

파이썬 프로그래밍

파이썬 함수



한국기술교육대학교
온라인평생교육원

■ 함수 인수 처리

1. 기본 인수 값

- 기본 인수 값
 - 함수를 호출할 때 인수를 넘겨주지 않아도 인수가 기본적으로 가지는 값

```
def incr(a, step=1):  
    return a + step
```

```
b = 1  
b = incr(b)    # 1 증가  
print b
```

```
b = incr(b, 10) # 10 증가  
print b
```

```
2  
12
```

- step에는 기본적으로 인수값 1이 할당됨
- incr(b)에서 b는 a에 들어감
- incr(b, 10) → 10은 step에 들어감
- 기본인자 없으면 error 발생 → 정의한 함수의 인자가 2개기 때문

■ 함수 인수 처리

1. 기본 인수 값

- [주의] 함수 정의를 할 때 기본 값을 지닌 인수 뒤에 일반적인 인수가 올 수 없음

```
def incr(step=1, a):  
    return a + step
```

```
File "<ipython-input-82-67693752f310>", line 1  
def incr(step=1, a):  
SyntaxError: non-default argument follows default argument
```

- step = 1 이라고 정의한 것은 앞 인자에 쓰지 못함
- 기본인자는 맨 마지막 뒤에 와야 함
- 인자가 여러 개일 경우 기본인자 값이 없는 것이 맨 앞 위치

- 함수 정의 시에 여러 개의 기본 인수 값 정의 가능

```
def incr(a, step=1, step2=10):  
    return a + step + step2  
  
print incr(10)
```

21

- 일반인자가 앞으로 위치해야 함

▣ 함수 인수 처리

2. 키워드 인수

- 인수 값 전달 시에 인수 이름과 함께 값을 전달하는 방식을 일컫는다.

```
def area(height, width):  
    return height * width  
  
#순서가 아닌 이름으로 값이 전달  
a = area(height='height string ', width=3)  
print a  
  
b = area(width=20, height=10)  
print b
```

```
height string height string height string  
200
```

- height 는 문자열, width에는 3, 이름으로 값이 들어감
- 이름으로 값이 들어가면 순서 상관 X

■ 함수 인수 처리

2. 키워드 인수

- 함수를 호출 할 때에 키워드 인수는 마지막에 놓여져야 한다.

```
print area(20, width=5)
```

```
100
```

- width=5 → 키워드 인자 : 정의한 문자를 사용하여 가리킴
- 키워드 인자는 기본 인자와 마찬가지로 맨 뒤 위치

- [주의] 함수 호출시에 키워드 인수 뒤에 일반 인수 값이 올 수 없다.

```
area(width=5, 20)
```

```
File "<ipython-input-80-32b5ca4bbf7f>", line 1
    area(width=5, 20)
SyntaxError: non-keyword arg after keyword arg
```

■ 함수 인수 처리

2. 키워드 인수

- 기본 인수값 및 키워드 인수의 혼용

```
def incr(a, step=1, step2=10, step3=100):  
    return a + step + step2 + step3  
  
print incr(10, 2, step2=100)
```

212

- 함수 호출 시에 키워드 인수 뒤에 일반 인수 값이 오면 에러

```
def incr(a, step=1, step2=10, step3=100):  
    return a + step + step2 + step3  
  
print incr(10, 2, step2=100, 200)
```

```
File "<ipython-input-89-3dbdc3e84e66>", line 4  
    print incr(10, 2, step2=100, 200)  
SyntaxError: non-keyword arg after keyword arg
```

- 키워드 인수는 중간에 위치할 수 없음

```
def incr(a, step=1, step2=10, step3=100):  
    return a + step + step2 + step3  
  
print incr(10, 2, step2=100, step3=200)
```

312

■ 함수 인수 처리

3. 가변 인수 리스트

- 함수 정의시에 일반적인 인수 선언 뒤에 *var 형식의 인수로 가변 인수를 선언할 수 있음
 - var에는 함수 호출시 넣어주는 인수 값들 중 일반 인수에 할당되는 값을 제외한 나머지 값들을 지닌 튜플 객체가 할당된다.

```
def varg(a, *arg):  
    print a, arg
```

```
varg(1)  
varg(2,3)  
varg(2,3,4,5,6)
```

```
1 ()  
2 (3,)  
2 (3, 4, 5, 6)
```

- *arg → 가변 인수를 받겠다는 의미
- *arg → 튜플 형태로 반환함
- 콤마(,)가 있어야지 원소가 하나인 튜플을 반영
- arg[0] → 튜플의 첫 번째 원소

- C언어의 printf문과 유사한 형태의 printf 정의 방법

```
def printf(format, *args):  
    print format % args
```

```
printf("I've spent %d days and %d night to do this", 6, 5)
```

```
I've spent 6 days and 5 night to do this
```

- 가변인수를 활용하면 C언어의 printf와 유사하게 사용 가능

■ 함수 인수 처리

4. 튜플 인수와 사전 인수로 함수 호출하기

- 함수 호출에 사용될 인수값들이 튜플에 있다면 "*튜플 변수"를 이용하여 함수 호출이 가능

```
def h(a, b, c):  
    print a,b,c  
  
args = (1, 2, 3)  
h(*args)
```

```
1 2 3
```

- 함수 정의 시 * 사용하게 되면 가변인수
- 함수 호출 시 * 사용하고 뒤에 튜플을 넣으면 튜플 전체 호출 가능

- 함수 호출에 사용될 인수값들이 사전에 있다면 "**사전 변수"를 이용하여 함수 호출이 가능

```
dargs = {'a':1, 'b':2, 'c':3}  
h(**dargs)
```

```
1 2 3
```

- 함수 호출 시 ** 사용하고 뒤에 사전을 넣으면 사전 전체 호출 가능
- 인자의 이름과 식별자가 동일한 키 값을 가져야 함