



Assignment

Assignment

이 글을 통해 배우는 개념

- assignment
- mutable & immutable
- argument & parameter
- call by assignment

글을 읽고 바라는 정도

- 나중에 코딩을 하다가 '아.. 저번에 걔가 말한게 이거구나' 하고 환기

assignment

들어가기

- 나는 아래 코드를 설명할 수 있다.
 - 다음으로 넘어가시거나, 애가 설명 잘하나 검사하시면 됩니다

```
# 각각의 번호에 해당하는 코드들이 무얼하는 걸까요? 왜 4번의 결과가 나왔을까요?
# 1.
a = [1, 2, 3]

# 2.
b = a

# 3.
b[0] = 4

# 4. OUTPUT
print(a) # [4, 2, 3]
```

▼ 답안

1. [1, 2, 3] 이라는 리스트를 a에 대입(할당)을 했습니다.
2. a라는 변수를 b에 대입(할당)을 했습니다.
3. 4라는 값을 (b의 0번째 인덱스 / b리스트의 첫번째 원소)에 대입(할당)을 했습니다.
4. a를 출력하였고 [4, 2, 3] 출력이 되었습니다.

Assignment

- 할당, 대입, 참조 등으로 말할 수 있습니다.

- `a = [1, 2, 3]` 에서 리스트 객체 `[1, 2, 3]` 의 이름을 지어주는 행위입니다. `[1, 2, 3]` 이라는 리스트를 추후에 사용하기 위하여 컴퓨터가 값을 찾을 수 있게 이름을 붙혀주고 변수 `a`에 저장하는 것입니다.
- `id()` 함수를 통해서 해당 변수의 위치를 알 수 있습니다.

```
a = [1, 2, 3]
id(a)

>>>
4532767808

---
b = a

# a와 b의 메모리 주소가 같음
id(a) == id(b)
>>>

True
```

위의 코드를 보시면 `a`와 `b`의 주소값이 같음을 확인할 수 있습니다. `[1, 2, 3]`이라는 리스트를 `a`라고도 부르고 `b`라고도 부르는 것을 의미합니다.

왜 `[4, 2, 3]`이 출력이 되었는가?

- `b`와 `a`에 할당된 list, `[1, 2, 3]`이 `mutable` 한 type이기 때문입니다.
 - ▼ ※ mutable

주소(`id()`)는 그대로하고 값만 변경할 수 있는 객체. 가변 객체.

ex) 리스트, 딕셔너리, numpy의 배열

↔ immutable : 고정된 값을 갖는 객체. 새 값을 저장하려면 새 객체를 만들어야함.

ex) 숫자, 문자열, 튜플

파이썬 공식문서 FAQ, 리스트 y를 변경할 때 리스트 x도 변경되는 이유는 무엇입니까?

정리를 해보겠습니다.

1. 새로운 변수를 만들어 할당을 하는 과정은 **단순히 객체를 가르키는 이름을 만들어주는 과정**이다.
 - 파이썬은 모든 것이 `객체` (object)입니다.
2. 위의 코드는 `[1, 2, 3]` 이라는 리스트 객체를 `a`라고도 부르고 `b`라고도 부르겠다고 해준 것 입니다.
3. 따라서 `b`를 바꾸면 `a`도 바뀌게 되는 것입니다.
4. `immutable` 한 타입의 객체는 한번 이름을 지으면 값을 변경할 수 없고 무조건 **새로운 객체**를 만들어야 합니다.

확인

copy

저는 한 객체를 다루는 여러 이름을 만드는 게 아니라 객체를 복사하고 싶습니다!

```
import copy as cp
```

```
# 참조 (객체의 다른 이름 짓기)
a = [1,2,3]
b = a
print(id(a) is id(b)) # True

# 복사 (객체의 값을 복사해서 새로운 객체 만들기)
b = list(a)
print(id(a) is id(b)) # False
b = a[:]
print(id(a) is id(b)) # False
b = a.copy()
print(id(a) is id(b)) # False
b = cp.deepcopy(a)
print(id(a) is id(b)) # False
```

▼ 더 알아보기: copy와 deepcopy의 차이

copy면 독립적인가?

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

b[0][0] = 7
print(a)
```

copy와 deepcopy

copy는 객체에 포함된 레퍼런스가 가르키는 값을 복사하지는 않음 (얕은 복사, shallow copy)

```
#import copy 위에서 이미함
x = [[1,2,3],[4,5,6],[7,8,9]]
def is_same_id(a, b):
    if id(a) == id(b):
        print('메모리 주소(id)가 같습니다.')
    else:
        print('메모리 주소(id)가 다릅니다.')
```

```
a = x
is_same_id(x, a)
is_same_id(x[0], a[0])
>>>
```

메모리 주소(id)가 같습니다.
메모리 주소(id)가 같습니다.

```
b = cp.copy(x) # x.copy(), x[:]
is_same_id(x, b)
is_same_id(x[0], b[0])
>>>
```

메모리 주소(id)가 다릅니다.
메모리 주소(id)가 같습니다.

```
c = cp.deepcopy(x)
is_same_id(x, c)
```

```
is_same_id(x[0], c[0])
>>

메모리 주소(id)가 다릅니다.
메모리 주소(id)가 다릅니다.
```

정리

할당 : 이름을 지어줌

mutable : 값이 변할 수 있는 객체 타입 (list, dict 등) <-> **immutable**

복사(copy) : 새로운 객체를 만들어 이름을 지음 <=> **참조** (기존에 있는 객체의 다른 이름을 추가)

call by assignment

개요

우리는 mutable한 객체와 immutable한 객체의 차이에 대해서 배웠습니다. 그렇다면 함수에서는 어떨까요?

다음은 리스트를 매개변수로 받아 리스트를 역순으로 출력하는 예제입니다. 다음 질문에 대답해보아요

```
def reverse(var: list):
    new_list = var
    while new_list:
        print(new_list[-1])
        new_list.pop()

a = [1,2,3]
reverse(a)
>>>

3
2
1
```

1. reverse() 함수의 인자와 매개변수는 무엇인가요?
2. new_list = var은 무엇을 의미하나요?
3. reverse() 함수가 출력된 후 a 변수엔 어떤 변화가 있을까요?

▼ **print(a)** 실행

```
>>>

[]
```

위의 함수에서 함수에 전달되는 값인 리스트 a를 **인자** (argument), 함수 정의에 나타나 있는 이름인 var을 **매개변수** (parameter)라고 말을 합니다.

인자를 매개변수로 받는 과정은 var = a 처럼 a가 가르키는 객체를 var이라고도 부르겠다고 하는 과정입니다.

위 과정을 인자가 대입으로 전달 된다고 하고, **call by assignment** 혹은 **call by object reference** 라고 부릅니다.

reverse 함수를 짰 사람은 new_list를 통해서 새로운 리스트를 만들어 기존의 리스트를 변경시키지 않으려고 시도했습니다. 하지만, `new_list = var` 은 단순히 var을 new_list 라고도 부른다는 뜻이기 때문에 함수가 종료되면 a의 값은 변경이 되어있습니다.

immutable한 자료형의 경우, 객체의 값을 변경할 수 없기 때문에 위와 같은 문제가 발생하지 않습니다.

퀴즈

입력받는 list를 안 바꾸고 reverse를 재구현 해봅시다!

조건


`new_list = var` 만 바꿀것.

▼ 답안

```
def reverse(var: list):
    new_list = list(var) #var.copy(), var[:]
    while new_list:
        print(new_list[-1])
        new_list.pop()

a = [1,2,3]
reverse(a)
```

▼ 더 알아보기: mutable한 default parameter의 문제점

 [Copy of Better Way 24 \(김태균\)](#)

정리

- `argument` 는 함수에게 전달해주는 값, `parameter` 는 함수가 정의한, `argument`를 대입해줄 변수
- 함수에서 인자가 `대입` (할당)으로 전달됨.

레퍼런스

파이썬 공식문서 FAQ

Programming FAQ - Python 3.9.7 documentation

Yes. The pdb module is a simple but adequate console-mode debugger for Python. It is part of the standard Python library, and is . You can also write your own debugger by using the code for pdb as an example.

 <https://docs.python.org/ko/3/faq/programming.html#why-did-changing-list-y-also-change-list-x>

- 리스트 <y'를 변경할 때 리스트 <x'도 변경되는 이유는 무엇입니까?
- 출력 매개변수가 있는 함수를 작성하려면 어떻게 해야 합니까 (참조에 의한 호출)?
- 파이썬에서 객체를 어떻게 복사합니까?


개인 레포

datacamp_data_science/assignment.ipynb at master · cozytk/datacamp_data_science

Contribute to cozytk/datacamp_data_science development by creating an account on GitHub.

cozytk/
datacamp_data_science



 https://github.com/cozytk/datacamp_data_science/blob/master/week1/assignment.ipynb

 1 Contributor
 0 Issues
 1 Star
 0 Forks



datacamp_data_science/default_parameter_on_func.ipynb at master · cozytk/datacamp_data_science

Contribute to cozytk/datacamp_data_science development by creating an account on GitHub.

cozytk/
datacamp_data_science

 https://github.com/cozytk/datacamp_data_science/blob/master/week2/default_parameter_on_func.ipynb

 1 Contributor
 0 Issues
 1 Star
 0 Forks