

22. 08 . 09 火 _ 함수 정의 , 인자 값 전달 방법

함수 정의 및 사용 방법 : 함수 정의 방법

함수 정의



함수 호출

```
1 # 함수 정의 방법
2 ~ def 함수이름 (매개변수) :
3     # 함수 호출 시 실행 될 알고리즘 구문
4     # 함수 호출 시 실행 될 알고리즘 구문
5     # 반환 값 - 반환 값은 옵션 (반환 값 생략 가능)
6
7
8 # 함수 호출
9 # - 함수를 호출하기 위해서는
10 #   호출 구문이 선언부 보다 앞서 위치해야 됨
11 함수이름(인자값)
```

- ✓ 매개변수 (Parameter) : 함수 정의 시 선언된 변수를 지칭
Ex) 3번 라인 sum 함수의 argA, argB 변수
- ✓ 인자 값 (Argument) : 함수 호출 시 매개변수에 전달되는 값
Ex) 9번 라인 sum 함수 호출 시 인자 값 2, 3

영진전문대학교 겸정 일본IT과 정영철 교수

```
1 # sum 함수 정의
2 # 두 개의 수를 입력 받아 더한 값을 반환
3 def sum(argA, argB):
4     result = argA + argB
5
6     return result
7
8 # sum 함수 호출 - 인자 값 2, 3
9 print(sum(2, 3)) # 결과 값 5
```

함수의 가장 큰 이점은 **재사용**

특정기능을수행하는 코드를 함수 단위로 묶음

인자 값 전달 방법

1. positional(디폴트) argument
2. keyword(키워드) argument
3. default(디폴트) argument

함수를 정의할 때 생략해도 되는 것

1. 매개변수
2. 리턴 값



return 값 여러 개면 튜플로 반환한다.

return

1. 값 반환
2. 현재 실행 중인 함수를 종료한다

매개 변수와 반환 값(return)은 필요에 따라 생략 가능

```
# 2개의 정수를 입력 받아 합을 반환하는 함수를 작성 하시오.
# 정의

# static int getSumAvg(int argA, int argB) {
#     int    sum = argA + argB;
#     float   avg = sum / 2;

#     return sum, avg;
# }

def getSumAvg(argValue):

    if argValue < 0:
        return

    print("양수")

    return "양수"

print(-1)

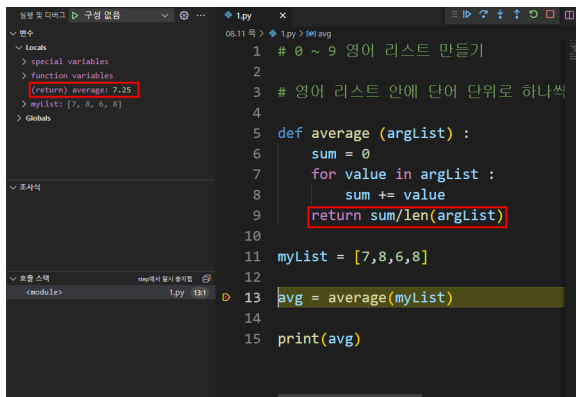
print(2)
```

```
# 매개 변수와 매개 변수가 없는 함수 예제
def printHello() :
    print("hello")

printHello() # hello
```

매개변수와 반환 값

```
def average (argList) :
    sum = 0
    for value in argList :
        sum += value
```



```

return sum/len(argList)

myList = [7,8,6,8]

# average 함수 호출 후 반환 값을 avg 변수에 저장
avg = average(myList)

print(avg)

```

Python 의 경우 함수 반환 값이 두 개 이상일 경우 Tuple 형으로 반환

```

def sumAndMultiply(argA, argB):
    resultSum = argA + argB
    resultMultiply = argA * argB
    # 반환 값이 두 개 이상일 경우 Tuple형으로 반환
    # Tuple은 List와 같음, 단 원소 내용은 변경 불가
    return resultSum, resultMultiply

# sumAndMultiply 함수 호출 후 Tuple형으로 반환 값 획득
result = sumAndMultiply(2, 3)
print(result) # (5, 6)

# 각각의 변수 공간에 할당 해주는 것
sum, multiply = sumAndMultiply(4, 5)
print(sum, multiply) # 9 20

```

```

print(type(sumAndMultiply( 4, 5 )))    =>    <class

```

```

def betty (argA,argB):
    Sum = argA+argB
    Mul = argA*argB
    return Sum , Mul

# 함수 출력
print(betty(2,3)) # 튜플로 반환 -> (5, 6)

print("-----")

Sum , Mul = betty (2,3)
print(Sum , Mul)          # 일반적으로 반환 -> 5 6

```

3. CALL BY VALUE vs CALL BY REFERENCE

함수를 호출할 때 인자값 전달방법

1. CALL BY VALUE - 값이 복사되어 전달
Primitive variables -> int , float , string , boolean
2. CALL BY REFERENCE - (메모리 주소 값)
Reference variables -> Object -> List , Tuple , Dictionary

Call-by-value vs Call-by-reference

```
1 ~ def foo(a):  
2 |     a = a + 2  
3  
4 value = 1  
5  
6 # Call by value  
7 # Primitive variables -> int, float, string, boolean  
8 foo(value)
```

값을 복사해서 사용

Call-by-value vs Call-by-reference

```
12 ~ def bar(a):  
13 |     a.append(1)  
14  
15 value = [2, 3]  
16  
17 # Call by reference  
18 # Reference variables -> Object -> List, Tuple, Dictionary  
19 bar(value)  
20  
21 print(value) # 2, 3, 1
```

용

주소 값만 빌려와 사