Lecture 1 : 함수의 인자

1. 인자의 기본값 사용

- 함수의 인자 값이 지정되지 않을 경우, 기본값으로 지정하여 사용 가능
- **인자의 기본값이 지정되는 매개 변수는 뒤에 위치해야 함**

> def **greet(name, message='안녕')** :
> print(name, message)
>
> **greet('철수', '잘 있지~~')**
> **greet('영희')**

In [ ]:
```
#
#
def greet(name, message='안녕') :   # 두번째 인자 값이 지정되지 않을 경우는 기본값
    print(name, message)

greet('철수', '잘 있지~~')          # 첫번째 인자, 두번째 인자 값을 지정하여 함수
greet('영희')                       # 첫번째 인자 값만 지정하여 함수 호출
```

In [ ]:
```
#
#
def greet(name='동양', message) :   # (오류) 인자의 기본값이 지정되는 매개 변수는
    print(name, message)

greet('잘 있지~~')
```

2. 인자의 기본값은 오직 한 번만 사용

- 인자의 기본값은 **첫 함수 호출 시 오직 한 번만 지정됨**

> def **default_arg(a, obj=[])** :
> obj.append(a)
> return obj
>
> **default_arg(1)**
> **default_arg(2)**
> **default_arg(3)**

In [ ]:
```
# 튜플 obj=[]는 첫번째 함수 호출시 한 번만 지정됨
# 연속된 호출 간에 기본값이 공유됨
def default_arg(a, obj=[]) :
    obj.append(a)
    return obj

print(default_arg(1))
```

```
print(default_arg(2))
print(default_arg(3))
```

In [ ]:
```
# 연속된 호출 간에 기본값을 공유하지 않으려면
# (1) 함수 헤더에서 obj=None을 지정
# (2) 함수 호출시마다 매번 새로운 목록 생성

def default_arg(a, obj=None) :
    if obj is None :
        obj = []
    obj.append(a)
    return obj

print(default_arg(1))
print(default_arg(2))
print(default_arg(3))
```

3. 함수 값의 반환

- 파이썬 함수는 여러개의 값을 반환 가능 (튜플 값으로 묶어 반환)

> def **getprime()** :
> return 2, 3, 5, 7
>
> **a, b, c, d = getprime()**
> **prime = getprime()**

In [ ]:
```
def getprime() :
   return 2, 3, 5, 7     # 4개 값을 튜플로 묶어 반환함

a, b, c, d = getprime()  # 4개의 값을 변수 a, b, c, d로 받음
print(a, b, c, d)
```

In [ ]:
```
def getprime() :
   return 2, 3, 5, 7     # 4개 값을 튜플로 묶어 반환함

prime = getprime()  # 4개의 값을 튜플로 받음
print(prime[0], prime[1], prime[2], prime[3])
```

4. 매개변수로 함수 사용

- 파이썬 함수는 매개 변수로 함수 사용 가능

> def **my_apply(func, x, y)** :
> return func(x, y)
>
> def **my_add(a, b)** :
> return a+b
>
> def **my_mult(a, b)** :
> return a*b

> my_apply(my_add, 3, 5)
> my_apply(my_mult, 3, 5)

In [ ]:
```
def my_apply(func, x, y) :
   return func(x, y)

def my_add(a, b) :
   return a+b

def my_mult(a, b) :
   return a*b

print(my_apply(my_add, 3, 5))
print(my_apply(my_mult, 3, 5))
```

In [ ]:
```

```