컬렉션 관리

컬렉션 관리 함수

❖ enumerate

- enumerate(시퀀스 [, start])
  - 시퀀스의 인덱스와 요소를 튜플 묶어서 순회

```
score = [88, 95, 70, 100, 99]

for s in score:
    print("성적 : ", s)
```

```
성적 :  88
성적 :  95
성적 :  70
성적 :  100
성적 :  99
```

컬렉션 관리 함수

❖ enumerate

```
score = [88, 95, 70, 100, 99]

no = 1
for s in score:
    print(str(no) + "번 학생의 성적 : ", s)
    no += 1
```

```
1번 학생의 성적 :  88
2번 학생의 성적 :  95
3번 학생의 성적 :  70
4번 학생의 성적 :  100
5번 학생의 성적 :  99
```

컬렉션 관리 함수

❖ enumerate

```
score = [88, 95, 70, 100, 99]

for no in range(len(score)):
    print(str(no+1) + "번 학생의 성적 : ", score[no])
```

```
1번 학생의 성적 :  88
2번 학생의 성적 :  95
3번 학생의 성적 :  70
4번 학생의 성적 :  100
5번 학생의 성적 :  99
```

컬렉션 관리 함수

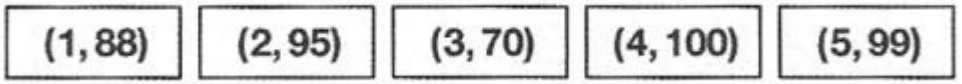
❖ enumerate

```
race = ['저그', '테란', '프로토스']

list(enumerate(race))
```

```
[(0, '저그'), (1, '테란'), (2, '프로토스')]
```

```
score = [88, 95, 70, 100, 99]

for no, s in enumerate(score, 1):
    print(str(no) + "번 학생의 성적 : ", s)
```

```
1번 학생의 성적 :  88
2번 학생의 성적 :  95
3번 학생의 성적 :  70
4번 학생의 성적 :  100
5번 학생의 성적 :  99
```



컬렉션 관리 함수

❖ zip

- zip(시퀀스1, 시퀀스2) -> [(값1, 값2), ...]
- 시퀀스의 길이가 다른 경우 가장 짧은 시퀀스의 길이에 맞춤

```
dates = ["월", "화", "수", "목", "금", "토", "일"]
food = ["갈비탕", "순대국", "칼국수", "삼겹살"]

menu = zip(dates, food)
for d, f in menu:
    print("%s요일 메뉴: %s" % (d, f))
```

```
월요일 메뉴: 갈비탕
화요일 메뉴: 순대국
수요일 메뉴: 칼국수
목요일 메뉴: 삼겹살
```

컬렉션 관리 함수

❖ zip

```
menu_dic = dict(zip(dates, food))
print(menu_dic)
```

```
{'월': '갈비탕', '화': '순대국', '수': '칼국수', '목': '삼겹살'}
```

컬렉션 관리 함수

❖ any(), all()

- any(시퀀스)
  - 시퀀스에 하나라도 True가 있으면 True 리턴

- all(시퀀스)
  - 시퀀스의 모든 요소가 True이면 True 리턴

```
adult = [True, False, True, False]

print(any(adult))
print(all(adult))
```

```
True
False
```

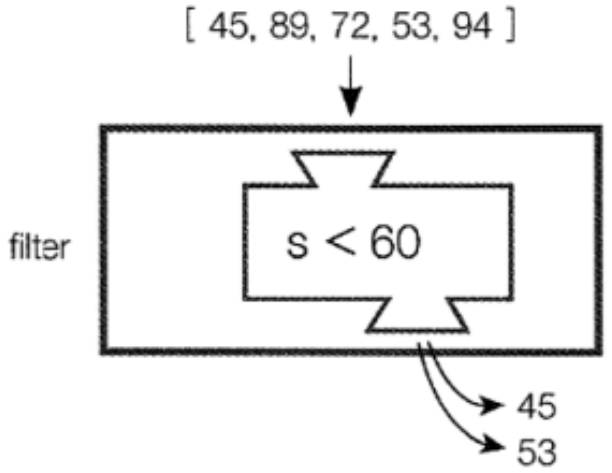
람다 함수

❖ filter

- filter(판정함수, 시퀀스) -> 시퀀스
  - 시퀀스의 각 요소를 판정함수에 전달하여 True를 리턴하는 요소로만 구성된 새로운 시퀀스 리턴

```
def flunk(s):
    return s < 60

score = [ 45, 89, 72, 53, 94 ]
for s in filter(flunk, score):
    print(s)
```

```
45
53
```



람다 함수

❖ map

```
def total(s, b):
    return s + b

score = [ 45, 89, 72, 53, 94 ]
bonus = [2, 3, 0, 0, 5]
for s in map(total, score, bonus):
    print(s, end = ", ")
```

```
47, 92, 72, 53, 99,
```

람다 함수

❖ 람다 함수

- 한 줄로 정의되는 함수의 축약 표현
- 함수의 이름이 없음
  - 변수에 대입해서 사용
- lambda 인수:식

```
lambda x: x + 1

def increase(x):
    return x + 1

score = [ 45, 89, 72, 53, 94 ]
for s in filter(lambda x: x < 60, score):
    print(s)
```

```
45
53
```

람다 함수

❖ 람다 함수

```
score = [ 45, 89, 72, 53, 94 ]
for s in map(lambda x: x / 2, score):
    print(s, end = ", ")
```

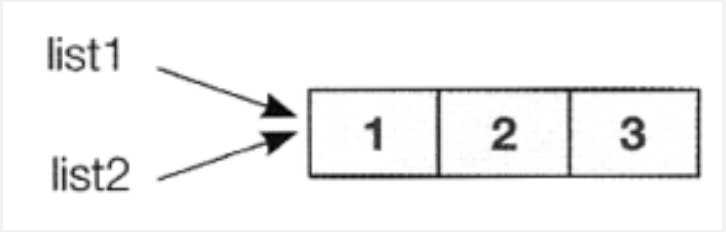
```
22.5, 44.5, 36.0, 26.5, 47.0,
```

컬렉션의 사본

❖ 리스트의 사본

- 시퀀스.copy() -> 시퀀스 복사본

```
list1 = [1, 2, 3]
list2 = list1

print(list1 == list2)

list2[1] = 100
print(list1)
print(list2)

print(list1 == list2)
```




```
True
[1, 100, 3]
[1, 100, 3]
True
```

컬렉션의 사본

❖ 리스트의 사본

```
list1 = [1, 2, 3]
list2 = list1.copy()     # list2 = list1[:]와 동일

print(list1 == list2)

list2[1] = 100
print(list1)
print(list2)

print(list1 == list2)
```

```
True
[1, 2, 3]
[1, 100, 3]
False
```

컬렉션의 사본

❖ 리스트의 사본

```
list0 = ['a', 'b']
list1 = [list0, 1, 2]
list2 = list1.copy()  # 얕은 복사

list2[0][1] = 'c'
print(list1)
print(list2)
```

```
[['a', 'c'], 1, 2]
[['a', 'c'], 1, 2]
```

컬렉션의 사본

❖ 리스트의 사본

```
import copy

list0 = ['a', 'b']
list1 = [list0, 1, 2]
list2 = copy.deepcopy(list1)  # 깊은 복사

list2[0][1] = 'c'
print(list1)
print(list2)
```

```
[['a', 'b'], 1, 2]
[['a', 'c'], 1, 2]
```

컬렉션의 사본

❖ is 연산자

- 두 변수가 같은 객체를 가리키고 있는지 조사

```
list1 = [1, 2, 3]
list2 = list1
list3 = list1.copy()  # 얕은 복사

print("list1 == list2", list1 is list2)
print("list1 == list3", list1 is list3)
print("list2 == list3", list2 is list3)
```

```
list1 == list2 True
list1 == list3 False
list2 == list3 False
```