Lecture 5. 파이썬 컬렉션

BAE SEKANG(배세강)

Contents

Mutable & Immutable

♣ 가변 V.S 불변

- 가변 객체: 객체를 생성한 후에도 그 값을 수정하거나, 삭제, 추가할 수 있다.
- 불변 객체: 객체를 생성한 후 값을 변경할 수 없다.

	Mutable(가변)	Immutable(불변)
설명	메모리 안에 담긴 값이 변할 수 있는 객체	메모리 안에 담긴 값이 항상 변하지 않는 객체
예시	list, set, dictionary 등	int, float, tuple 등

Mutable & Immutable

♣ 가변 V.S 불변

- id(): 객체가 생성되어 저장된 객체의 고유 식별값을 반환한다.
- 내부 최적화에 따라 id가 같게 나오는 경우도 있다.

```
# id() 활용 예시 1
a = 30
b = [10, 20, 30]
print(id(a))
print(id(30))
print(id(b))
print(id(b[0]))
print(id(b[2]))
```

```
>> 140707673032520
>> 140707673032520
>> 1385427854784
>> 140707673031880
>> 140707673032520
```

Mutable & Immutable

♣ 가변 V.S 불변

- id(): 객체가 생성되어 저장된 메모리 주소를 반환한다.

```
# id() 활용 예시 2
a = [10, 20]
b = [a, [30, 40]]
print(b)
a[0] = 40
print(b)
print(id(a))
print(id(b[0]))
```

```
>> [[10, 20], [30, 40]]

>> [[40, 20], [30, 40]]
>> 2771632350656
>> 2771632350656
```

Mutable & Immutable

♣ 가변 V.S 불변

- 가변 객체는 내부 값을 수정해도 가변 객체 자체의 id() 값은 변하지 않는다.

```
# id() 활용 예시 2
a = [1, "2", 3]
print(id(a))
a.append(4)
print(id(a))
print(id(a[1]))
print(id("2"))
print(id("2"+"2"))
```

```
>> 2142827472320

>> 2142827472320
>> 140707653826320
>> 140707653826320
>> 2142829306736
```

List

♣ 얕은 복사와 깊은 복사

- 얕은 복사: 원본을 복사하여 새로운 객체를 만들지만, 내부 요소들은 원본 객체의 요소에 대한 참조값만 복사한다.
- 깊은 복사: 원본을 복사하여 새로운 객체를 만들되, 내부 모든 요소들을 재귀적으로 복사하여 완전히 새로운 객체를 생성한다.

```
a = [10, [20, 30]]
b = a.copy()
b[1][0] = 100
print(a)
# deepcopy는?
import copy
b = copy.deepcopy(a)
```

>> [10, [100, 30]]

Dictionary

♣ 딕셔너리

- key를 활용하여 value를 찾을 수 있다.
- key는 immutable한 자료형으로 해야한다.

```
my_key = 50
a = {10: "Hello", "K": 35, my_key: [10, 20]}
print(a[10])
print(a["K"])
print(a[50])
a[10] = "Hi!"                    #딕셔너리의 값 변경
print(a)
print(type(a))
```

```
>> Hello
>> 35
>> [10, 20]

>> {10: 'Hi!', 'K': 35, 50: [10, 20]}
>> <class 'dict'>
```

Dictionary

♣ 딕셔너리 활용

- del, get(), keys(), values(), items()

```
d = {10: "a", 20: "b", 30: None}
del d[30]
print(d)
print(d.get(10, "No!"))
print(d.get(40, "No!"))
print(d.keys())
print(d.values())
print(d.items())
```

```
>> {10: 'a', 20: 'b’}
>> a
>> No!
>> dict_keys([10, 20])
>> dict_values(['a', 'b’])
>> dict_items([(10, 'a'), (20, 'b')])
```

Dictionary

[Practice]

다음 세 인물의 이름을 key로, 나이를 value로 가진 딕셔너리를 생성하고, 이를 출력한다.

Alice: 19세
Bob: 23세
Charlie: 31세

Tuple

♣ 튜플

- immutable한 자료형

```
a = (10, 20, 30)
print(a)
print(type(a))
print(a[0])
a[1] = 200          #Tuple 내부 원소 변경 불가!
```

```
>> (10, 20, 30)
>> <class 'tuple'>
10
>> TypeError
```

```
a = (10)
b = (10,)
print(type(a))
print(type(b))
```

```
>> <class 'int'>
>> <class 'tuple'>
```

Tuple

♣ 튜플

- tuple 자체는 immutable하지만, 내부에 mutable한 객체는 바뀔 수 있다.

```
t = ([1, 2], 3)
t[0].append(99)
print(t)
```

>> ([1, 2, 99], 3)

Tuple

♣ 언패킹(Unpacking)

- 개별 원소를 추출한다.

```
my_tuple = (10, 20, 30)
a, b, c = my_tuple
print(a, b, c)
d = 40, 50, 60
print(d)
# *는 tuple을 풀어서 원소들을 넘겨준다.
print(*d)
```

>> 10 20 30

>> (40, 50, 60)

>> 40 50 60

Set

♣ 집합(Set)

- 중복된 원소를 갖지 않는다.
- 원소 간의 순서가 없다. (출력 시 사용자에게 편하게 정렬되어 출력되기도 함)

```
my_set = {10, 20, 30}
print(type(my_set))
my_set.add(40)
my_set.add(40)
my_set.remove(10)
print(my_set)
```

>> <class 'set'>

>> {40, 20, 30}

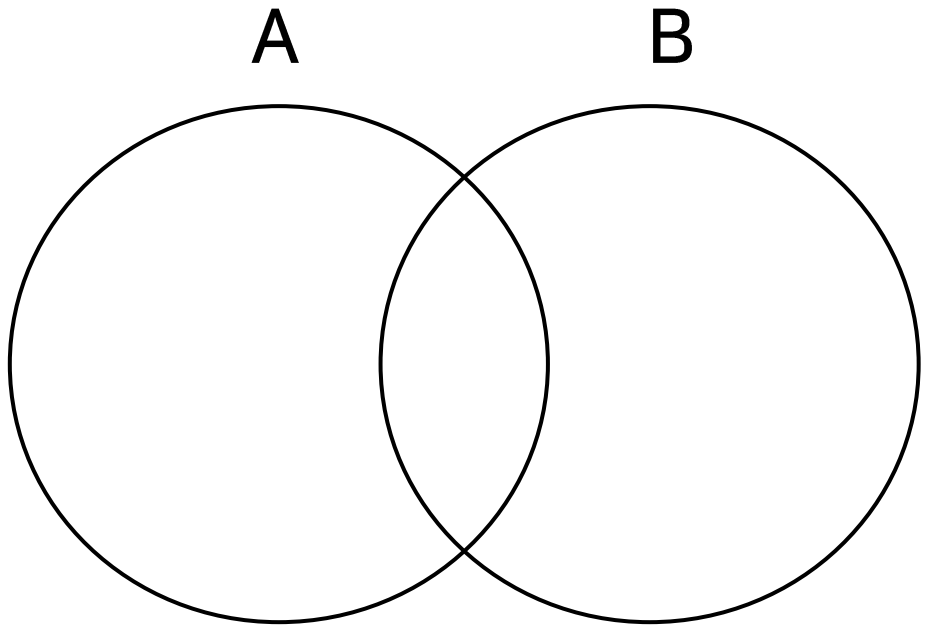
Set

♣ 집합의 연산

- union, intersection, difference, symmetric difference

```
A = {1, 2, 3}
B = {3, 4, 5}

A | B   # 합집합: {1, 2, 3, 4, 5}
A & B   # 교집합: {3}
A - B   # 차집합: {1, 2}
A ^ B   # 대칭차집합: {1, 2, 4, 5}
```



QUIZ

Problem 1.

다음 프로그램의 출력 결과는?

```
d = {90: "A", 80: "B"}
d[70] = "C"
s1 = [90, 80]
s2, s3 = [80, 70, 90], [80, 90]
scores = s1, s2, s3
scores = list(scores)
s2[2]=70
s3[0]=80
print(d[scores[1][2]])
```

THANK YOU!

Any Question?