

Part. 01

파이썬 프로그래밍 기초

# | 함수의 이해 및 사용 (코드의 재사용 1)

FASTCAMPUS  
ONLINE

강사. 변영호

# I 함수란?

입력에 따라 정해진 일을 수행 후 출력값을 생성하는 것



# I 함수의 사용 이유?

반복적인 작업을 계속 동일한 코드로 작성하지 않고,  
함수를 작성하여 코드를 재사용 할 수 있게 함

```
>>> numbers = [1, 2, 3, 4, 5]
>>> _sum = 0
>>> for num in numbers:
...     _sum += num
```

동일한 기능이 필요한 경우  
복사/붙여넣기를 사용

```
>>> numbers = [1, 2, 3, 4, 5, 6, 7]
>>> _sum = 0
>>> for num in numbers:
...     _sum += num
```

# I 함수의 구조

```
def 함수이름(파라미터1, 파라미터2...):
```

실행할 코드 1

실행할 코드 2

return 리턴값

함수이름: 함수를 호출할 때 사용

파라미터: 함수가 받는 입력 값(없을 수도 있음)

리턴값: 함수가 내보내는 출력 값(없을 수도 있음)

```
>>> def summation(start, end):  
>>>     _sum = 0  
>>>     for num in range(start, end+1):  
...         _sum += num  
...     return _sum
```

```
>>> summation(1, 5)
```

```
>>> summation(1, 7)
```

# I 함수 파라미터(입력값)의 이해

함수의 입력 값을 parameter(파라미터)라고 한다.

```
def add(x, y):  
    return x + y
```



각 파라미터는 ,로 구별  
여기서는 x, y 두개의 값을 입력으로 받음

```
add(30, 40)
```



x에 30, y에 40을 전달하여 함수 호출

# I 디폴트 파라미터

함수 파라미터에 기본값 명시 가능

이 경우, 호출시 값을 명시하지 않으면 기본값이 사용 됨

```
def add_all(x, y=1, z = 2, a = 3):  
    return x + y + z + a
```

전부 유효한 add\_all 함수 호출

```
add_all(3)          // 9  
add_all(3, 4)       // 12  
add_all(3, 4, 5)    // 15  
add_all(3, 4, 5, 6) // 18
```

주의!

디폴트 파라미터는 마지막 부터 명시 가능

아래의 경우 SyntaxError 발생

```
def add_all(x, y=1, z, a = 3):  
    return x + y + z + a
```

```
def add_all(x, y=1, z=1, a):  
    return x + y + z + a
```

# I 네임드 파라미터 사용하기

호출 시 파라미터 명을 값과 함께 명시 가능

```
def sub(x, y):  
    return x - y
```



각 파라미터는 ,로 구별  
여기서는 x, y 두개의 값을 입력으로 받음

sub(30, 40)



-10

sub(y = 30, x = 40)



10

sub(x = 30, y = 40)




-10

# I 가변길이 파라미터 사용하기

입력값의 개수가 미리 정해져 있지 않을 때 사용 (파라미터를 tuple로 사용)

```
def 함수이름(*파라미터1):
    실행할 코드 1
    실행할 코드 2

    return 리턴값
```

```
>>> def add_all(*args):
...     total = 0
...     for i in args:  tuple
...         total = total + i
...     return total
```

호출 예시

```
>>> add_all():
>>> add_all(1):
>>> add_all(1, 2):
>>> add_all(1, 2, 3):
```



# I 키워드 파라미터 사용하기

입력값의 개수가 미리 정해져 있지 않고, 네임드 파라미터로 사용 가능  
(파라미터를 dict로 사용)

```
def 함수이름(**파라미터1):
```

실행할 코드 1

실행할 코드 2

return 리턴값

```
>>> def print_all(**kwargs):
...     print(kwargs) → dict
```

호출 예시

```
>>> print_all(a = 1, b = 2): // {a : 1, b : 2}
```

```
>>> print_all(age=10, class=2019): // {age: 10, class: 2019}
```

```
>>> print_all(var=5): // {var: 5}
```

```
>>> print_all(): // {}
```

# I 함수 리턴의 이해

return 키워드는 함수를 종료함과 동시에 호출한 코드에 출력값을 전달하는 역할

```
def add(x, y):  
    if x == 0:  
        return y
```



x가 0인 경우 y 반환  
나머지 경우 x+y 반환

```
    return x + y
```

```
c = add(30, 40)
```



c = 70

```
c = add(0, 40)
```



c = 40

# I 함수 리턴의 이해 2

return 뒤에 반환 값을 명시 하지 않을 경우

```
def add(x, y):  
    if x == 0:  
        return  
  
    print(x+y)
```



x가 0인 경우 함수 종료  
나머지 경우 x+y 출력 후 함수 종료

이때, 함수 마지막에 return이 없는 경우는 return이 생략된 것

add(30, 40)



70출력

add(0, 40)



# I 변수의 scope(유효범위)

```
>>> x = 100
>>> def increment(x):
...     x = x + 1
...     return x
>>> increment(200)
>>> print(x)
```



파라미터를 포함한 함수 내에서 생성된 변수의 scope은 함수의 코드 블록이다.

따라서, 마지막 print(x)문은 첫번째 줄에서 선언된 변수 x의 값 100을 출력한다.

```
>>> def increment(x):
...     x = x + 1
...     return x
>>> increment(200)
>>> print(x)
```



결과는?