아','blood':'A'},
    {'name': '손우진','blood':'A'},
    {'name': '박은주','blood':'A'},
]

import itertools

```

```
grouped_data=itertools.groupby(data, key=operator.itemgetter('blood'))

import pprint

result={}
for key,group_data in grouped_data:
  result[key]=list(group_data)

pprint.pprint(result)
```

```
{'A': [{'blood': 'A', 'name': '김지아'},
       {'blood': 'A', 'name': '손우진'},
       {'blood': 'A', 'name': '박은주'}],
 'AB': [{'blood': 'AB', 'name': '최상현'}],
 'B': [{'blood': 'B', 'name': '김지민'}],
 'O': [{'blood': 'O', 'name': '이민서'}]}
```

026

```
students = ['한민서','황지민','이영철','이광수','김승민']
rewards=['사탕','초콜릿','젤리']

result=zip(students,rewards)
print(list(result))
```

```
[('한민서', '사탕'), ('황지민', '초콜릿'), ('이영철', '젤리')]
```

```
import itertools

students = ['한민서', '황지민', '이영철', '이광수', '김승민']
rewards = ['사탕', '초콜릿', '젤리']

result = itertools.zip_longest(students, rewards)
print(list(result))
```

```
[('한민서', '사탕'), ('황지민', '초콜릿'), ('이영철', '젤리'), ('이광수', None), ('김승민', None)]
```

```
import itertools

students = ['한민서', '황지민', '이영철', '이광수', '김승민']
rewards = ['사탕', '초콜릿', '젤리']

result = itertools.zip_longest(students, rewards, fillvalue='새우깡')
print(list(result))
```

```
[('한민서', '사탕'), ('황지민', '초콜릿'), ('이영철', '젤리'), ('이광수', '새우깡'), ('김승민', '새우깡')]
```

027

```
import itertools

list(itertools.permutations(['1','2','3'],r=3))
```

```
[('1', '2', '3'),
 ('1', '3', '2'),
 ('2', '1', '3'),
 ('2', '3', '1'),
 ('3', '1', '2'),
 ('3', '2', '1')]
```

```
for n1,n2 in itertools.permutations(['1','2','3'],r=2):
  print(n1+n2)

```

```
12
13
21
23
31
32
```

```
list(itertools.combinations(['1','2','3'],r=2))
```

```
[('1', '2'), ('1', '3'), ('2', '3')]
```

```
list(itertools.combinations_with_replacement(['1','2','3'],r=2))
```

```
[('1', '1'), ('1', '2'), ('1', '3'), ('2', '2'), ('2', '3'), ('3', '3')]
```

```
1
```

028

```
1 import itertools
2 list(itertools.combinations(range(1,46),6))
```

```
(1, 2, 3, 4, 9, 14),
(1, 2, 3, 4, 9, 15),
(1, 2, 3, 4, 9, 16),
(1, 2, 3, 4, 9, 17),
(1, 2, 3, 4, 9, 18),
(1, 2, 3, 4, 9, 19),
(1, 2, 3, 4, 9, 20),
(1, 2, 3, 4, 9, 21),
(1, 2, 3, 4, 9, 22),
(1, 2, 3, 4, 9, 23),
(1, 2, 3, 4, 9, 24),
(1, 2, 3, 4, 9, 25),
(1, 2, 3, 4, 9, 26),
(1, 2, 3, 4, 9, 27),
(1, 2, 3, 4, 9, 28),
(1, 2, 3, 4, 9, 29),
(1, 2, 3, 4, 9, 30),
(1, 2, 3, 4, 9, 31),
(1, 2, 3, 4, 9, 32),
(1, 2, 3, 4, 9, 33),
(1, 2, 3, 4, 9, 34),
(1, 2, 3, 4, 9, 35),
(1, 2, 3, 4, 9, 36),
(1, 2, 3, 4, 9, 37),
(1, 2, 3, 4, 9, 38),
(1, 2, 3, 4, 9, 39),
(1, 2, 3, 4, 9, 40),
(1, 2, 3, 4, 9, 41),
(1, 2, 3, 4, 9, 42),
(1, 2, 3, 4, 9, 43),
(1, 2, 3, 4, 9, 44),
(1, 2, 3, 4, 9, 45),
(1, 2, 3, 4, 10, 11),
(1, 2, 3, 4, 10, 12),
(1, 2, 3, 4, 10, 13),
(1, 2, 3, 4, 10, 14),
(1, 2, 3, 4, 10, 15),
(1, 2, 3, 4, 10, 16),
(1, 2, 3, 4, 10, 17),
(1, 2, 3, 4, 10, 18),
(1, 2, 3, 4, 10, 19),
(1, 2, 3, 4, 10, 20),
(1, 2, 3, 4, 10, 21),
(1, 2, 3, 4, 10, 22),
(1, 2, 3, 4, 10, 23),
(1, 2, 3, 4, 10, 24),
(1, 2, 3, 4, 10, 25),
(1, 2, 3, 4, 10, 26),
(1, 2, 3, 4, 10, 27),
(1, 2, 3, 4, 10, 28),
(1, 2, 3, 4, 10, 29),
(1, 2, 3, 4, 10, 30),
(1, 2, 3, 4, 10, 31),
(1, 2, 3, 4, 10, 32),
(1, 2, 3, 4, 10, 33),
(1, 2, 3, 4, 10, 34),
(1, 2, 3, 4, 10, 35),
(1, 2, 3, 4, 10, 36),
(1, 2, 3, 4, 10, 37)
```

```
1 len(list(itertools.combinations(range(1,46),r=6)))
```

```
8145060
```

029

```
1 import functools
2
3 def xy_compare(n1, n2):
4     if n1[1] > n2[1]:         # y 좌표가 크면
5         return 1
6     elif n1[1] == n2[1]:      # y 좌표가 같으면
7         if n1[0] > n2[0]:     # x 좌표가 크면
```

```
            return 1
        elif n1[0] == n2[0]:  # x 좌표가 같으면
            return 0
        else:                 # x 좌표가 작으면
            return -1
    else:                     # y 좌표가 작으면
        return -1

src = [(0, 4), (1, 2), (1, -1), (2, 2), (3, 3)]
result = sorted(src, key=functools.cmp_to_key(xy_compare))
print(result)
```

```
[(1, -1), (1, 2), (2, 2), (3, 3), (0, 4)]
```

```

```

030

```
import urllib.request

def get_wikidocs(page):
  print("wikidocs page: {}".format(page))
  resource='https://wikidocs.net/{}'.format(page)
  try:
    with urllib.request.urlopen(resource) as s:
      return s.read()
  except urllib.error.HTTPError:
      return 'Not Found'

```

```
print(get_wikidocs(1))
```

```
wikidocs page: 1
b'₩n<!DOCTYPE HTML>₩n<html lang="ko">₩n<head>₩n    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">₩n    <meta name="viewport
```

```
import urllib.request
from functools import lru_cache

@lru_cache(maxsize=32)
def get_wikidocs(page):
  print("wikidocs page: {}".format(page))
  resource='https://wikidocs.net/{}'.format(page)
  try:
    with urllib.request.urlopen(resource) as s:
      return s.read()
  except urllib.error.HTTPError:
      return 'Not Found'

```

```
first6=get_wikidocs(6)
```

```
wikidocs page: 6
```

```
pprint.pprint(first6)
```

```
b'₩x84₩xec₩x9d₩xb4 ₩xec₩x83₩x9d₩xea₩xb0₩x81₩xed₩x95₩x98₩xeb₩x8a₩x94 ₩xea₩xb2'
b'₩x83₩xec₩x9d₩x84 ₩xec₩xbb₩xb4₩xed₩x93₩xa8₩xed₩x84₩xb0₩xec₩x97₩x90 ₩xec₩xa7'
b'₩x80₩xec₩x8b₩x9c₩xed₩x95₩x98₩xeb₩x8a₩x94 ₩xed₩x96₩x89₩xec₩x9c'
b'₩x84₩xeb₩x9d₩xbc₩xea₩xb3₩xa0 ₩xed₩x95₩xa0 ₩xec₩x88₩x98 ₩xec₩x9e₩x88₩xeb'
b'₩x8b₩xa4. ₩xec₩x95₩x9e₩xec₩x9c₩xbc₩xeb₩xa1₩x9c ₩xec₩x82₩xb4₩xed₩x8e₩xb4'
b'₩xeb₩xb3₩xbc ₩xed₩x8c₩x8c₩xec₩x9d₩xb4₩xec₩x8d₩xac ₩xeb₩xac₩xb8₩xeb₩xb2₩x95'
b'₩xec₩x97₩x90₩xec₩x84₩x9c₩xeb₩x8f₩x84 ₩xeb₩xb3₩xb4₩xea₩xb2₩x8c ₩xeb₩x90₩x98'
b'₩xea₩xb2₩xa0₩xec₩xa7₩x80₩xeb₩xa7₩x8c, ₩xed₩x8c₩x8c₩xec₩x9d₩xb4₩xec₩x8d₩xac'
b'₩xec₩x9d₩x80 ₩xec₩x82₩xac₩xeb₩x9e₩x8c₩xec₩x9d₩xb4 ₩xec₩x83₩x9d₩xea₩xb0₩x81'
b'₩xed₩x95₩x98₩xeb₩x8a₩x94 ₩xeb₩xb0₩xa9₩xec₩x8b₩x9d₩xec₩x9d₩x84 ₩xea₩xb7₩xb8'
b'₩xeb₩x8c₩x80₩xeb₩xa1₩x9c ₩xed₩x91₩x9c₩xed₩x98₩x84₩xed₩x95₩xa0 ₩xec₩x88₩x98'
b' ₩xec₩x9e₩x88₩xeb₩x8a₩x94 ₩xec₩x96₩xb8₩xec₩x96₩xb4₩xec₩x9d₩xb4₩xeb₩x8b₩xa4'
b'. ₩xeb₩x94₩xb0₩xeb₩x9d₩xbc₩xec₩x84₩x9c ₩xed₩x94₩x84₩xeb₩xa1₩x9c₩xea₩xb7'
b'₩xb8₩xeb₩x9e₩x98₩xeb₩xa8₩xb8₩xeb₩x8a₩x94 ₩xea₩xb5₩xb3₩xec₩x9d₩xb4 ₩xec₩xbb'
b'₩xb4₩xed₩x93₩xa8₩xed₩x84₩xb0₩xec₩x9d₩x98 ₩xec₩x82₩xac₩xea₩xb3₩xa0 ₩xec₩xb2'
b'₩xb4₩xea₩xb3₩x84₩xec₩x97₩x90 ₩xeb₩xa7₩x9e₩xec₩xb6₩x94₩xec₩x96'
b'₩xb4₩xec₩x84₩x9c ₩xed₩x94₩x84₩xeb₩xa1₩x9c₩xea₩xb7₩xb8₩xeb₩x9e'
b'₩x98₩xeb₩xb0₩x8d₩xec₩x9d₩x84 ₩xed₩x95₩x98₩xeb₩xa0₩xa4₩xea₩xb3₩xa0 ₩xec₩x95'
b'₩xa0₩xec₩x93₩xb8 ₩xed₩x95₩x84₩xec₩x9a₩x94₩xea₩xb0₩x80 ₩xec₩x97₩x86₩xeb₩x8b'
b'₩xa4. ₩xec₩x9d₩xb4₩xec₩xa0₩x9c ₩xea₩xb3₩xa7 ₩xec₩x96₩xb4₩xeb₩x96₩xa4'
b' ₩xed₩x94₩x84₩xeb₩xa1₩x9c₩xea₩xb7₩xb8₩xeb₩x9e₩xa8₩xec₩x9d₩x84 ₩xea₩xb5₩xac'
b'₩xec₩x83₩x81₩xed₩x95₩x98₩xec₩x9e₩x90₩xeb₩xa7₩x88₩xec₩x9e₩x90 '
b'₩xeb₩xa8₩xb8₩xeb₩xa6₩xbf₩xec₩x86₩x8d₩xec₩x97₩x90₩xec₩x84₩x9c '
b'₩xec₩x83₩x9d₩xea₩xb0₩x81₩xed₩x95₩x9c ₩xeb₩x8c₩x80₩xeb₩xa1₩x9c ₩xec₩x88₩xa0'
b'₩xec₩x88₩xa0 ₩xec₩x8d₩xa8 ₩xeb₩x82₩xb4₩xeb₩xa0₩xa4₩xea₩xb0₩x80₩xeb₩x8a₩x94'
b' ₩xec₩x97₩xac₩xeb₩x9f₩xac₩xeb₩xb6₩x84₩xec₩x9d₩x98 ₩xeb₩xaa₩xa8₩xec₩x8a₩xb5'
b'₩xec₩x97₩x90 ₩xeb₩x86₩x80₩xeb₩x9d₩xbc₩xea₩xb2₩x8c ₩xeb₩x90₩xa0 ₩xea₩xb2'
b'₩x83₩xec₩x9d₩xb4₩xeb₩x8b₩xa4.</p>₩n<p>₩xeb₩x8b₩xa4₩xec₩x9d₩x8c ₩xec'
b'₩x86₩x8c₩xec₩x8a₩xa4 ₩xec₩xbd₩x94₩xeb₩x93₩x9c₩xeb₩xa5₩xbc ₩xeb₩xb3₩xb4₩xeb'
b'₩xa9₩xb4 ₩xec₩x9d₩xb4 ₩xeb₩xa7₩x90₩xec₩x9d₩xb4 ₩xec₩x89₩xbd₩xea₩xb2₩x8c'
b' ₩xec₩x9d₩xb4₩xed₩x95₩xb4₩xeb₩x90₩xa0 ₩xea₩xb2₩x83₩xec₩x9d₩xb4₩xeb₩x8b₩xa4'
b'.</p>₩n<pre><code class="language-python">if 4 in [1,2,3,4]: print(&quot;'
b'4₩xea₩xb0₩x80 ₩xec₩x9e₩x88₩xec₩x8a₩xb5₩xeb₩x8b₩x88₩xeb₩x8b₩xa4&quot;)₩n</c'
b'ode></pre>₩n<p>₩xec₩x9c₩x84 ₩xec₩x98₩x88₩xec₩xa0₩x9c₩xeb₩x8a₩x94 '
b'₩xeb₩x8b₩xa4₩xec₩x9d₩x8c₩xec₩xb2₩x98₩xeb₩x9f₩xbc ₩xec₩x9d₩xbd₩xec₩x9d₩x84 '
```

```
second6 = get_wikidocs(6)
```

```
third6=get_wikidocs(6)
```

```
assert first6==second6   # at internet, <assert python> 검색해보기, #assert는 true면 아무값도 반환x 근데 false일때 assertionError가 발생.
```

```
import urllib.request
from functools import lru_cache


@lru_cache(maxsize=32)
def get_wikidocs(page):
    print("wikidocs page:{}".format(page))  # 페이지 호출시 출력
    resource = 'https://wikidocs.net/{}'.format(page)
    try:
        with urllib.request.urlopen(resource) as s:
            return s.read()
    except urllib.error.HTTPError:
        return 'Not Found'


first_6 = get_wikidocs(6)
first_7 = get_wikidocs(7)

second_6 = get_wikidocs(6)
second_7 = get_wikidocs(7)

assert first_6 == second_6  # 처음 요청한 6번 페이지의 내용과 두번째 요청한 6번 페이지의 내용이 같은지를 확인
assert first_7 == second_7
```

```
wikidocs page:6
wikidocs page:7
```

031

```
def add_mul(choice,*args):
  if choice == "add":
    result=0
    for arg in args:
      result+=arg
  elif choice == "mul":
    result=1
    for arg in args:
      result*=arg
  return result
```

```
add_mul('add',1,2,3,4,5)
```

```
15
```

```
def add_mul(choice,*args):
  if choice == "add":
    result=0
    for arg in args:
      result+=arg
  elif choice == "mul":
    result=1
    for arg in args:
      result*=arg
  return result

def add(*args):
  return add_mul('add',*args)

def mul(*args):
  return add_mul('mul',*args)

print(add(1,2,3,4,5))
print(mul(1,2,3,4,5))
```

```
15
120
```

```
from functools import partial


# def add_mul(choice, *args):
#     if choice == "add":
#         result = 0
#         for i in args:
#             result = result + i
#     elif choice == "mul":
#         result = 1
#         for i in args:
#             result = result * i
#     return result


add = partial(add_mul, 'add')
mul = partial(add_mul, 'mul')

# print(add(1,2,3,4,5))  # 15 출력
# print(mul(1,2,3,4,5))  # 120 출력
```

```
add(1,2,3,4,5)
```

```
15
```

```
add1000=partial(add_mul,'add',1000) #새로운 객체를 만들어내는 partial함수
add1000(1,2,3,4,5) #add_mul('add',1000,1,2,3,4,5)
```

```
1015
```

```
add1000.func
```

```
<function __main__.add_mul(choice, *args)>
```

```
add1000.args
```

```
('add', 1000)
```

032

```
numbers = input("n개의 수를 입력하세요 > ")
numbers= numbers.split(' ')
n=[]
for i in numbers:
  n.append(int(i))

```

```
n개의 수를 입력하세요 > 1 2 3 4 5
['1', '2', '3', '4', '5']
```

```
# def get_num():
#   numbers = input("n개의 수를 입력하세요 > ")
#   numbers= numbers.split(' ')
#   n=[]
#   for i in numbers:
#     n.append(int(i))
#   return n
```

```
def add(data):
  result=0
  for i in data:
    result+=i
  return result

data=[1,2,3,4,5]
result = add(data)
print(result)

```

```
15
```

```
import functools

result = functools.reduce(lambda x, y: x + y, get_num())
print(result)
```

```
n개의 수를 입력하세요 > 1 2 3 4 5
15
```

```
functools.reduce(lambda x, y: x if x>y else y, get_num())
```

```
n개의 수를 입력하세요 > 1 2 12 2 8
12
```

```
functools.reduce(lambda x, y: x if x<y else y, get_num())
```

```
n개의 수를 입력하세요 > -90 83 12 0 -128
-128
```

```
1
```

033

```
import time

def elapsed(original_func):
  def wrapper(*args,**kwargs):
    s=time.time()
    rst=original_func(*args,**kwargs)
    e=time.time()
    print('elapsed itme: %f 초' %(e-s))
    return result
  return wrapper

@elapsed
def add(a,b):
  return a+b

@elapsed
def factorial(n):
  if n==1:
    return 1
  else:
    return n*factorial(n-1)

```

```
23
24 print(add(3,4))
25 print(factorial(10))
```

```
elapsed itme: 0.000001 초
15
elapsed itme: 0.000001 초
elapsed itme: 0.000504 초
elapsed itme: 0.000903 초
elapsed itme: 0.001332 초
elapsed itme: 0.002188 초
elapsed itme: 0.003058 초
elapsed itme: 0.003540 초
elapsed itme: 0.003981 초
elapsed itme: 0.004379 초
elapsed itme: 0.004417 초
15
```

```
1 import time
2 import functools
3
4 def elapsed(original_func):
5   @functools.wraps(original_func)
6   def wrapper(*args,**kwargs):
7     s=time.time()
8     rst=original_func(*args,**kwargs)
9     e=time.time()
10    print('elapsed itme: %f 초' %(e-s))
11    return result
12  return wrapper
13
14 @elapsed
15 def add(a,b):
16   return a+b
17 '''두 수 a,b의 덧셈 결과를 반환하는 함수'''
18
19 print(add(3,4))
```

```
elapsed itme: 0.000001 초
15
```

```
1 print(add)
```

```
<function elapsed.<locals>.wrapper at 0x7f4083f92c10>
```

```
1 help(add)
```

```
Help on function wrapper in module __main__:

wrapper(*args, **kwargs)
```

034

```
1 # from operator import itemgetter
2
3 # students = [
4 #     ("jane", 22, 'A'),
5 #     ("dave", 32, 'B'),
6 #     ("sally", 17, 'B'),
7 # ]
8
9 # result = sorted(students, key=itemgetter(1))
10 # print(result)
```

```
[('sally', 17, 'B'), ('jane', 22, 'A'), ('dave', 32, 'B')]
```

```
1 students = [
2     ("jane", 22, 'A'),
3     ("dave", 32, 'B'),
4     ("sally", 17, 'B'),
5 ]
```

```
1 from operator import itemgetter
2
3 sorted(students, key=operator.itemgetter(1))
```

```
[('sally', 17, 'B'), ('jane', 22, 'A'), ('dave', 32, 'B')]
```

```
sorted(students, key=operator.itemgetter(0))
```

```
[('dave', 32, 'B'), ('jane', 22, 'A'), ('sally', 17, 'B')]
```

```
sorted(students, key=operator.itemgetter(2))
```

```
[('jane', 22, 'A'), ('dave', 32, 'B'), ('sally', 17, 'B')]
```

```
students = [
    {'name':'jane','age':22,'blood':'A'},
    {'name':'dave','age':32,'blood':'B'},
    {'name':'sally','age':17,'blood':'B'}
]
```

```
sorted(students, key=operator.itemgetter('age'))
```

```
[{'name': 'sally', 'age': 17, 'blood': 'B'},
 {'name': 'jane', 'age': 22, 'blood': 'A'},
 {'name': 'dave', 'age': 32, 'blood': 'B'}]
```

```
sorted(students, key=operator.itemgetter('name'))
```

```
[{'name': 'dave', 'age': 32, 'blood': 'B'},
 {'name': 'jane', 'age': 22, 'blood': 'A'},
 {'name': 'sally', 'age': 17, 'blood': 'B'}]
```

```
sorted(students, key=operator.itemgetter('blood'))
```

```
[{'name': 'jane', 'age': 22, 'blood': 'A'},
 {'name': 'dave', 'age': 32, 'blood': 'B'},
 {'name': 'sally', 'age': 17, 'blood': 'B'}]
```

```
sorted(students, key=operator.itemgetter('blood','age'))
```

```
[{'name': 'jane', 'age': 22, 'blood': 'A'},
 {'name': 'sally', 'age': 17, 'blood': 'B'},
 {'name': 'dave', 'age': 32, 'blood': 'B'}]
```

```
class Student:
  def __init__(self,name,age,blood):
    self.name=name
    self.age=age
    self.blood=blood

students=[
    Student('jane',22,'A'),
    Student('dave',32,'B'),
    Student('sally',17,'B'),
]

print(students)
print(sorted(students,key=operator.attrgetter('age')))
```

```
[<__main__.Student object at 0x7f4083f938b0>, <__main__.Student object at 0x7f4083f936d0>, <__main__.Student object at 0x7f4083f93f10>]
[<__main__.Student object at 0x7f4083f93f10>, <__main__.Student object at 0x7f4083f938b0>, <__main__.Student object at 0x7f4083f936d0>]
```

```
1
```