

파이썬의 함수(function)

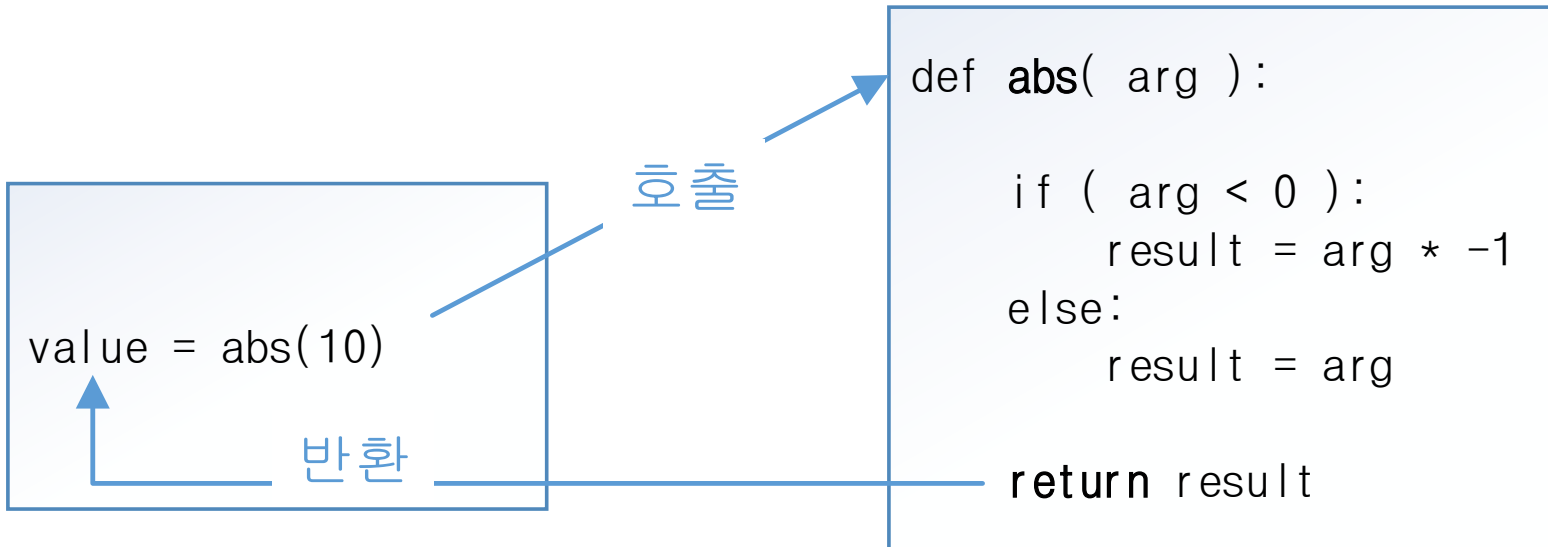
이번 장에서는 파이썬에서 함수 정의와 호출 방법을 알아봅니다.



1. 함수 사용 예
2. 함수 정의하기
3. 매개변수 사용하기
4. 반환하기
5. 지역변수
6. 재귀 함수
7. 함수를 변수에 저장하여 호출하기
8. 중첩 함수
9. 키워드 pass
10. 매개변수 전달 방식



1. 함수 사용 예





2. 함수 정의하기

□ 함수 정의하기

- 함수 정의는 def 키워드로 시작
- 반환값을 주고 싶을 때는 return 키워드를 사용
- 반환값이 없을 때는 return 키워드를 사용하지 않음

```
def 함수이름(매개변수 목록):
```

```
    명령
```

```
    ...
```

```
    return 반환값
```

```
# return 은 생략 가능함
```



3. 매개변수 사용하기

□ 매개변수 사용하기

```
>>> def print_hello(text, count):  
    for i in range(count):  
        print(text)
```

```
>>> print_hello( '안녕' , 2)  
안녕  
안녕
```



3. 매개변수 사용하기

□ 기본값 매개변수 (default argument) 사용하기

```
>>> def print_hello(text, count=1):  
    for i in range(count):  
        print(text)
```

매개변수의 기본값을 1로 정의

```
>>> print_hello( '안녕' )
```

안녕

```
>>> print_hello( '안녕' , 2)
```

안녕

안녕

```
>>> print_hello( '바이' , count=2)
```

바이

바이

호출할 때 두 번째 매개변수를 생략하면
기본값 1이 사용됨

매개변수 이름을 지정해서 값을 입력
하는 것도 가능함 (이때, 키워드 매개
변수 라고도 함)



3. 매개변수 사용하기

□ 가변 매개변수 (arbitrary argument) 사용하기

- 입력 개수가 달라질 수 있는 매개변수
- *를 이용하여 정의하면, 튜플 타입의 매개변수
- **를 이용하여 정의하면, 딕셔너리 타입의 매개변수

```
def print_string(*mytext):  
    result = ''  
    for s in mytext:  
        result = result + s  
    return result
```

매개변수 앞에 *를 붙이면 가변
매개변수로 정의됨

```
print_string('파이썬은', '정말', '재미있다')  
'파이썬은정말재미있다'
```



3. 매개변수 사용하기

□ 가변 매개변수 (arbitrary argument) 사용하기

```
>>> def intsum( *ints ):
    sum = 0
    for s in ints:
        sum += s
    return sum
```

```
>>> print(intsum(1, 2, 3))
6
>>> print(intsum(5, 7, 9, 11, 13))
45
```




3. 매개변수 사용하기

□ 가변 매개변수 (arbitrary argument) 사용하기

- 가변 매개변수는 다른 인수의 마지막에 위치해야 함
- 가변 매개변수는 한 개만 있어야 함

```
>>> def intsum( s, *ints )      # 정상
>>> def intsum( *ints, s )     # 에러
>>> def intsum( *ints, *nums ) # 에러
```



3. 매개변수 사용하기

- 가변 매개변수 (arbitrary argument) 사용하기
 - **를 이용하여 정의하면, 딕셔너리 타입의 매개변수

```
>>> def print_age(**persons):  
    for s in persons.keys():  
        print('{0} = {1}'.format(s, persons[s]))
```

```
>>> print_age(kim=22, lee=23, choi=21)  
kim = 22  
lee = 23  
choi = 21
```



4. 반환하기

□ 값을 반환하기

- return 문에 결과 데이터를 전달
- return 문에 아무 값도 넣지 않아도 됨

```
def sum(a, b) :  
    return a+b
```

```
result = sum(2, 3)  
print(result)
```

```
def print_sum(a, b) :  
    print(a+b)  
    return
```

```
result = print_sum(2, 3)  
print(result)
```



4. 반환하기

- 연습문제 : 다음을 실행해보시오.

```
def func_multiple(x, y, z) :  
    return z, y, x  
  
a, b, c = func_multiple(1,2,3)  
print(a, b, c)  
  
d = func_multiple(1,2,3)  
print(type(d))  
print(d)
```



5. 지역변수

□ 지역변수 : 함수 안에서 만든 변수

- 함수 내부에서만 사용되고 함수가 종료하면 사라짐

```
>>> def sum_test() :  
    a = 1  
    print('a:{0}'.format(a))
```

```
>>> a = 0  
>>> sum_test()  
a:1  
>>> a  
0
```

(참고) locals() 함수는 함수 내의 지역변수를 출력해 줌



□ 전역변수의 사용

- 함수 안에서 함수 밖의 변수를 사용할 때 **global** 키워드를 이용한다

```
>>> def sum_test() :  
    global a  
    a = 1  
    print('a:{0}'.format(a))
```

```
>>> a = 0  
>>> sum_test()  
a:1  
>>> a  
1
```



6. 재귀 함수

□ 재귀 함수

- 자기 자신을 호출하는 함수

```
>>> def factorial(n) :  
    if n == 0:  
        return 1  
    elif n > 0 :  
        return factorial(n-1)*n
```

```
>>> factorial(5)  
120
```



7. 함수를 변수에 저장하여 호출하기

□ 예 1

```
>>> def print_hello(s) :  
        print(s)  
>>> p = print_hello  
>>> p(123)  
123  
>>> p('hello')  
hello
```




7. 함수를 변수에 저장하여 호출하기

□ 예 2

```
>>> def plus(a, b) :  
    return a+b
```

```
>>> def minus(a, b) :  
    return a-b
```

```
>>> myfunc=[plus, minus]
```

```
>>> myfunc[0](1, 2)
```

```
3
```

```
>>> myfunc[1](1, 2)
```

```
-1
```

☞ plus 함수 호출

☞ minus 함수 호출



7. 함수를 변수에 저장하여 호출하기

□ 예 3 : 함수를 다른 함수의 매개변수로 사용하기

```
>>> def print_hello() :  
    print('hello, python')  
>>> def greet(s) :  
    s()  
>>> greet(print_hello)  
Hello, python
```





8. 중첩 함수 (nested function)

□ 중첩 함수 : 함수 안에 정의된 함수

- 중첩 함수는 자신이 소속되어 있는 함수의 매개변수를 사용할 수 있음
- 중첩 함수는 자신이 속해있는 함수의 밖에서는 호출할 수 없음

```
>>> def print_hello(s) :
```

```
    def nest1() :
```

```
        print(s)
```

```
        return 0
```

```
    v = nest1()
```

```
    print(v)
```

```
>>> print_hello('hi')
```



9. 키워드 pass

- pass 키워드는 함수의 구현을 나중에 하고 싶을 때 사용

```
def empty_func()  
    pass
```





10. 매개변수 전달 방식

□ 파이썬의 매개변수 전달 방식 : 혼합 방식

- 파이썬은 모든 것이 객체이며 다음 2 종류가 있음
- **불변 객체 (immutable object)** : int, float, complex, tuples, string, bytes 등
- **가변 객체 (mutable object)** : list, dict, set, bytearray 등
- **불변 객체**가 매개변수로 전달될 때는 **call-by-value** 로 전달
- **가변 객체**가 매개변수로 전달될 때는 **call-by-reference** 로 전달

[참고]

- Call-by-value : 실 매개변수의 값이 전달되며, 함수 내에서 형식 매개변수의 값을 변경해도 실 매개변수의 값은 변하지 않음
- Call-by-reference : 실 매개변수의 주소가 전달



10. 매개변수 전달 방식

□ 예 1 : 불변 객체를 전달할 때 (call-by-value)

```
def sum(x): # x는 형식 매개변수  
    x = x + 10
```

```
a = 5
```

```
sum(a) # a는 실 매개변수
```

```
print(a)
```

5

← a의 값은 변경되지 않음



10. 매개변수 전달 방식

□ 예 2 : 가변 객체를 전달할 때 (call-by-reference)

```
def sum(x) :      # x는 형식 매개변수
    x[0] = x[0] + 10

a = [5]           # a는 리스트로 가변 객체
sum(a)          # a는 실 매개변수

print(a[0])
```

15

← a[0]의 값이 변경되었음



10. 매개변수 전달 방식

□ 예 3 :

```
>>> def spam(eggs) :  
    eggs.append(1)  
    eggs = [2, 3]
```

☞ 형식 매개변수 eggs와는 다른 객체 생성

```
>>> ham = [0]  
>>> spam(ham)  
>>> print(ham)  
[0, 1]
```

☞ ham 객체의 주소가 전달되며, 객체 메소드인 append가 호출되어 1이 추가되었음



10. 매개변수 전달 방식

□ 연습문제 1: 다음 출력 결과는 무엇인가?

```
>>> def spam(eggs) :  
    eggs = [2, 3]  
    eggs.append(1)  
    print(eggs)
```

```
>>> ham = [0]  
>>> spam(ham)  
>>> print(ham)
```



10. 매개변수 전달 방식

□ 연습문제 2: 다음 출력 결과는 무엇인가?

```
>>> def spam(eggs) :  
    eggs.append(100)  
    eggs = [2, 3]  
    eggs.append(1)  
    print(eggs)
```

```
>>> ham = [0]  
>>> spam(ham)  
>>> print(ham)
```



- 파이썬의 내장함수 (built-in function)을 조사하시오.

<https://docs.python.org/3/library/functions.html>

