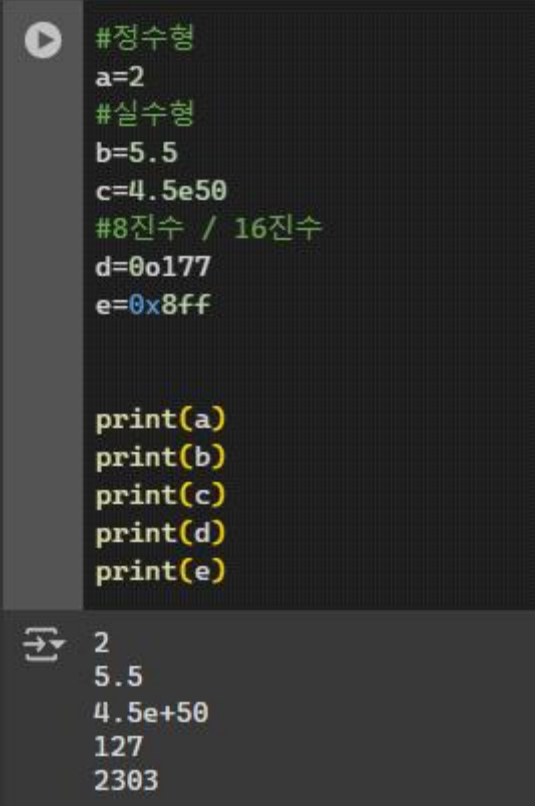


# 디지털 영상처리 연구실 연구보고서

정지우

## <파이썬>

### --숫자형



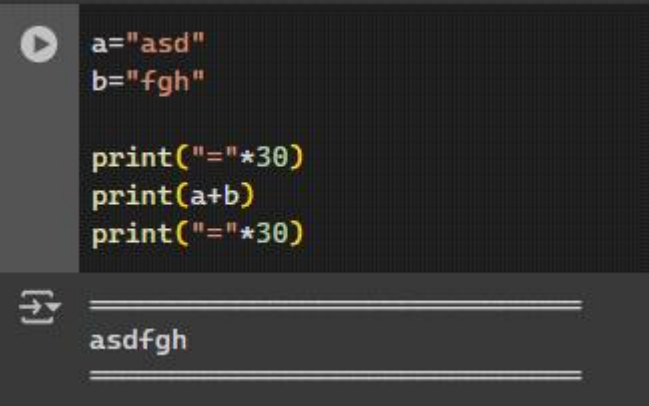
```
#정수형
a=2
#실수형
b=5.5
c=4.5e50
#8진수 / 16진수
d=0o177
e=0x8ff

print(a)
print(b)
print(c)
print(d)
print(e)
```

2  
5.5  
4.5e+50  
127  
2303

C언어와 다르게 자료형이 필요가 없음

### --문자열



```
a="asd"
b="fgh"

print("="*30)
print(a+b)
print("="*30)
```

=====
asdfgh
=====

좌측의 그림과 같이  
문자열을 더하거나 곱할 수 있음

```
a="asd"
b="fgh"

print("="*30)
print(a+b)
print("="*30)

c=a+b
c[3]
```

asdfgh

'f'

문자열을 배열처럼 사용할 수 있음

```
a=39
"에러의 확률은 %d %%입니다." %a
```

'에러의 확률은 39 %입니다.'

문자열 안에 문자는 다음과 같이 %d,%f,%s를 사용하고,  
%기호를 사용하기 위해서는 %%을 사용한다.

```
name = "정지우"; age = 24;
print(f"나의 이름은 {name}이고 나이는 {age}입니다")
```

나의 이름은 정지우이고 나이는 24입니다

또는 위 그림처럼 formating을 하여 바로 변수를 넣을 수도 있다.

## --리스트

```
i=[1,2,3,['a','b','c']]
i[3]
```

['a', 'b', 'c']

C에서는 배열이라고 하지만 파이썬에서는 리스트라고 한다.

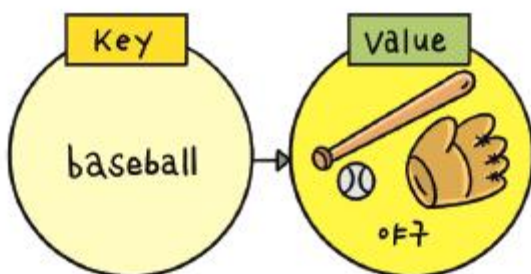
리스트 안에 리스트가 포함가능

```
i=[1,2,3,['a','b','c']]
i[3][0]
```

'a'

이렇게 2차원적으로 호출하면 리스트 안에 리스트값을 얻을 수 있다.

## --딕셔너리



이렇게 한 쌍의 값을 가지고 있는 것을 의미한다.

예를들어

```
>>> dic = {'name': 'pey', 'phone': '010-9999-1234', 'birth': '1118'}
```

이러한 코드가 있을 때, 아래 사진과 같은 의미를 가진다.

key	value
name	pey
phone	010-9999-1234
birth	1118

또한 리스트도 넣을 수 있다.

```
>>> a = {'a': [1, 2, 3]}
```

## --딕셔너리 사용하기

```
>>> dic = {'name': 'pey', 'phone': '010-9999-1234', 'birth': '1118'}
>>> dic['name']
'pey'
>>> dic['phone']
'010-9999-1234'
>>> dic['birth']
'1118'
```

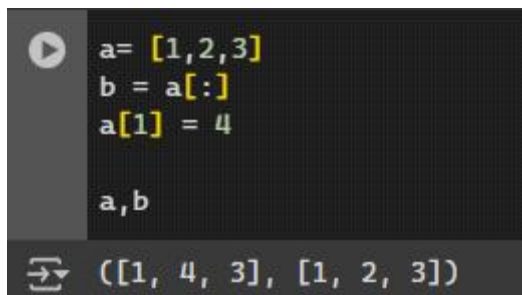
위 그림과 같이 '딕셔너리\_변수\_이름[Key]'를 사용해야 한다.

## --리스트 복사

```
>>> a = [1, 2, 3]
>>> id(a)
4303029896

>>> b = a

>>> id(a)
4303029896
>>> id(b)
4303029896
```

A screenshot of a Python REPL (Jupyter Notebook) showing a list slicing operation. The code defines a list 'a' with values [1, 2, 3], then creates a new list 'b' by slicing 'a' from the beginning to the end (a[:]). Then, the value at index 1 of list 'a' is changed to 4. Finally, the variables 'a' and 'b' are printed. The output shows that 'a' is now [1, 4, 3] and 'b' remains [1, 2, 3], demonstrating that slicing creates a new list object.

```
a= [1,2,3]
b = a[:]
a[1] = 4

a,b
([1, 4, 3], [1, 2, 3])
```

이렇게 : 써서 슬라이싱 하던가,

```
from copy import copy
a = [1, 2, 3]
b = copy(a)
a[1] = 4
|
a,b
```

⇒ ([1, 4, 3], [1, 2, 3])

이렇게 copy를 사용해야한다.

--if문

```
money=True;
if money:|
    print("asd")
    print("asdf sdf")
    print("dfdsf")
```

⇒ asd  
asdf sdf  
dfdsf

```
money=True;
if money:
    print("asd")
    print("asdf sdf")
    print("dfdsf")
```

⇒ File "<ipython-input-82-e994ef6acbe6>", line 5  
print("dfdsf")  
^  
IndentationError: unexpected indent

```
money=True;
if money:
    print("asd")
    print("asdf sdf")
    print("dfdsf")
```

⇒ File "<ipython-input-83-b402662c5558>", line 4  
print("asdf sdf")  
^  
IndentationError: unexpected indent

C언어와 다르게 if문 안에 조건문을 줄 찍어쓰기를 신경 써야한다.

```
1  #include <stdio.h>
2  #include <iostream>
3  using namespace std;
4
5  void main()
6  {
7      int a = 1;
8      if (a == 1)
9      {
10         printf("asdf\n");
11         printf("asdf\n");
12         printf("asdf");
13     }
14 }
15
```

Microsoft Visual S  
asdf  
asdf  
asdf  
C:\Users\WOK\Desktop  
디버깅이 중지될  
하도록 설정합니  
이 창을 닫으려면

--and, or, not

```
>>> money = 2000
>>> card = True
>>> if money >= 3000 or card:
...     print("택시를 타고 가라")
... else:
...     print("걸어가라")
...
택시를 타고 가라
>>>
```

이렇게 if조건에 and, or, not을 사용할 수 있다.

--in, not in

```
>>> pocket = ['paper', 'cellphone', 'money']
>>> if 'money' in pocket:
...     print("택시를 타고 가라")
... else:
...     print("걸어가라")
...
택시를 타고 가라
>>>
```

위 그림과 같이 리스트 안에 존재하는지를 알 수 도 있다.