22. 08 . 09 火 _ 함수 정의 , 인자 값 전달 방법

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

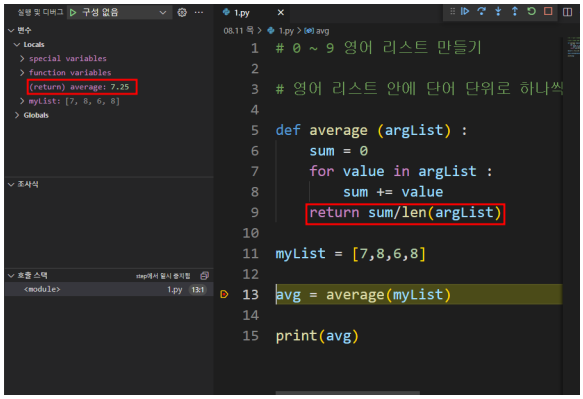
매개 변수와 반환 값(return)은 필요에 따라 생략 가능

```
# 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
```

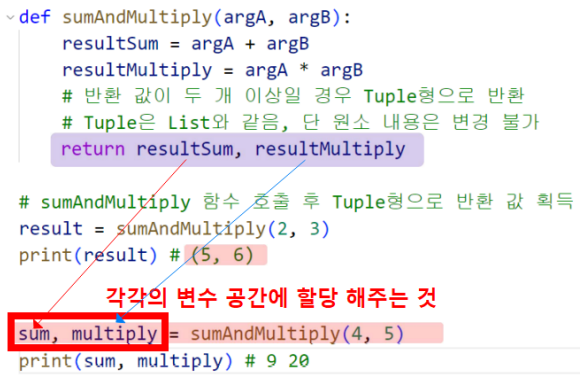
매개변수와 반환 값

```
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




```
print(type(sumAndMultiply( 4, 5 )))    =>     <clas
```

```
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

```python
def foo(a):
    a = a + 2

value = 1

# Call by value
# Primitive variables -> int, float, string, boolean
foo(value)
```

값을 복사해서 사용

Call-by-value vs Call-by-reference

```python
def bar(a):
    a.append(1)

value = [2, 3]

# Call by reference
# Reference variables -> Object -> List, Tuple, Dicitionary
bar(value)

print(value) # 2, 3, 1
```

주소 값만 빌려와 사용