Variables and Types

Contents

- Variable
- Data Types
- Dynamic Typing
- Mutable & Immutable
- Shallow copy & Deep copy

Variable

- 변수(variable) : 데이터를 저장할 수 있는 메모리 공간에 붙여진 이름
  - 동적 타입: 파이썬은 동적 타입 언어로, 변수의 타입을 미리 선언할 필요없이 변수에 할당되는 값에 따라 자동으로 타입이 결정
  - 변수 선언과 할당: 변수를 선언할 때 특정 타입을 명시적으로 지정할 필요가 없으며, 변수는 값을 할당함으로써 생성
    - 예를 들어, x = 10과 같이 변수 x에 정수 10을 할당하여 변수를 생성
  - 식별자와 네이밍 규칙: 변수는 식별자(Identifier)로 사용
    - 변수 이름은 문자로 시작해야 하며, 문자, 숫자, 밑줄(_)을 사용 사용
    - 대소문자를 구분하며, 예약어를 변수 이름으로 사용할 수 없음
  - 스코프: 파이썬은 변수의 스코프(scope)를 지원
    - 스코프는 변수가 접근 가능한 범위를 나타내며, 변수의 선언 위치에 따라 전역 스코프와 지역 스코프가 있음
  - 메모리 관리: 파이썬은 변수와 객체의 메모리 관리를 자동으로 처리
    - 변수가 생성될 때 해당 값에 대한 메모리 공간이 할당되며, 변수가 더 이상 필요하지 않을 때 자동으로 메모리에서 해제(garbage collection)

Data Types

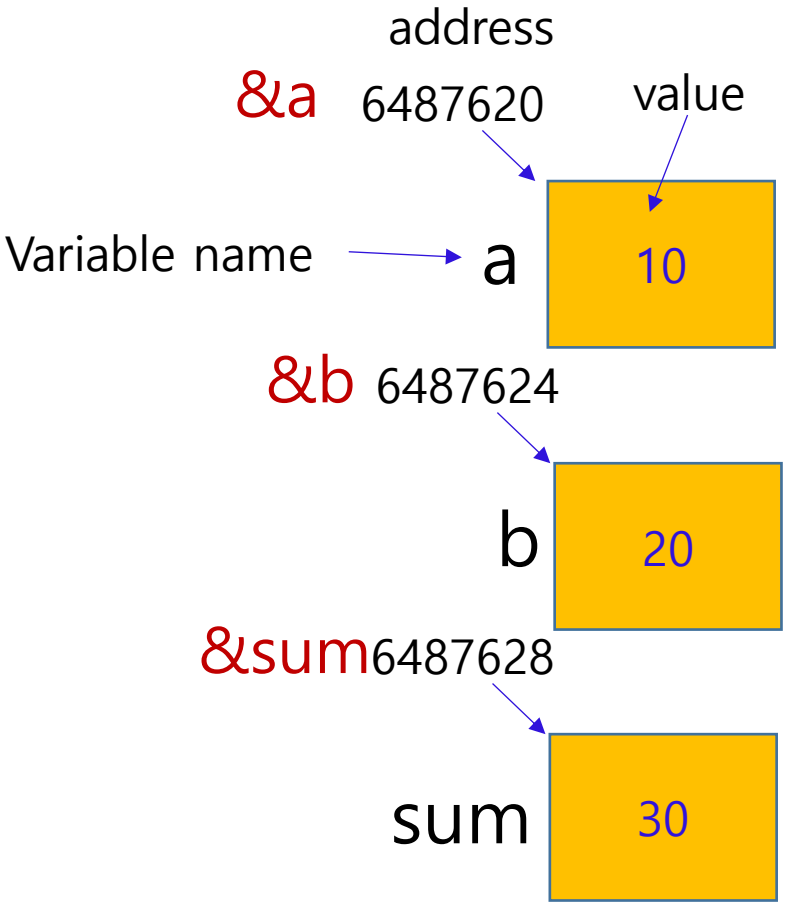
- 숫자(Number) 타입: 숫자를 나타내는 데이터 타입으로, 정수, 실수 등이 포함
  - int: 정수를 나타내는 타입 (예: 10, -5, 0)
  - float: 실수를 나타내는 타입(예: 3.14, -2.5, 0.0)
- 문자열(String) 타입: 문자들의 시퀀스를 나타내는 데이터 타입
  - 예: "Hello", 'Python', "123"

Data Types(Data Structure)

- 리스트(List) 타입: 여러 개의 요소들을 순서대로 저장하는 데이터 타입
  - 예: [1, 2, 3], ["apple", "banana", "orange"]
- 튜플(Tuple) 타입: 여러 개의 요소들을 순서대로 저장하는 데이터 타입으로, 리스트와 유사하지만 변경할 수 없음
  - 예: (1, 2, 3), ("apple", "banana", "orange")
- 딕셔너리(Dictionary) 타입: 키-값(key-value) 쌍으로 데이터를 저장하는 데이터 타입
  - 예: {"name": "John", "age": 25}
- 집합(Set) 타입: 중복되지 않는 요소들로 구성된 데이터 타입
  - 예: {1, 2, 3}, {"apple", "banana", "orange"}

변수(Variable) - C

- **C 언어에서 변수는 데이터 타입 사용하여 변수 선언을 해야함**

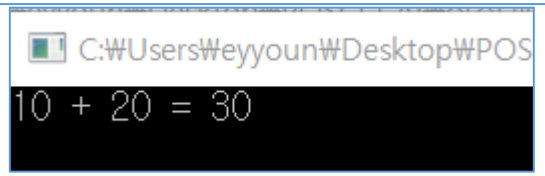



```c
// C
#include <stdio.h>
int main(void)
{
 int a=10, b=20, sum=0;
 sum = a + b;
 printf("%d + %d = %d\n", a, b, sum);
 return 0;
}
```

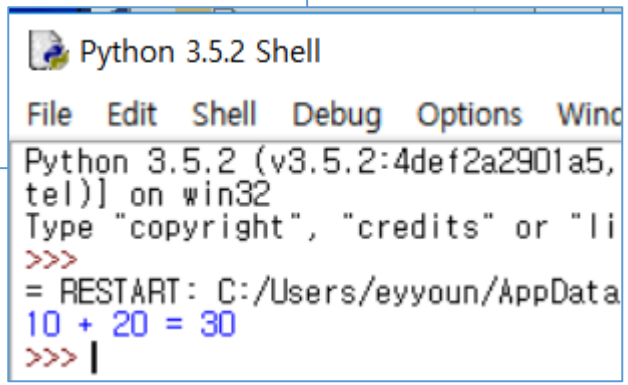
변수 – C & Python

- Python에서 변수는 데이터 타입 생략하고 변수 선언 가능

```c
// C
#include <stdio.h>
int main(void)
{
 int a=10, b=20, sum=0;
 sum = a + b;
 printf("%d + %d = %d\n", a, b, sum);
 return 0;
}
```




```python
# Python
a=10
b=20
sum = a + b
print(a, "+", b, "=", sum)
```



Dynamic Typing

- 동적 타이핑 (dynamic typing)

  런타임에 구문을 통해 Data Type을 설정

```
In [1]:
a = 1;                  print(id(a))
a = 1.5;                print(id(a))
a = 'a';                print(id(a))
a = "hello world!";     print(id(a))
```

```
1856682448
2331000664688
2330963603048
2331002601200
```

변수의 활용

파이썬에서 변수의 데이터 형식은 값을 넣는 순간마다 변경될 수 있는 유연한 구조

```
>>> a = 10
>>> type(a)
<class 'int'>

>>> a = 10.5
>>> type(a)
<class 'float'>
```

변수에는 다른 변수의 값도 저장 가능

```
>>> width = 10
>>> height = 20
>>> area = width * height
>>> print(area)
200
```

변수의 활용

문자열, 실수 저장

```
>>> s = '안녕하세요?'
>>> print(s)
안녕하세요?

>>> pi = 3.141592
>>> print(pi)
3.141592
```

변수 실습

변수 x 와 변수 y의 값을 서로 바꾸는 프로그램을 작성

다음과 같은 프로그램으로는 변수의 값을 교환할 수 없음

```
a = 10          #(1)
b = 20          #(2)
a = b           #(3)
b = a           #(4)
```

변수 실습

해결 방법 1

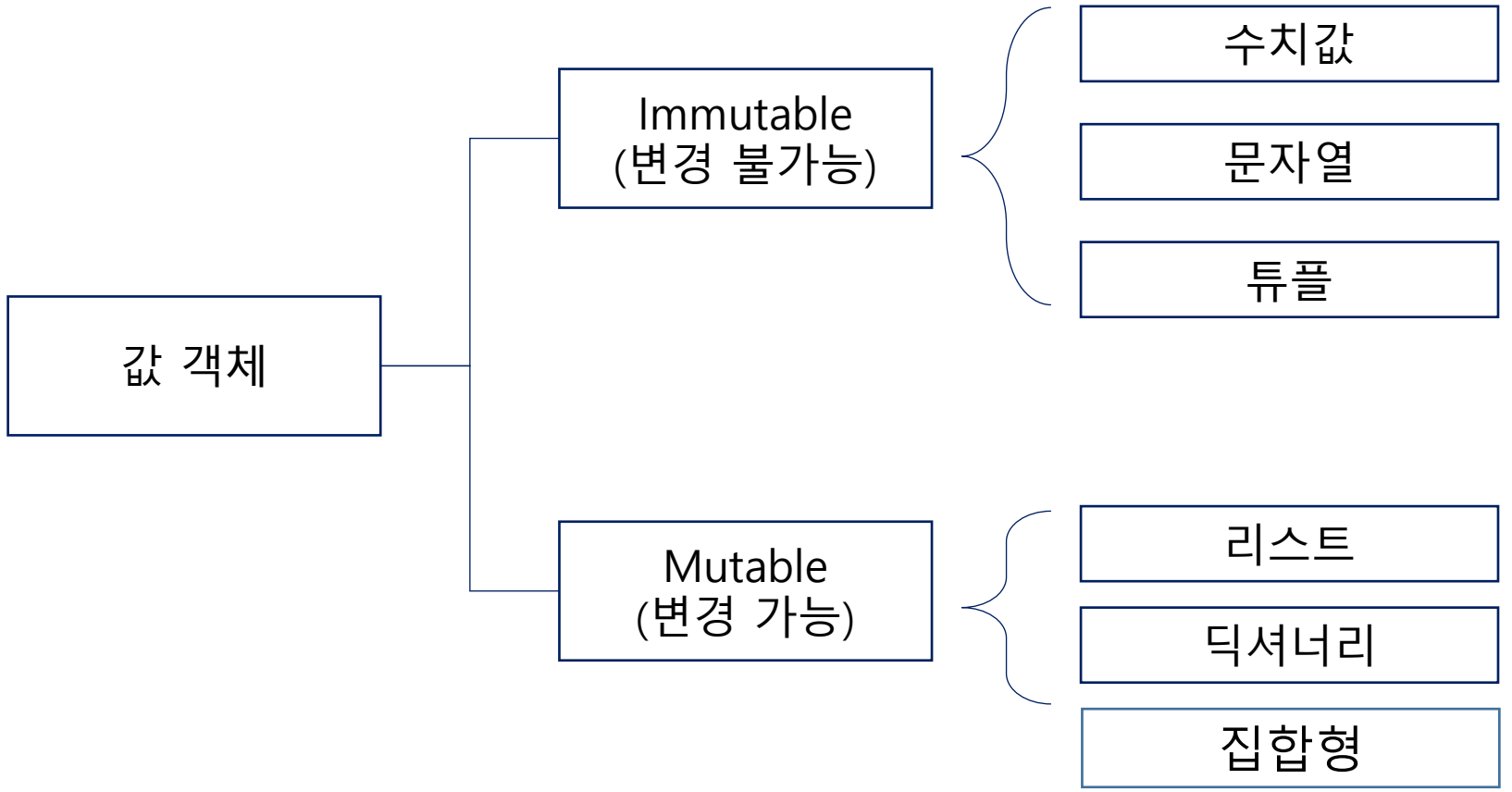
```
x = 90
y = 95
print(x, y) # 90 95

temp= x
x = y
y= temp

print(x, y) # 95 90
```

해결 방법 2

```
>>> a = 10
>>> b = 20
>>> a,b = b,a
>>> print (a,b)
20 10
```

파이썬 객체

- 파이썬은 모두 객체이므로 변경여부 관리가 중요
- 객체가 생성되고 함수 파라미터 등으로 전달될 때에도 변경이 가능한 객체와 불가능한 객체를 동일한 방식으로 관리

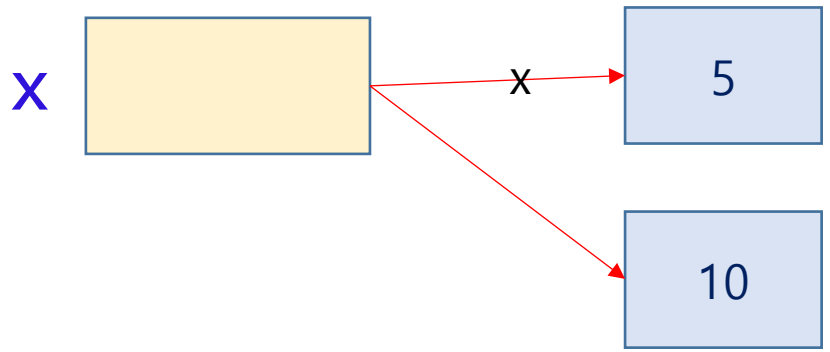


Mutable & Immutable

- Mutable(가변)과 Immutable(불변)은 객체의 상태를 변경할 수 있는지 여부에 따라 분류
- Immutable 객체는 한 번 생성되면 그 값을 변경할 수 없으며, Mutable 객체는 값을 변경할 수 있음

- Immutable 객체의 예시로는 정수(int), 부동소수점(float), 불리언(bool), 문자열(str), 튜플(tuple) 등이 있음
  - 정수형 변수를 생성하고 그 값을 변경하려고 하면 새로운 정수 객체가 생성되고 변수가 이를 참조

```
x = 5
print(x)  # 출력: 5

x = 10
print(x)  # 출력: 10
```



Mutable & Immutable

- Mutable 객체의 예시로는 리스트(list), 딕셔너리(dict), 집합(set) 등이 있음
- 이러한 객체들은 값을 변경할 수 있으며, 변수가 해당 객체를 직접 참조
- 객체의 내용이 변경되면 변수를 통해 즉시 접근할 수 있음

```
my_list = [1, 2, 3]
print(my_list)  # 출력: [1, 2, 3]

my_list.append(4)
print(my_list)  # 출력: [1, 2, 3, 4]

my_list[0] = 0
print(my_list)  # 출력: [0, 2, 3, 4]
```

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

list2 = list1

list2.append(4)

print(list1)  # 출력: [1, 2, 3, 4]
print(list2)  # 출력: [1, 2, 3, 4]
```

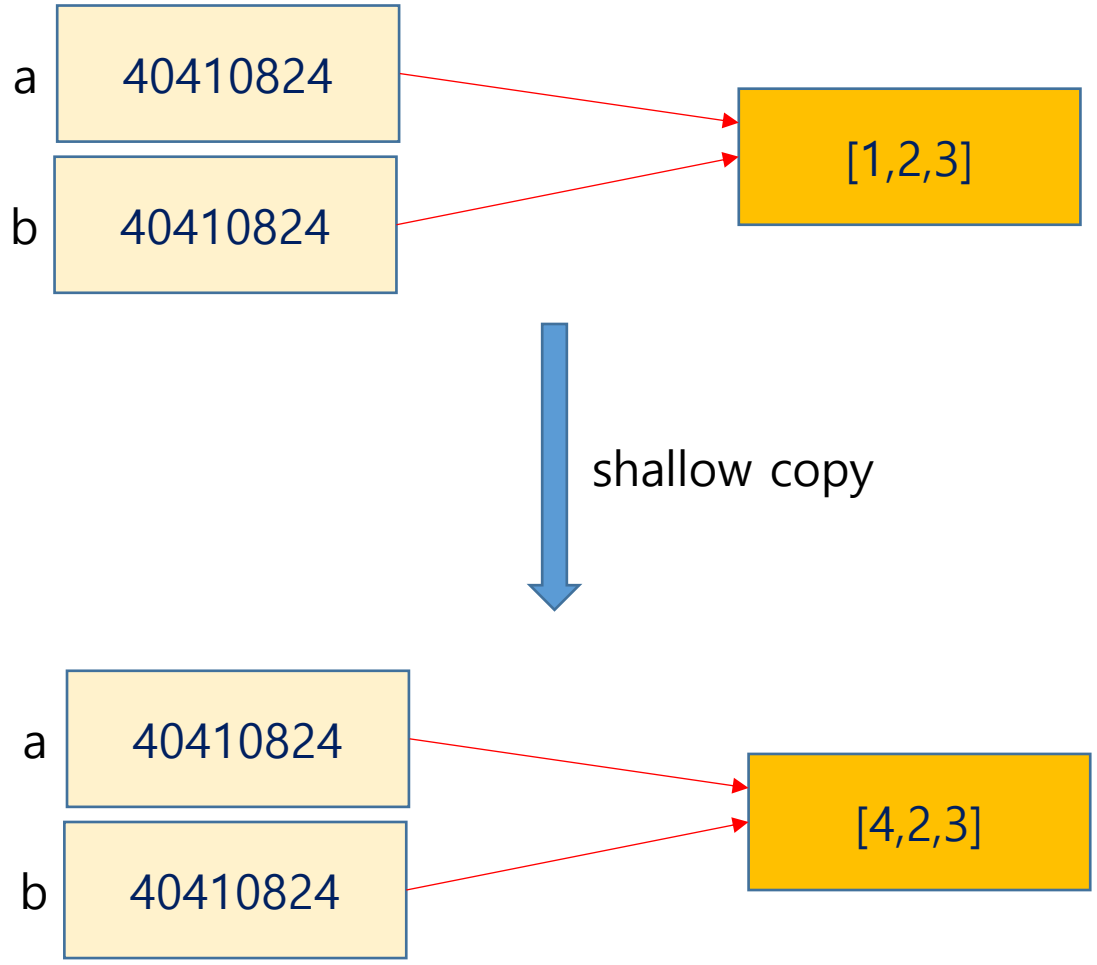
Shallow copy & Deep copy

- Mutable은 주로 리스트, 딕셔너리 타입으로 내부 값인 요소에 추가하는 것이므로 변수나 함수 파라미터로 사용해도 변경(swallow copy 사용)
- Mutable 처리할 경우 처음 값이 변경되지 않으려면 참조만 복사하지 말고 전체 값을 복사해야 별도의 참조가 생겨 다른 값 객체로 인식함(deep copy 이용)

Shallow copy & Deep copy

- **리스트 얕은 복사**

```
>>> a = [1, 2, 3]
>>> b = a
>>> id(a), id(b)
(40410824, 40410824)
>>> a, b
([1, 2, 3], [1, 2, 3])
>>> a[0] = 4
>>> id(a), id(b)
(40410824, 40410824)
>>> a, b
([4, 2, 3], [4, 2, 3])
>>>
```

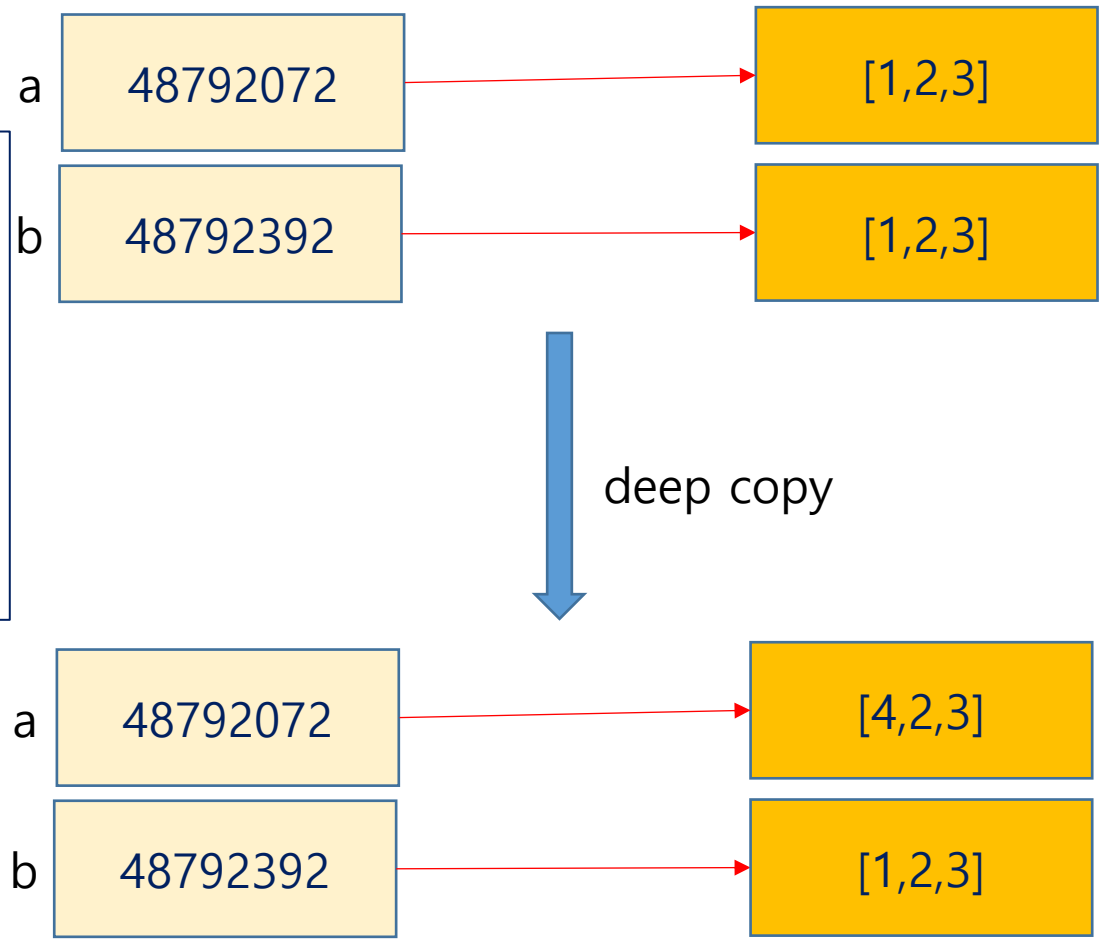


Shallow copy & Deep copy

- **copy 모듈 사용**

```
>>> import copy
>>> a = [1, 2, 3]
>>> b = copy.deepcopy(a)
>>> a[0] = 4
>>> a, b
([4, 2, 3], [1, 2, 3])
>>>
```

copy 모듈의 copy()함수는 주소가 복사되는 얕은 복사(shallow copy)를 하며, deepcopy()함수는 객체를 공유하지 않는 깊은 복사(deep copy)를 함



Shallow copy & Deep copy

- **copy 모듈 사용**

```
>>> import copy
>>> a = [1, 2, 3]
>>> b = copy.deepcopy(a)
>>> a[0] = 4
>>> a, b
([4, 2, 3], [1, 2, 3])
>>>
```

copy 모듈의 copy()함수는 주소가 복사되는 얕은 복사(shallow copy)를 하며, deepcopy()함수는 객체를 공유하지 않는 깊은 복사(deep copy)를 함