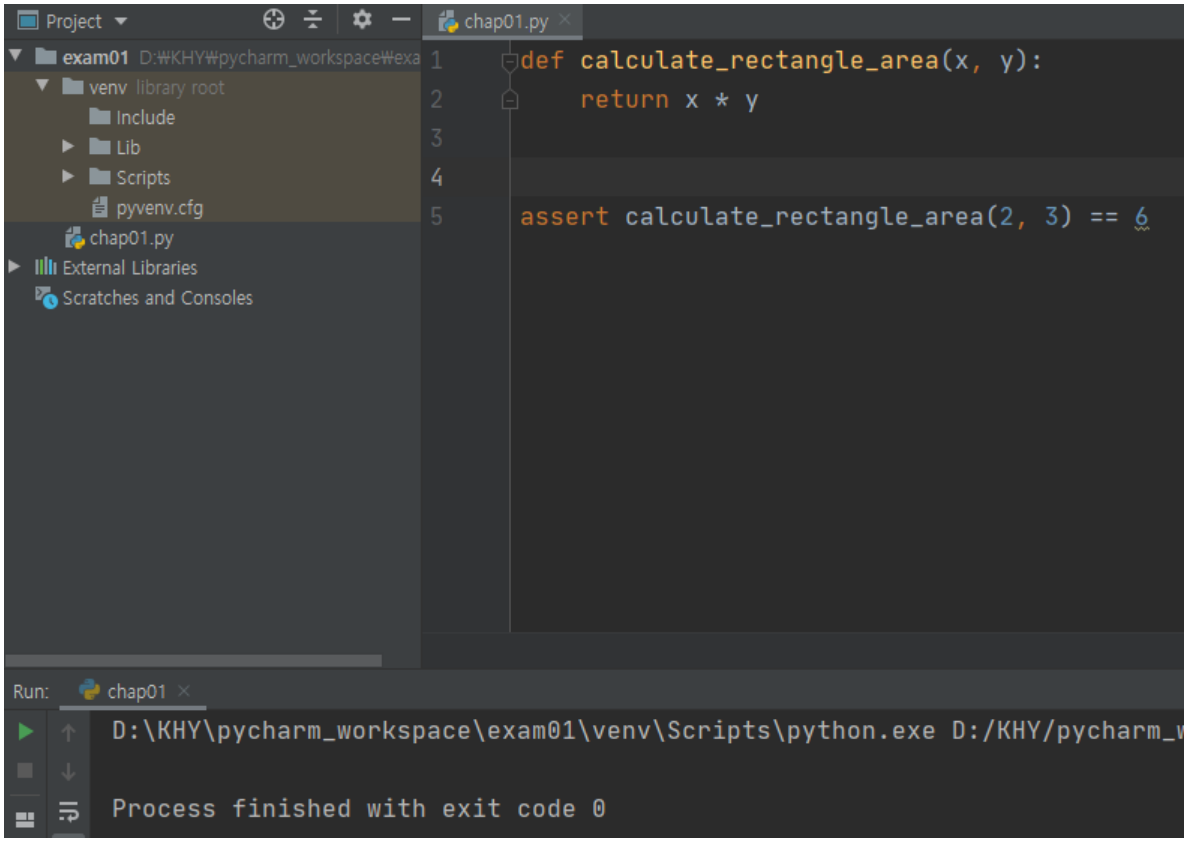
파이썬에서의 함수

- def

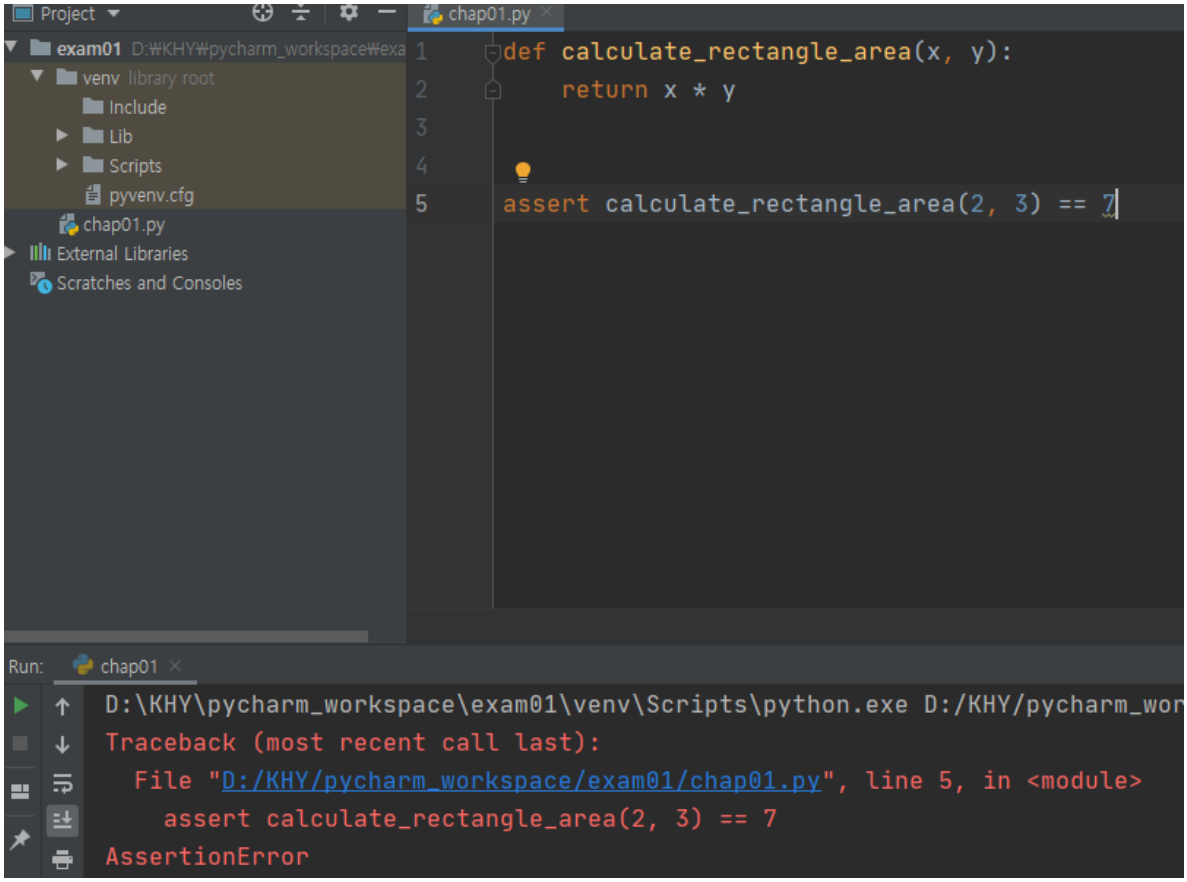
```
def 함수이름 (인자1, 인자2, ...옵션)
    실행문1
    실행문2
    return 반환값(옵션)
```

> 리턴타입을 명시해주지 않음

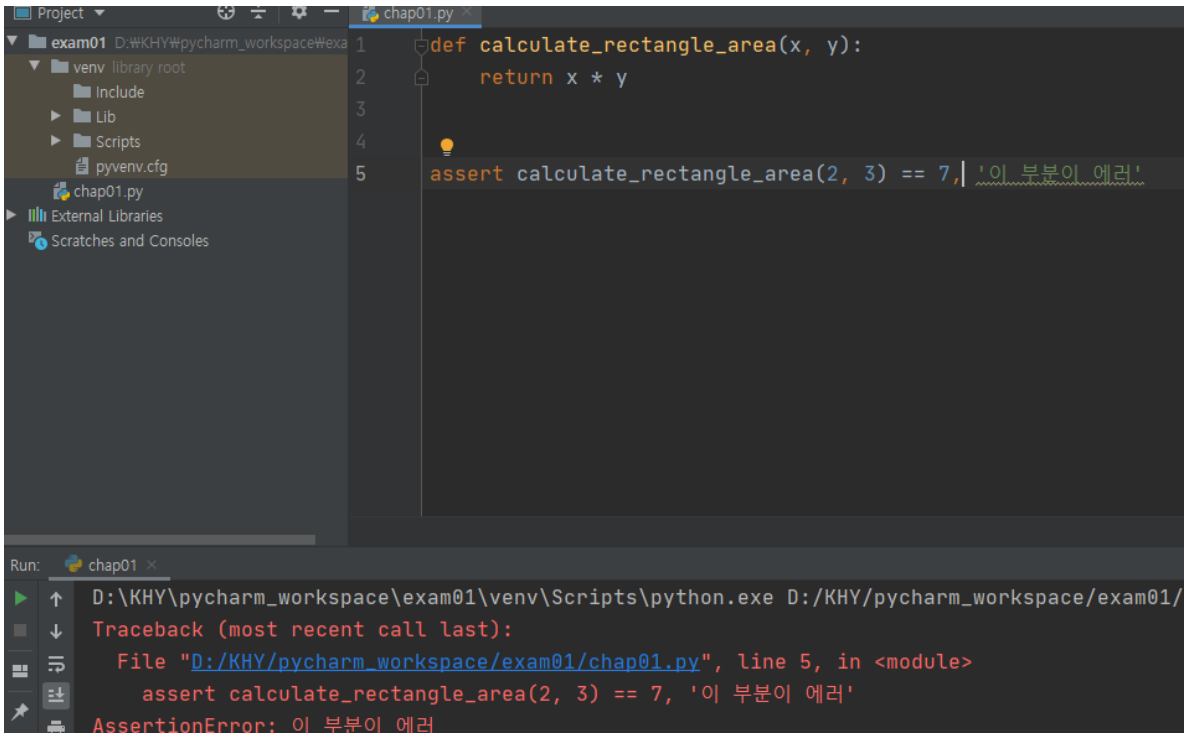


> 함수 정의 후 실행문 쓸 땐 2줄 이상 띄우는 것이 좋음(가독성)

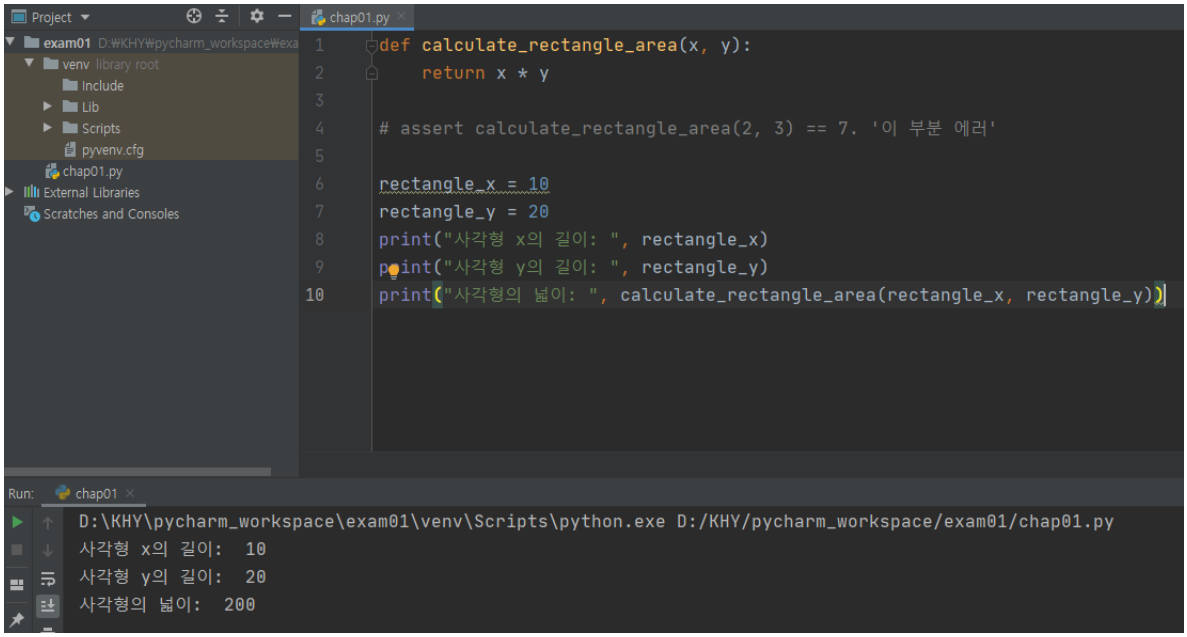
> assert문(가정 설정문) : 뒤의 조건이 True가 아니면 AssertError 발생 시킴(단위 테스트)



실행문이 틀렸을 경우

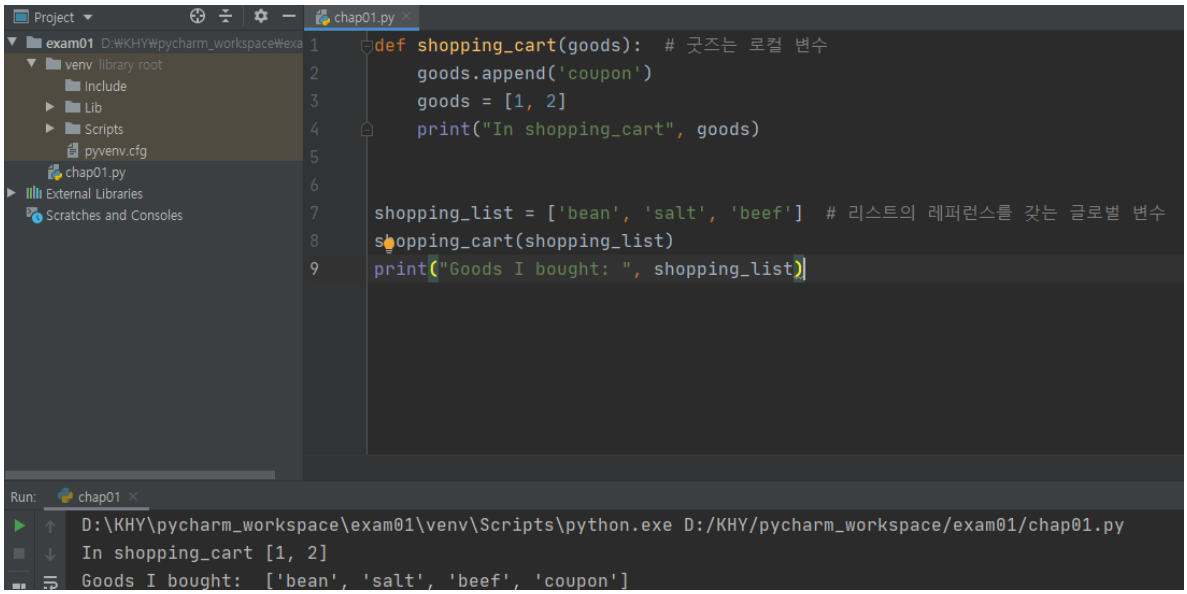


arrest문에 위 처럼 작성하면 에러를 좀 더 쉽게 찾을 수 있음



> 실행결과

- 변수의 scope 영역

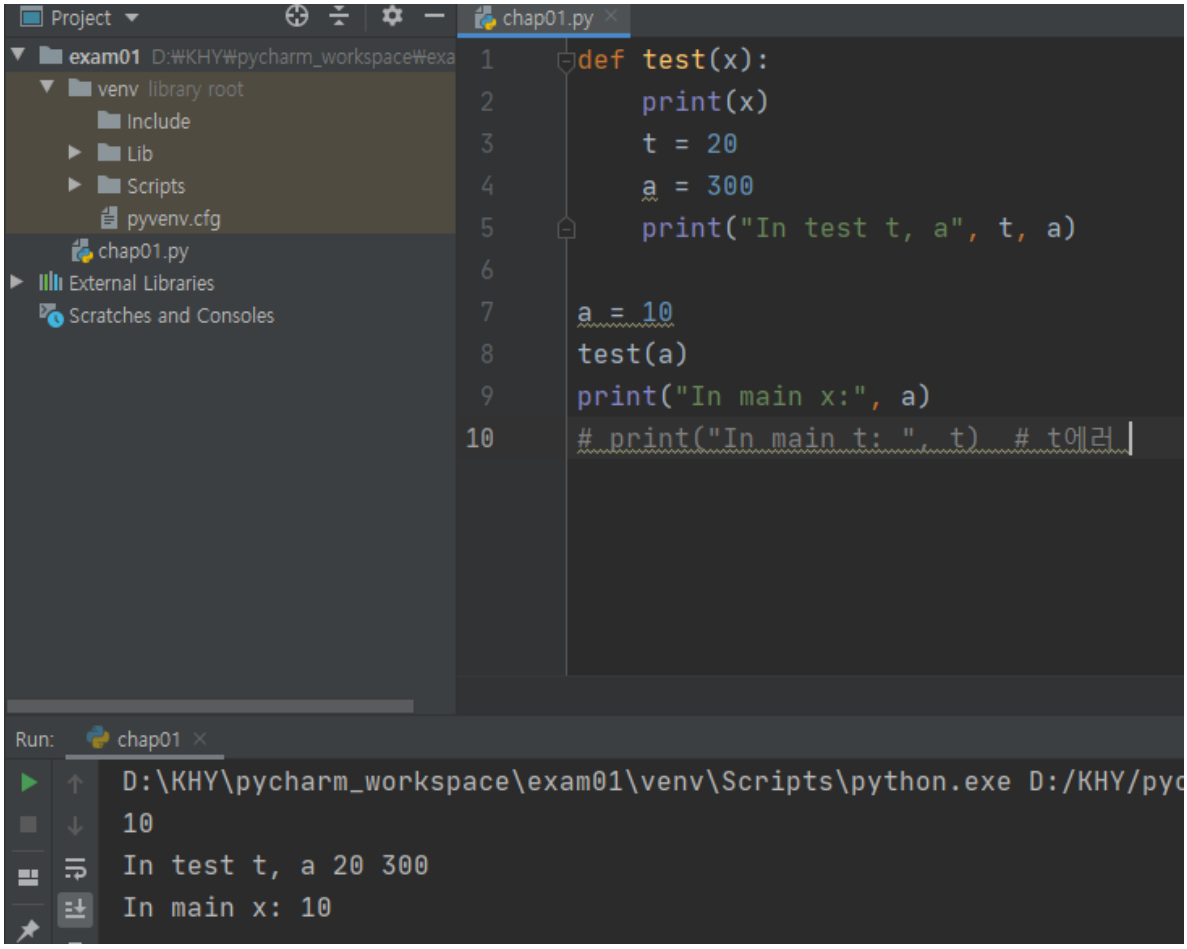


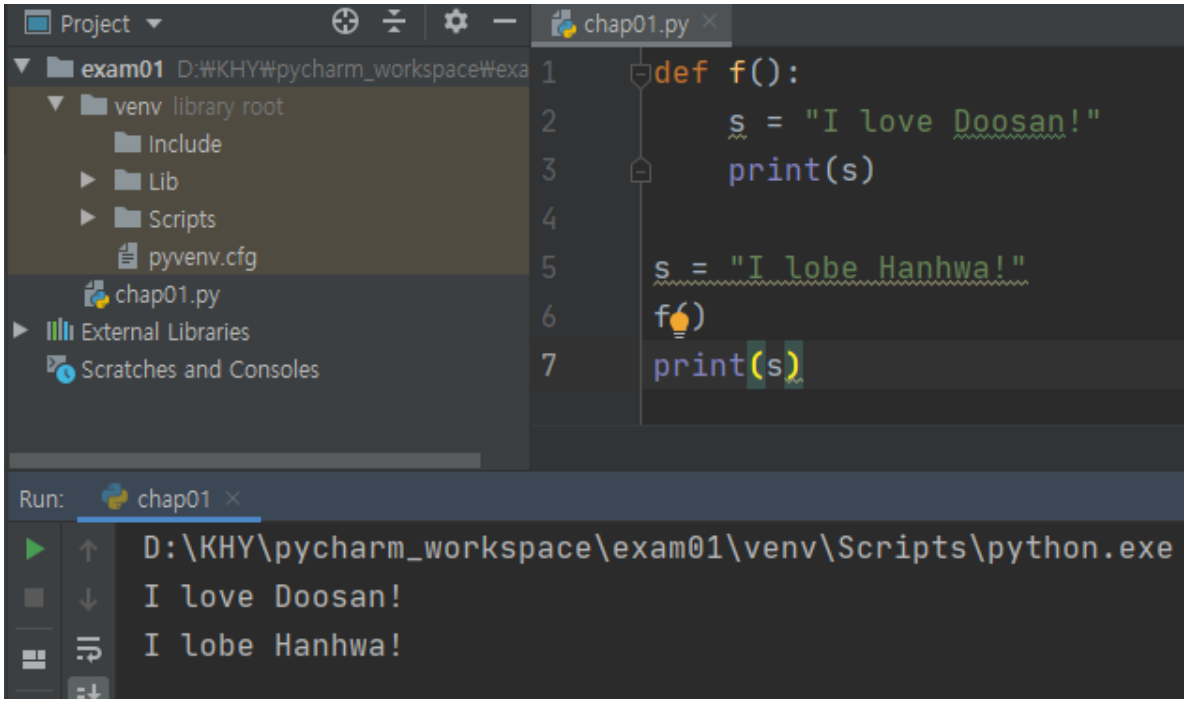
> 함수 인자에 레퍼런스 타입이 들어가면 call by reference로 호출

```
public class MyClass{ // java 예시
    private int field1;
    ...
    public setField1(int field1){
        this.field1 = field1; // 뒤의 field1은 로컬 변수(함수 내에서 선언)
    }
}
```

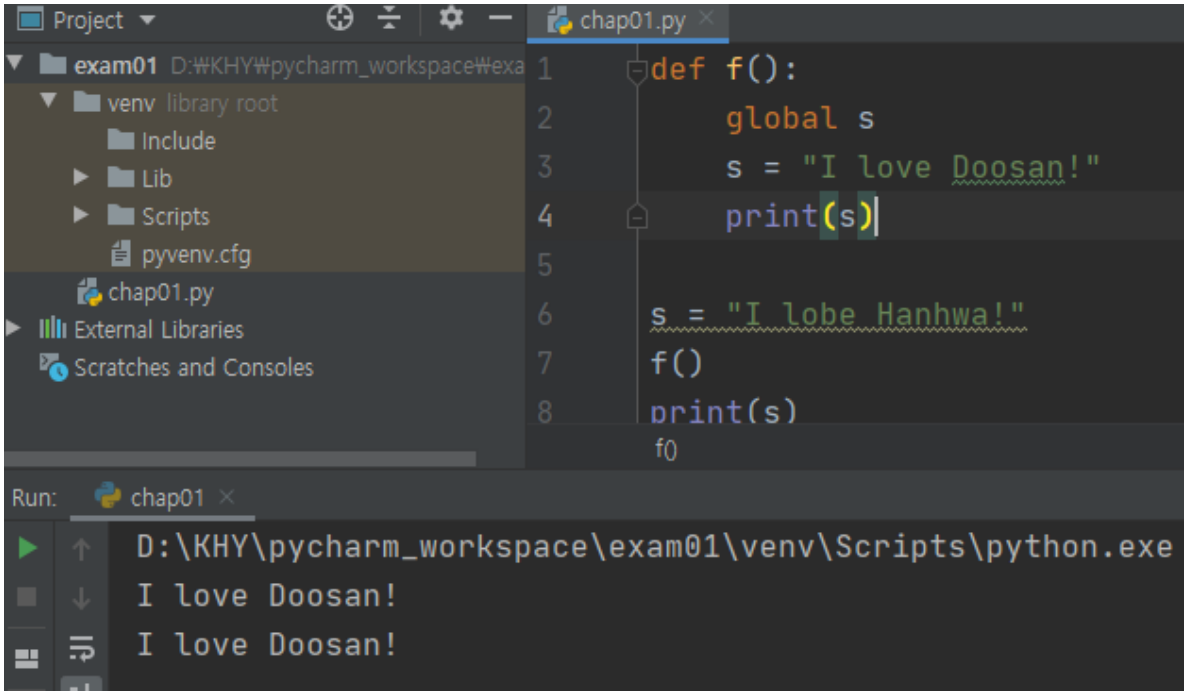
> 1. 스코프 영역이 다름(goods와 shopping_list)
> 2. 매개인자 타입이 레퍼런스 타입이면 오리지널 객체에 수정을 해줄 수 있음(goods를 [1, 2] 리스트로 재참조 했으므로 shopping_list와의 연결고리가 없어짐)

> stack에 shopping_list / Heap에 리스트 내용

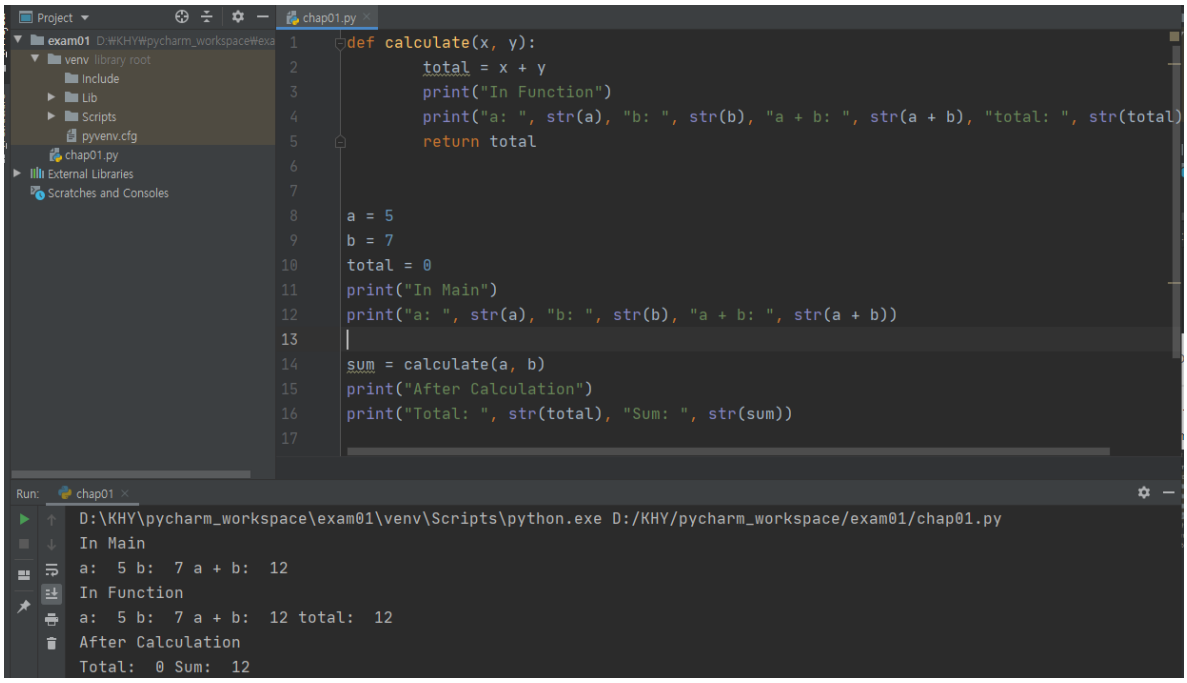




> static 타입의 언어에서 메소드 혹은 함수에서 글로벌 변수에 접근이 가능함
>
> 하지만 파이썬과 같은 다이나믹 타입의 언어에서는 타입으로 변수를 만들지 않고 바로 만들기 때문에 함수 내부에서 글로벌 변수를 접근하고 싶을 때 문제 발생

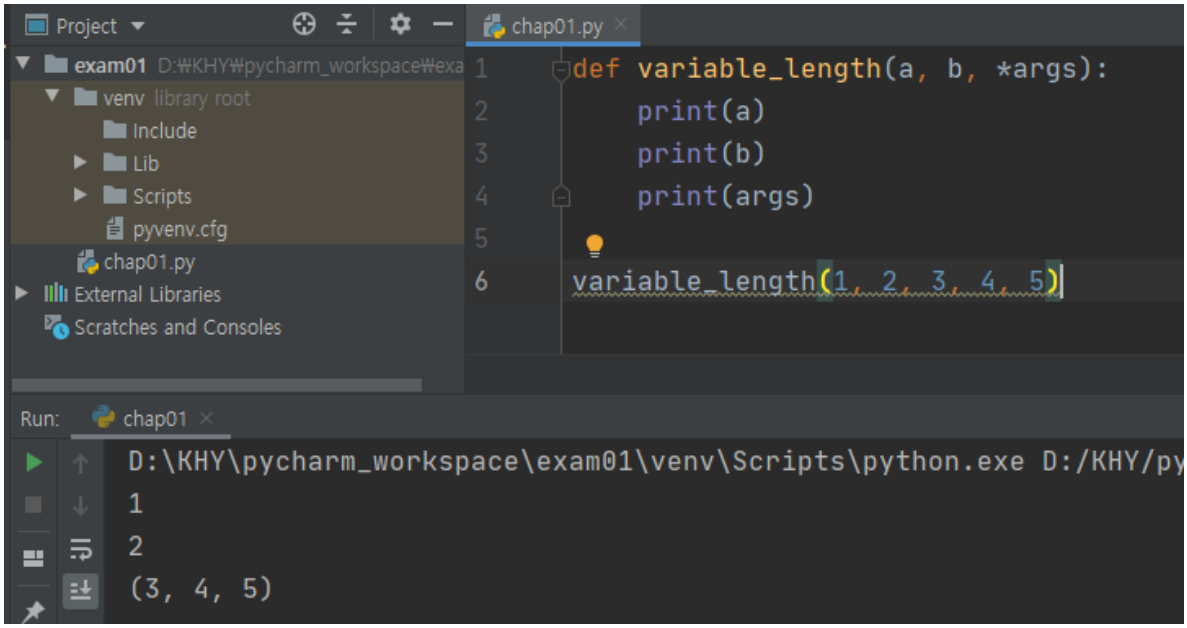


> global ~: 앞으로의 ~변수는 글로벌 영역의 ~변수라는 것을 선언

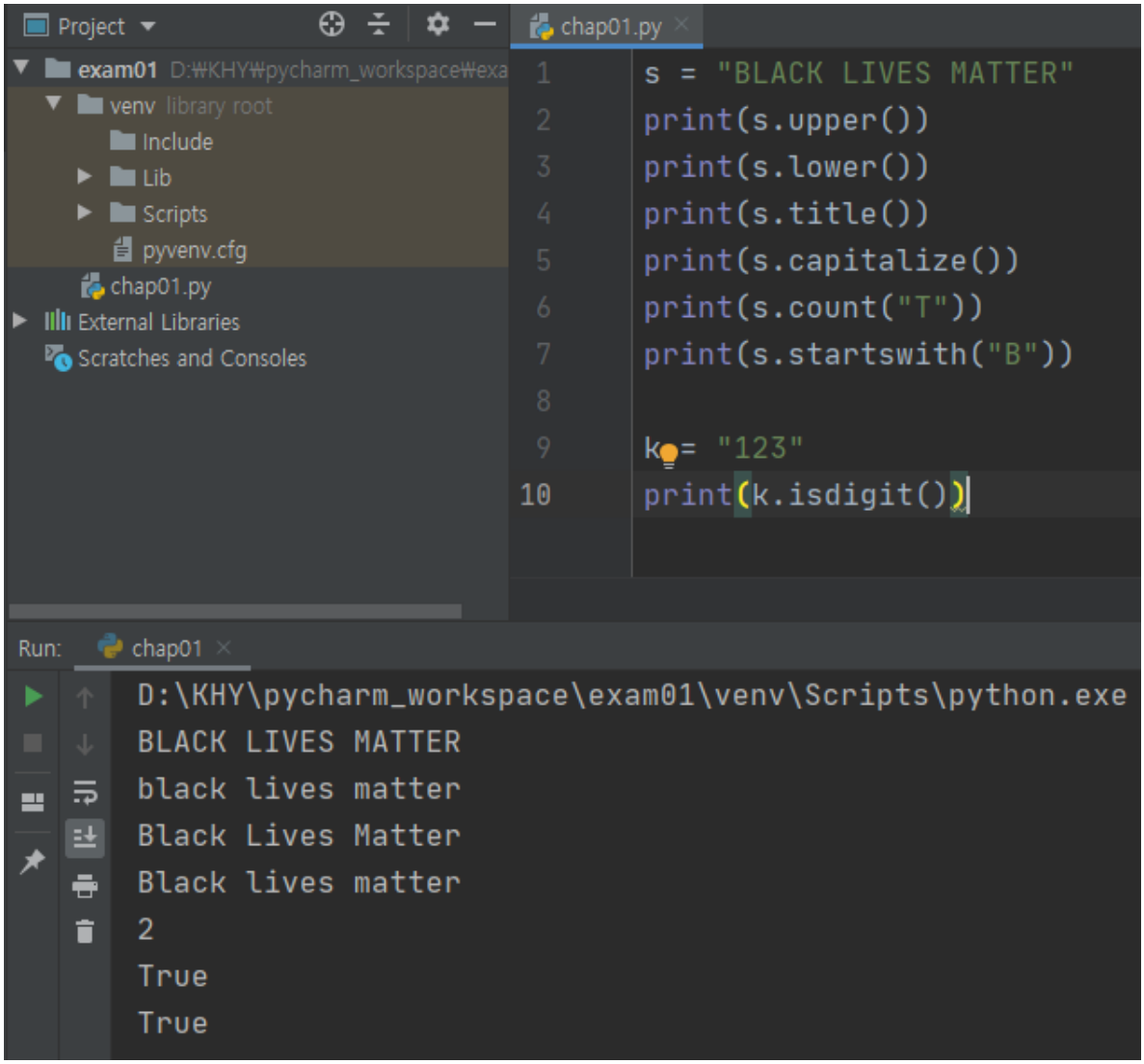


> 직접 명시해서 파라미터에 참조값을 넣을 수 있음

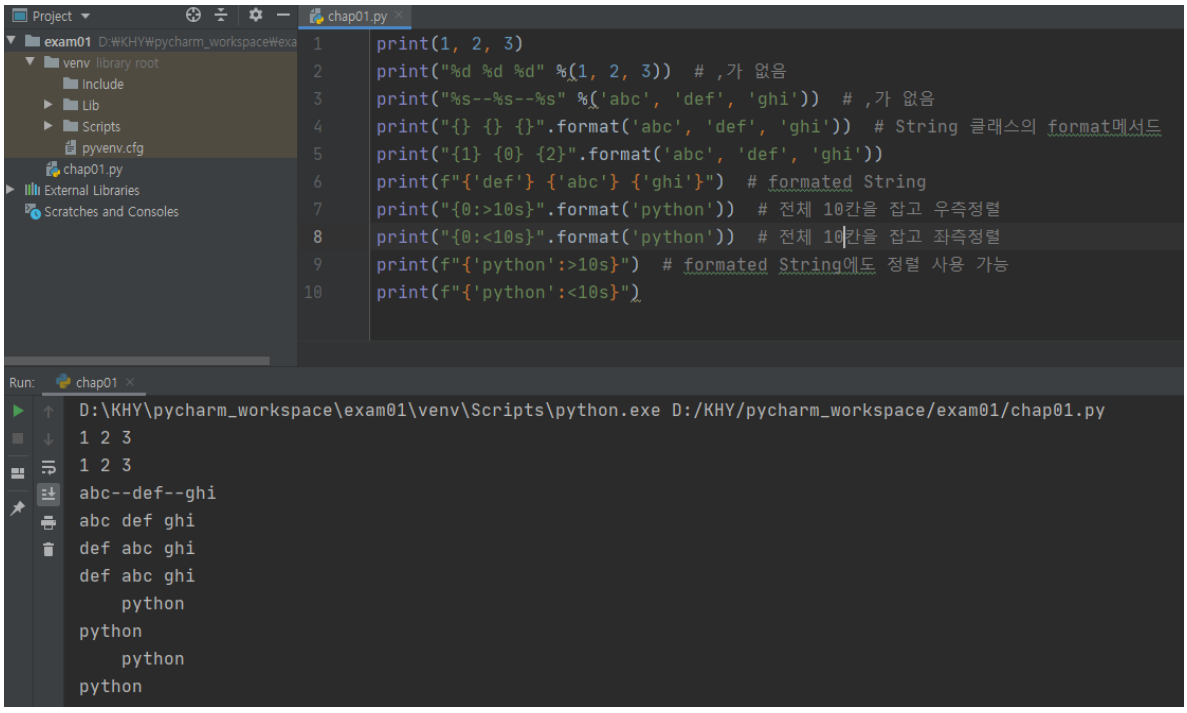
> 디폴트값 주기



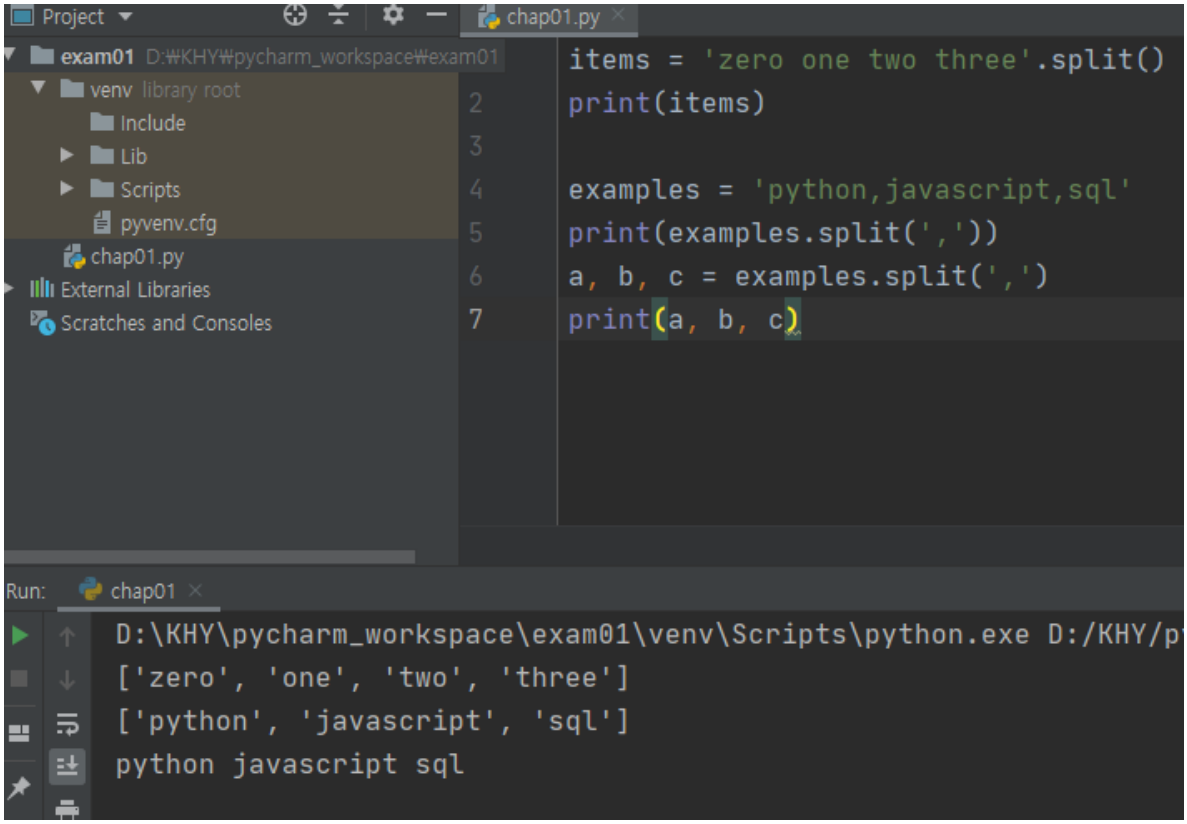
> 가변인자(*~)
>
> 가변인자는 항상 마지막에 위치해야 함



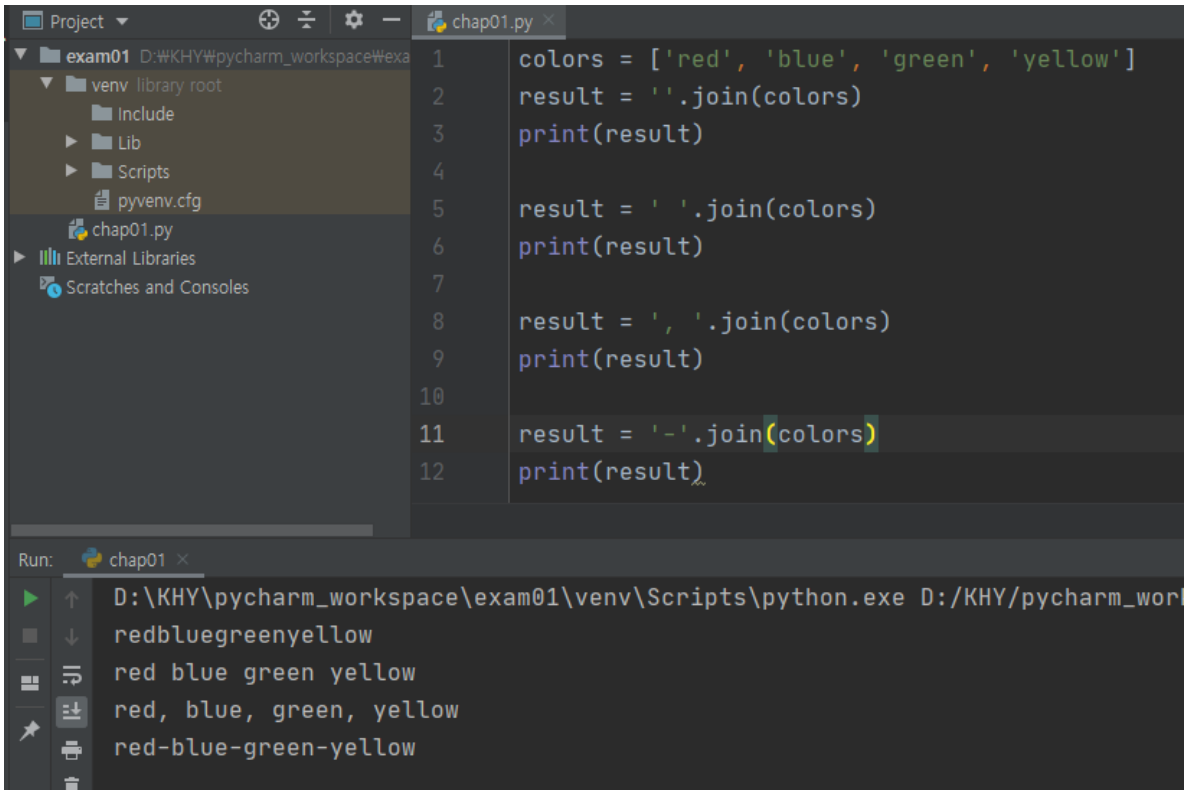
> isdigit() : 숫자로 되어 있는 지 검사



> print 출력 방법

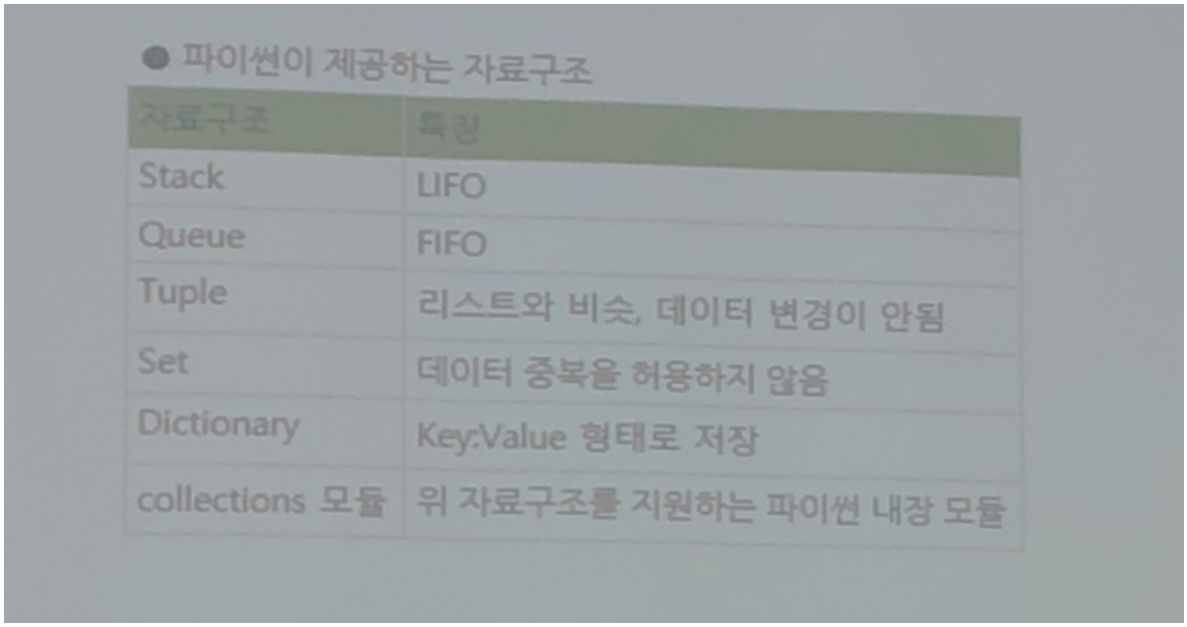


> split()의 결과는 리스트



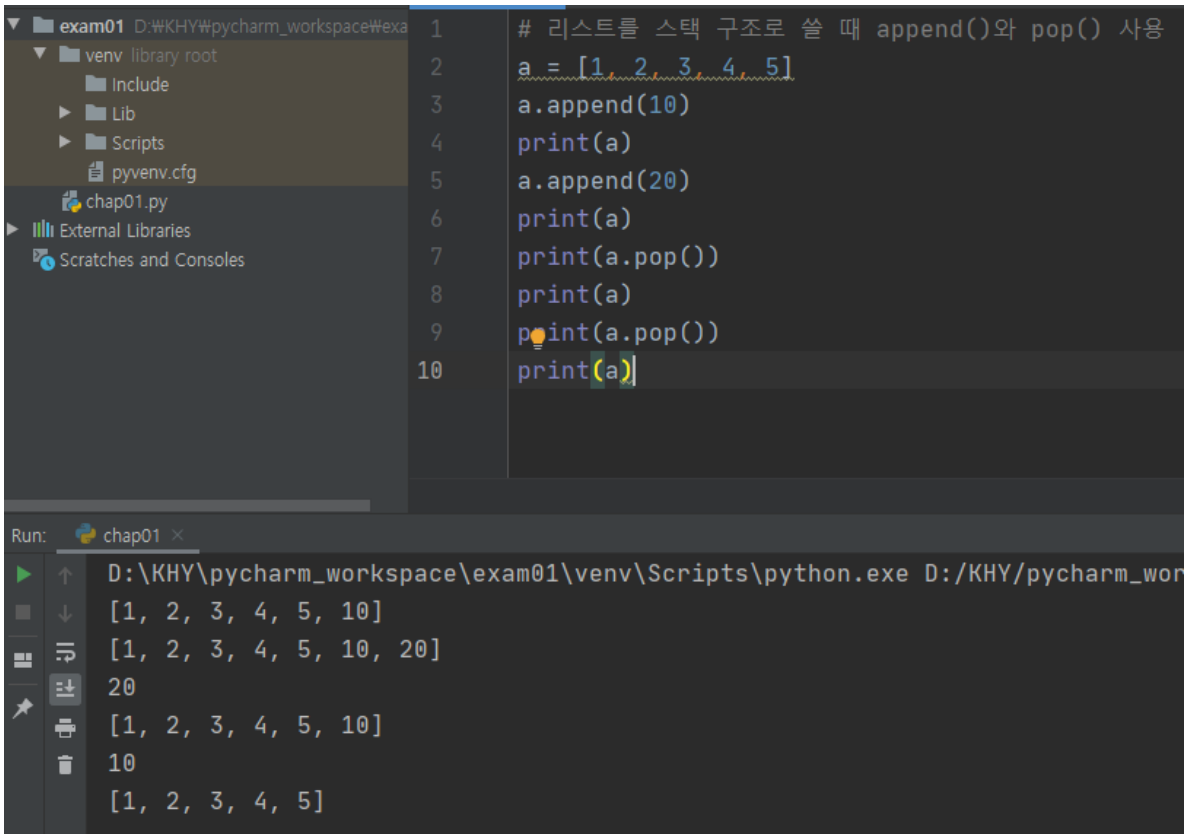
> join() : 리스트를 문자열로 구분자를 지정해서 넣는 메서드인 듯

파이썬의 자료구조

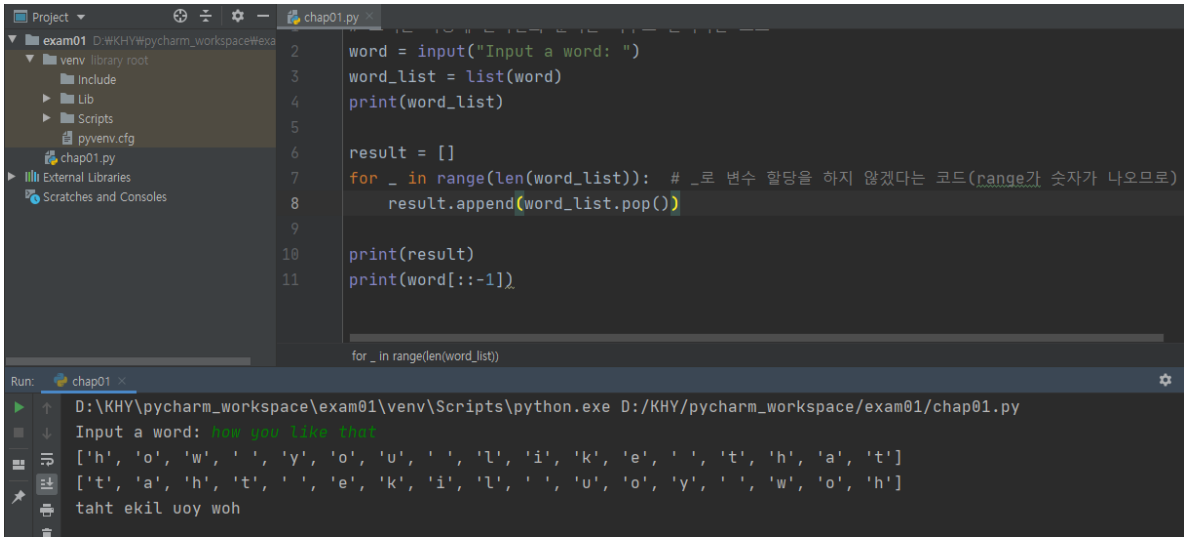


> Set : 집합(중복데이터 허용하지 않음), 순서가 없음(따라서 첨자 없음)

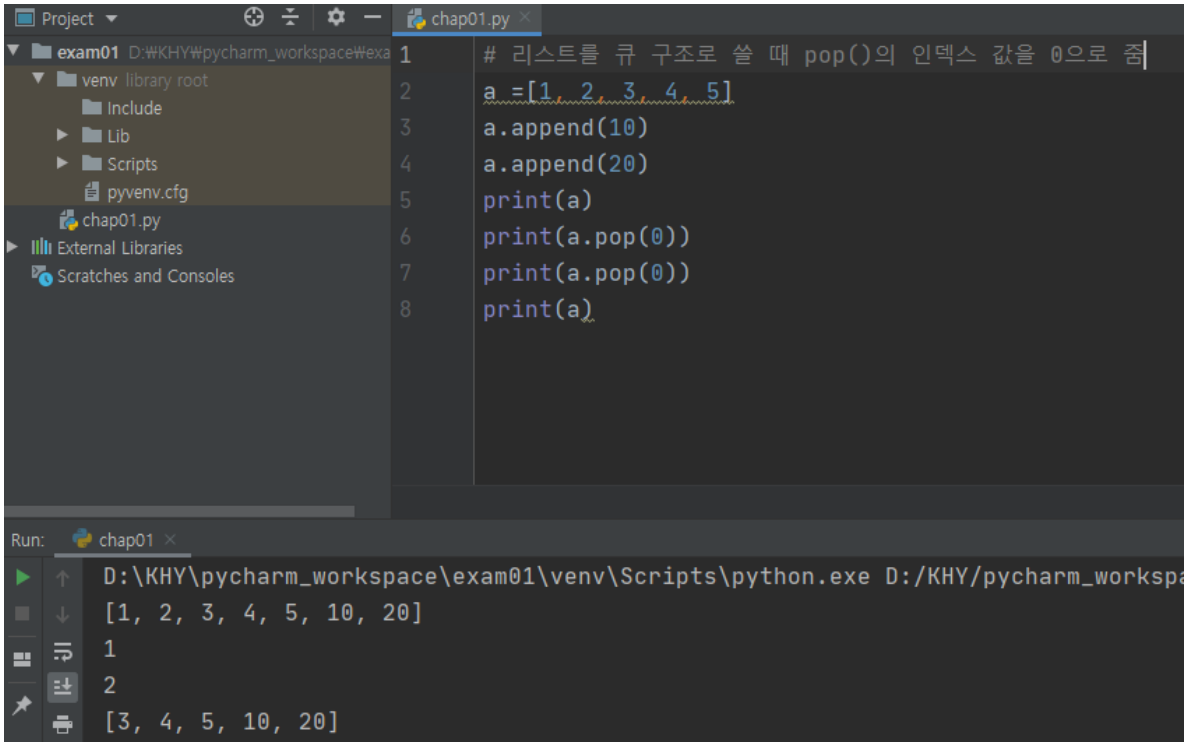
- stack



> append() pop()

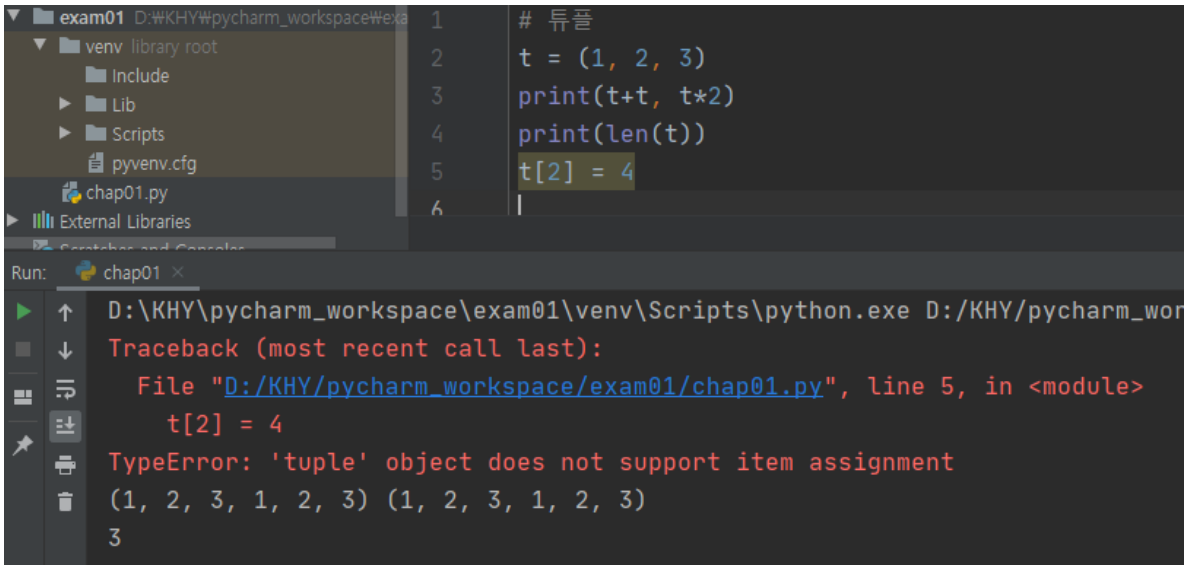


- Queue

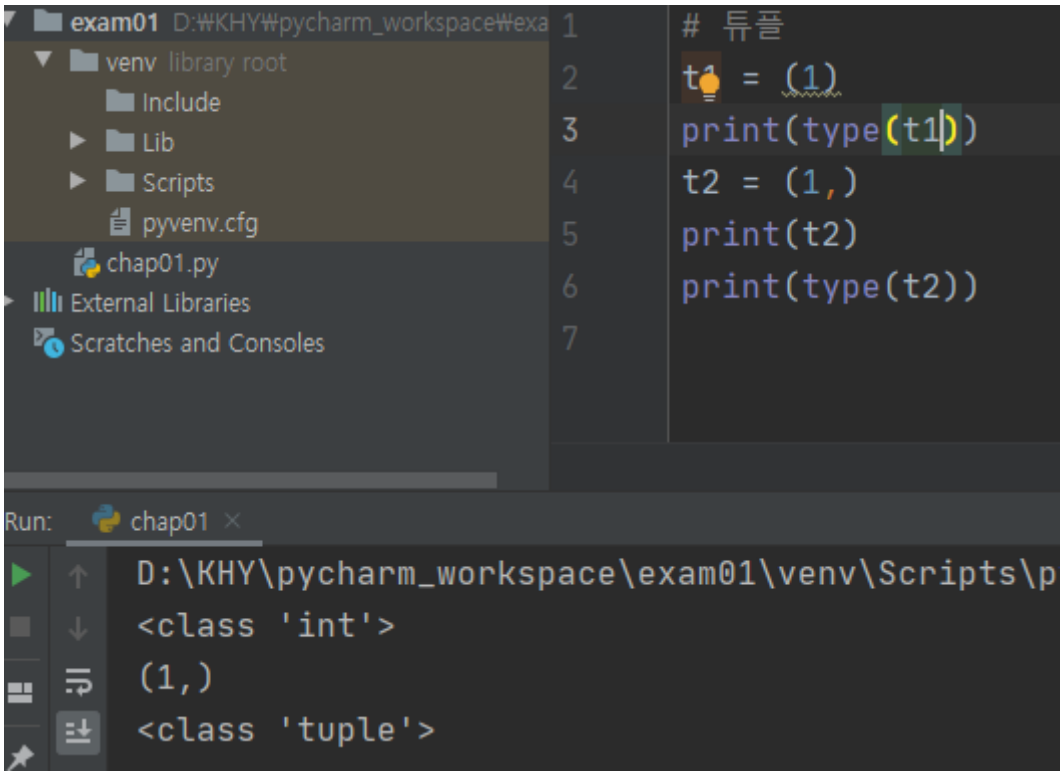


> append() pop(0)

- 튜플

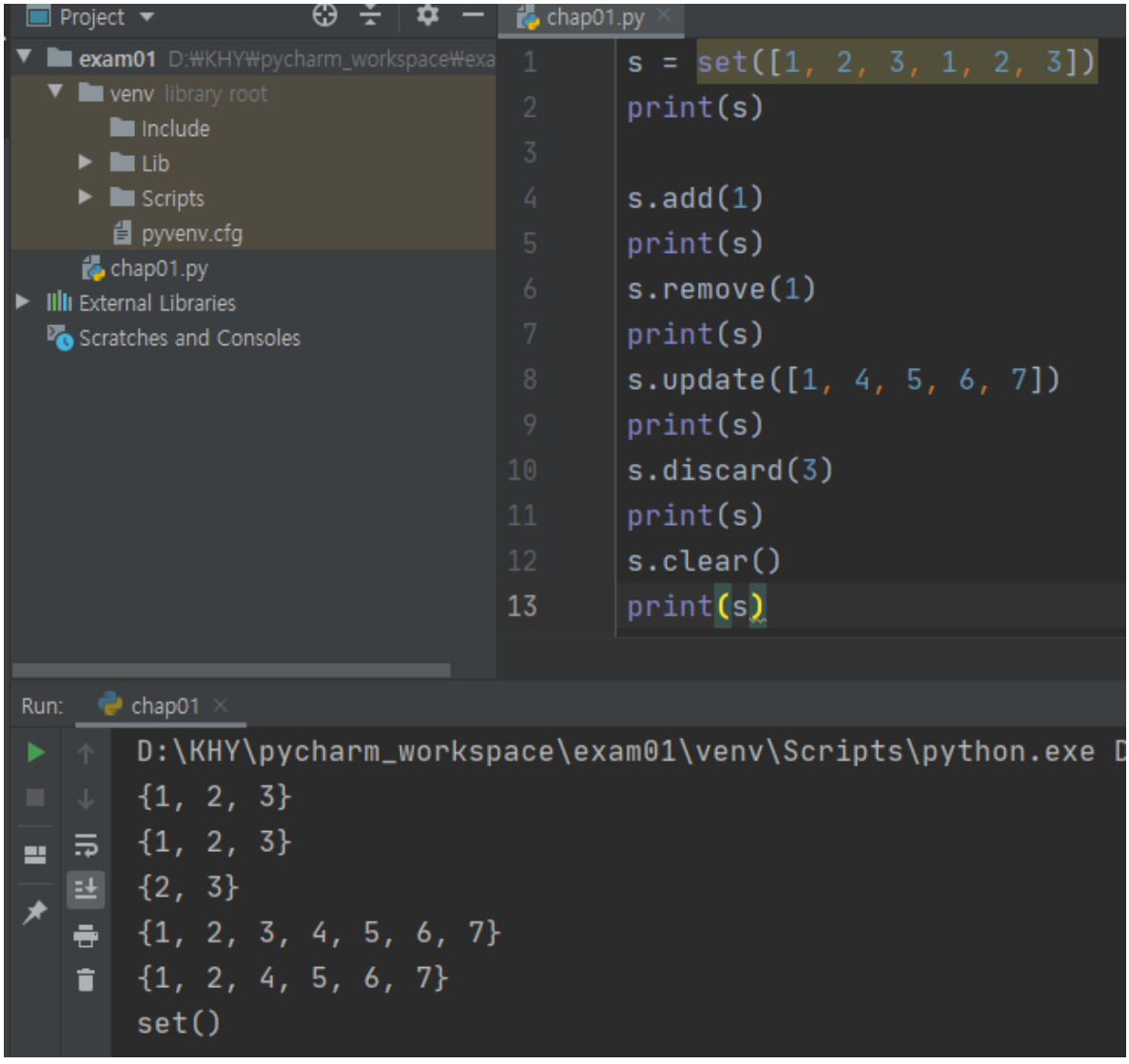


> 항목 바꾸기 불가능



> 원소가 하나 일 때 콤마 필요

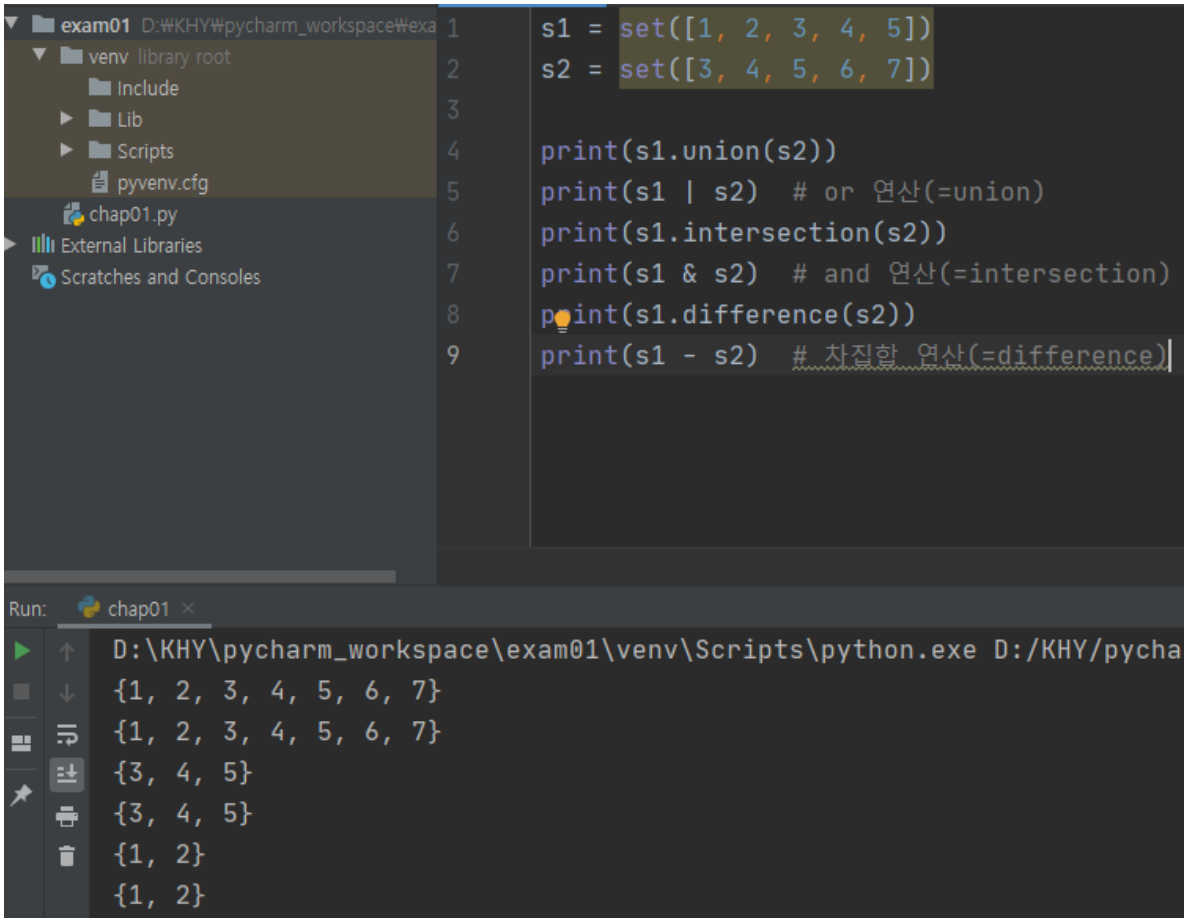
- set



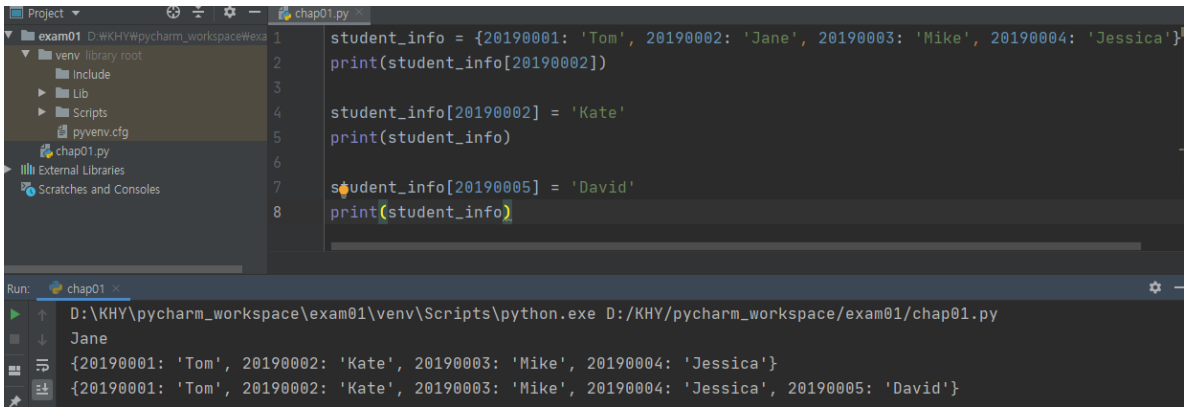
> 중복되지 않음

> discard() : 지우려는 요소가 없어도 정상종료

> remove(): 지우려는 요소가 없으면 KeyError



- dictionary



> 딕셔너리 벨류를 명시적으로 변경 및 추가

> key() 키를 리스트를 만들어 줌

> values() 값을 리스트로 만들어 줌

> items() 키와 값 쌍을 튜플 타입으로 만든 후 리스트로 만들어 줌

> in 연산자로 참/거짓 값 판별

Collection 모듈

- 기본 자료구조를 확장하여 미리 제작하고 파이썬 모듈로 제공
- deque, OrderedDic, defaultdict, Counter, namedtuple
- defaultdic과 Counter를 많이 사용함

deque

```python
from collections import deque  # collections라는 컬렉션 안의 deque 객체 사용

deque_list = deque()  # deque은 양쪽 모두 입출력이 가능(queue는 한 쪽만 입출력 가능
for i in range(5):
    deque_list.append(i)

print(deque_list)

print(deque_list.pop())
print(deque_list.pop())
print(deque_list)

deque_list.appendleft(5)  # 양쪽 입출력이 가능하므로 appendleft() 메서드 이용
deque_list.appendleft(6)
print(deque_list)

print(deque_list.popleft())  # 양쪽 입출력이 가능하므로 popleft() 메서드 이용
print(deque_list.popleft())
print(deque_list)
```

```
deque([0, 1, 2, 3, 4])
4
3
deque([0, 1, 2])
deque([6, 5, 0, 1, 2])
6
5
deque([0, 1, 2])

Process finished with exit code 0
```

```python
from collections import deque  # collections라는 컬렉션 안의 deque 객체 사용

deque_list = deque()  # deque은 양쪽 모두 입출력이 가능(queue는 한 쪽만 입출력 가능)
for i in range(5):
    deque_list.append(i)

print(deque_list)

deque_list.rotate(2)
print(deque_list)

deque_list.rotate(-2)
print(deque_list)

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

deque_list.extendleft([9, 10])
print(deque_list)
```

```
D:\KHY\pycharm_workspace\exam01\venv\Scri
deque([0, 1, 2, 3, 4])
deque([3, 4, 0, 1, 2])
deque([0, 1, 2, 3, 4])
deque([0, 1, 2, 3, 4, 5, 6, 7])
deque([10, 9, 0, 1, 2, 3, 4, 5, 6, 7])
```

> rotate(n) : 회전함수, 리스트 내부에선 n칸 만큼 회전

OrderedDict

```python
from collections import OrderedDict  # 집어넣은 순서를 보장해주는 OrderedDict

d = dict()  # 딕셔너리 타입의 객체 생성
d['x'] = 100  # 키와 벨류 명시적으로 작성
d['y'] = 200
d['z'] = 300
d['a'] = 400
print(d)

od = OrderedDict(sorted(d.items(), key=lambda  x: x[0]))
# items()로 인해 (키:벨류)로 튜플이 들어오므로
# x의 0번 째(키)로(lambda  x: x[0] 부분) sort 하겠다는 뜻

print()
print(od)

od_list = sorted(od.items(), key=lambda x: x[1])
print()
print(od_list)
```

```
D:\KHY\pycharm_workspace\exam01\venv\Scripts\python.exe D:/KHY/pycha
{'x': 100, 'y': 200, 'z': 300, 'a': 400}

OrderedDict([('a', 400), ('x', 100), ('y', 200), ('z', 300)])

[('x', 100), ('y', 200), ('z', 300), ('a', 400)]
```

> 잘 쓰이지는 않음

defaultdict

```python
from collections import defaultdict

d = defaultdict(lambda: 100)  # 기본 디폴트 값을 100으로 줌
print(d['first'])  # 디폴트 값이 있기 때문에 아무 이름으로 지어서 출력해도 나옴

s = [('yellow', 1), ('blue', 2), ('yellow', 3), ('blue', 4), ('red', 1)]
# 튜플의 리스트 s
d2 = defaultdict(list)  # d2의 타입을 리스트로 만듦

d3 = defaultdict(int)  # 인티저 타입의 기본 디폴트 값은 0으로 되어있음
print(d3['aaa'])

for k, v in s:
    d2[k].append(v)  # d2가 리스트 타입이기 때문에 append() 가능

print(d2.items())
```

```
D:\KHY\pycharm_workspace\exam01\venv\Scripts\python.exe D:/KHY/pycharm
100
0
dict_items([('yellow', [1, 3]), ('blue', [2, 4]), ('red', [1])])
```

> 어떤 값을 누적할 때 많이 쓰임

Counter

```
from collections import Counter

c = Counter('scientist')  # 카운팅만을 위한 것
print(c)

print(c['i'])
print(c['n'])
```

```
D:\KHY\pycharm_workspace\exam01\venv\Scripts\python.exe D:
Counter({'s': 2, 'i': 2, 't': 2, 'c': 1, 'e': 1, 'n': 1})
2
1
```

```
from collections import Counter

c = Counter({'red': 4, 'blue': 2})  # 카운팅만을 위한 것
print(c)

print(list(c.elements()))  # getlist를 만드는 것과 같음
```

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

```
from collections import Counter

c = Counter(cats=4, dogs=6)  # 카운팅만을 위한 것
print(c)

print(list(c.elements()))  # getlist를 만드는 것과 같음
```

```
D:\KHY\pycharm_workspace\exam01\venv\Scripts\python.exe D:/KHY/pycharm_workspace/e
Counter({'dogs': 6, 'cats': 4})
['cats', 'cats', 'cats', 'cats', 'dogs', 'dogs', 'dogs', 'dogs', 'dogs', 'dogs']
```

> 키워드 매개변수

```
from collections import Counter

c = Counter(a=4, b=2, c=0, d=-2)  # 0과 음수는 False로 옴
d = Counter(a=1, b=2, c=3, d=4)

print(c + d)
print(c & d)
print(c | d)

c.subtract(d)
print(c)
```

```
Counter({'a': 5, 'b': 4, 'c': 3, 'd': 2})
Counter({'b': 2, 'a': 1})
Counter({'a': 4, 'd': 4, 'c': 3, 'b': 2})
Counter({'a': 3, 'b': 0, 'c': -3, 'd': -6})
```

실습

```
from collections import defaultdict
from collections import OrderedDict

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)

word_count = defaultdict(lambda : 0)  # 단어의 출현 갯수를 카운팅하는 것이므로 defaultdict
# lambda : 0 은 값을 0으로 주겠다(int로 사용해도 무방)

for word in text:
    word_count[word] += 1
print(word_count)

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

```
D:\KHY\pycharm_workspace\exam01\venv\Scripts\
['a', 'press', 'release', 'is', 'the', 'quick
defaultdict(<function <lambda> at 0x000002185
and 3
to 3
a 2
press 2
release 2
can 2
you 2
is 1
the 1
```

리스트 심화

- 리스트 컴프리헨션(한 줄 코드로 이뤄져있어 실행속도 향상)

```
result = []
for i in range(10):
    result.append(i)
print(result)
print()

result2 = [i for i in range(10)]  # list comprehension
print(result2)
```

```
D:\KHY\pycharm_workspace\exam01\v
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
result = []
for i in range(10):
    if i % 2 == 0:
        result.append(i)
print(result)
print()

result2 = [i for i in range(10) if i % 2 == 0]  # list comprehension
print(result2)
```

```
D:\KHY\pycharm_wor
[0, 2, 4, 6, 8]

[0, 2, 4, 6, 8]
```

```python
result3 = [i if i % 2 == 0 else 99 for i in range(10)]  # list comprehension
#else문이 추가되면 for문이 뒤로 빠짐
print(result3)
```

```
D:\KHY\pycharm_workspace\exam01\venv\
[0, 99, 2, 99, 4, 99, 6, 99, 8, 99]
```

> elif 문은 사용할 수 없음

```python
word1 = 'Hello'
word2 = 'world'
result = [i + j for i in word1 for j in word2]
print(result)
```

```
D:\KHY\pycharm_workspace\exam01\venv\Scripts\p
['Hw', 'Ho', 'Hr', 'Hl', 'Hd', 'ew', 'eo', 'er
```

```python
"""
result = []
for i in word1:
    for j in word2:
        result.append(i + j)
"""
```

> 이중 for문 컴프리헨션

```python
case1 = ['A', 'B', 'C']
case2 = ['A', 'B', 'C']
result = [i + j for i in case1 for j in case2 if i != j]
print(result)
```

```
D:\KHY\pycharm_workspace\exam01\venv\S
['AB', 'AC', 'BA', 'BC', 'CA', 'CB']
```

```python
case1 = ['A', 'B', 'C']
case2 = ['D', 'E', 'F']
result = [[i + j for i in case1] for j in case2]  # 뒤에 있는 for문이 주가 됨
print(result)
```

```
D:\KHY\pycharm_workspace\exam01\venv\Scripts\python.exe D:/KHY/pyc
[['AD', 'BD', 'CD'], ['AE', 'BE', 'CE'], ['AF', 'BF', 'CF']]
```

> 이중리스트
>
> 뒤에 작성한 for문 코드가 베이스