Lesson 7

Collection의 복사

목차

Mutable(가변)과 Immutable(불변)

Mutable 자료형과 Immutable 자료형

Python

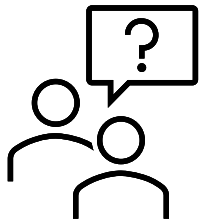
- Mutable : 생성 후에도 **변경이 가능한 자료형**.
  - 리스트(list), 셋(set), 딕셔너리(dictionary)
- Immutable : 생성된 후에는 **변경이 불가능한 자료형**.
  - 정수, 실수, 문자열, 튜플

코드

```
word="hello"
print(id(word))
word="python"
print(id(word))
li=[1,2,3]
print(id(li))
li[0]=10
print(id(li))
```

```
1799113442416
1799113456816
1799111851648
1799111851648
```

리스트는 변경후에도 id가 동일함



1.불변(Immutable) 자료형에는 어떤 것이 있는가?

얕은 복사(shallow copy)와 깊은 복사(deep copy)

얕은 복사(Shallow copy)

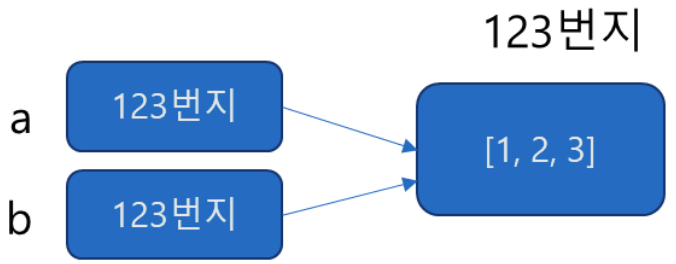
Python

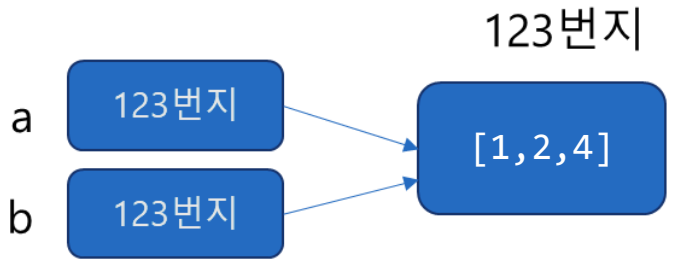
방법1. 대입 연산자(=) 사용

코드

```
a = [1, 2, 3]
b = a     # 주소를 넘겨주는 방식
print(a) # [1, 2, 3]
print(b) # [1, 2, 3]
# 데이터 변경
b[2] = 4

print(b) # [1, 2, 4]
print(a) # [1, 2, 4]
```





※ b의 요소를 변경했는데 a도 변경되는 것을 볼 수 있다.
얕은 복사로 인해 발생.

얕은 복사(Shallow copy) 방법

Python

방법2. 전체 슬라이싱 [:]

코드

```
a = [1, 2, 3]
b = a[:]
b[0] = 100
b.append(5)
print('a:', a)
print('b:', b)
```

```
a: [1, 2, 3]
b: [100, 2, 3, 5]
```

얕은 복사(Shallow copy) 방법

Python

방법3. copy()메소드 사용

코드

```
a = [1, 2, 3]
b = a.copy()
b[0] = 100
b.append(5)
print(a)
print(b)
```

```
a: [1, 2, 3]
b: [100, 2, 3, 5]
```

얕은 복사(Shallow copy) 방법

Python

[:]와 copy()도 완벽하지 않다!

코드

```
a = [[1, 2, 3],[4,5,6]]
b = a.copy()
b[0][0] = 100
b.append(5)
print(a)
print(b)
```

mutable 변수 내부에 또 **mutable**이 있는 경우, 얕은 복사로는 **mutable** 내부의 메모리 주소는 달라지지 않는다.

```
[[100, 2, 3], [4, 5, 6]]
[[100, 2, 3], [4, 5, 6], 5]
```

깊은 복사(Deep copy)

Python

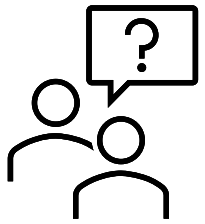
copy모듈의 deepcopy()함수 사용

코드

```
import copy

a = [[1, 2, 3],[4,5,6]]
b = copy.deepcopy(a)
b[0][0] = 100
b.append(5)
print(a)
print(b)
```

```
[[1, 2, 3], [4, 5, 6]]
[[100, 2, 3], [4, 5, 6], 5]
```



1.불변(Immutable) 자료형에는 어떤 것이 있는가?

2.가변(Mutable) 자료형에는 어떤 것이 있는가?