모두를 위한 파이썬 프로그래밍

9주 함수의 이해와 활용
(함수, 변수의 사용범위)

함수 1

파이썬 프로그램의 기본 모습

■ 프로그램의 기본 틀

함수 선언 부분 (funtion define)

변수 선언 부분(variable define)

메인 코드(Main code)

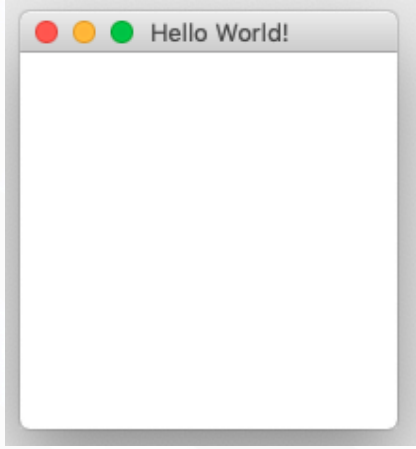
■ 함수의 개념

- ✓ 데이터를 넣으면 가공된 데이터를 돌려주는 상자의 개념
- ✓ 함수의 형식

  함수명()

자주보게 되는 파이썬 프로그램 구조

```
from tkinter import * #임포트문

rootWin = None #전역변수 생성

def GUIMain():
    setupGUI()
    rootWin.mainloop() #GUI 실행

def setupGUI():
    global rootWin #rootWin을 전역변수로 지정

    rootWin = Tk() # 윈도우 오브젝트 만들기
    rootWin.title("Hello World!")

GUIMain()
```



왜 함수가 필요할까?

■ 덧셈, 곱셈의 실행순서가 불규칙적인 프로그램을 작성해 보자

1. 두수를 입력받아 합계를 출력한다.
2. 두수를 입력받아 곱셈을 출력한다.
3. 두수를 입력받아 합계를 출력한다.
4. 두수를 입력받아 합계를 출력한다.
5. 두수를 입력받아 곱셈을 출력한다.
6. 두수를 입력받아 합계를 출력한다.

✓ 입, 출력문을 여러 번 복사해서 작성할 수 있다.

✓ 그러나, 계산 순서가 위와 다르게 진행된다면?

✓ 합계를 100번, 곱셈을 50번 해야 한다면?

→ 순차적인 프로그램에서는 메인함수 하나만 있어도 충분하다. 그러나 프로그램은 비순차적인 경우가 많다.

함수의 기본 형식

def 키워드 사용

- ✓ 인수(Argument), 인자, 파라메터(매개변수, parameter)
- ✓ 반환값(Return Value)

```
def  function_name(parameter)
       실행할 명령  # command

       …
       return 반환값  # value
```

함수의 종류

- ✓ 인자가 없는 함수
- ✓ 인자가 있는 함수
- ✓ 리턴값이 있는 함수
- ✓ 리턴값이 없는 함수

함수의 기본 형식

■ 인수(Argument)와 반환값(return value)이 없고 호출만 하는 경우

- ✓ 인수(Argument) : X
- ✓ 반환값(Return Value) : X

```
def print_title():
    print('''
    ----------------------
       Korea University
    ----------------------''')


print_title()
print_title()
```

```
----------------------
   Korea University
----------------------

----------------------
   Korea University
----------------------
```

함수의 기본 형식

■ 인수는 있으나 반환값(return value)이 없는 경우

- ✓ 인수(Argument) : v1, v2
- ✓ 반환값(Return Value) : X

```
def plus(v1, v2):
    result = 0
    result = v1 + v2
    print('두 수의 합은: ', result)


a = int(input('첫 번째 수를 입력하시오 '))
b = int(input('두 번째 수를 입력하시오 '))
plus(a, b)
```

```
첫 번째 수를 입력하시오 234
두 번째 수를 입력하시오 4324
두 수의 합은: 4558
```

함수의 기본 형식

■ 반환값(return value)이 있는 경우

- ✓ 인수(Argument) : v1, v2
- ✓ 반환값(Return Value) : result

```
def plus(v1, v2):
    result = 0
    result = v1 + v2
    return result


a = int(input('첫 번째 수를 입력하시오 > '))
b = int(input('두 번째 수를 입력하시오 > '))
hap = plus(a, b)
print('두 수의 합: ', hap)
```

```
첫 번째 수를 입력하시오 123
두 번째 수를 입력하시오 456
두 수의 합: 579
```

함수의 기본 형식

■ 반환값(return value)이 여러 개 있는 경우(변수 → 변수에 반환)

- ✓ 인수(Argument) : v1, v2
- ✓ 반환값(Return Value) : result_list

```
def multi_operator(v1, v2):
    add = v1 + v2
    mul = v1 * v2
    return add, mul

input_a = int(input('첫 번째 수를 입력하시오 > '))
input_b = int(input('두 번째 수를 입력하시오 > '))
result1, result2 = multi_operator(input_a, input_b)
print('두 수의 합: %d,  두 수의 곱: %d' % (result1, result2))
```

```
첫 번째 수를 입력하시오 > 10
두 번째 수를 입력하시오 > 20
두 수의 합: 30,  두 수의 곱: 200
```

함수의 기본 형식

■ 반환값(return value)이 여러 개 있는 경우(변수 → 리스트에 반환)

- ✓ 인수(Argument) : v1, v2
- ✓ 반환값(Return Value) : result_list

```
def multi_operator(v1, v2):
    add = v1 + v2
    mul = v1 * v2
    return add, mul


input_a = int(input('첫 번째 수를 입력하시오 > '))
input_b = int(input('두 번째 수를 입력하시오 > '))
result = multi_operator(input_a, input_b)
# 반환변수가 1개일 경우 리스트로 반환됨
print('두 수의 합: %d,  두 수의 곱: %d' % (result[0], result[1]))
```

```
첫 번째 수를 입력하시오 > 10
두 번째 수를 입력하시오 > 20
두 수의 합: 30,  두 수의 곱: 200
```

함수의 기본 형식

■ 반환값(return value)이 여러 개 있는 경우(list → list 반환)

- ✓ 인수(Argument) : v1, v2
- ✓ 반환값(Return Value) : result_list

```
def multi_operator(v1, v2):
    result_list = []
    add = v1 + v2
    mul = v1 * v2
    result_list.append(add)
    result_list.append(mul)
    return result_list


input_a = int(input('첫 번째 수를 입력하시오 > '))
input_b = int(input('두 번째 수를 입력하시오 > '))
result = multi_operator(input_a, input_b)    # 리스트로 리턴 -> 리스트로 반환됨
print('두 수의 합: %d,  두 수의 곱: %d' % (result[0], result[1]))
```

함수의 기본 형식

- parameter값과 반환값이 여러 개 있는 경우
  - ✓ parameter : v1 … v2
  - ✓

```
def all_add_fuc(*para):
    result = 0
    for num in para:
        result += num

    return result, num


hap = all_add_fuc(10)
print(hap)
hap, num = all_add_fuc(10, 20, 30, 40)
print(hap, num)
```

(10, 10)
100 40

함수의 기본 형식

■ 변수 인식이 안되는 이유

```
1   def plus(v1, v2):
2       result1 = 0
3       result1 = v1 + v2
4       print(result2)   # 변수 인식 불가
5
6
7   def multiply(v1, v2):
8       result2 = 0
9       result2 = v1 + v2
10      print(result3)
11
12  result3 = 999
13  plus(10, 20)
14  multiply(100, 100)
15  print(result1, result2, result3)      # result3을 제외하고 인식불가
```

함수의 기본 형식

■ 프로그램내에서 변수의 활용범위가 정해져 있다.

- ✓ 지역 변수 : 한정된 지역에서만 사용
- ✓ 전역 변수 : 프로그램 전체에서 사용가능

myAdd.py

plus<function>

```
    result1 = 0
    result1 = v1 + v2
    print(result2)
```

multiply<function>

```
    result2 = 0
    result2 = v1 * v2
    print(result3)
```

```
result3 = 999

a = int(input('첫 번째 수를 입력하시오 '))
b = int(input('두 번째 수를 입력하시오 '))
plus(a, b)
print('두 수의 합: ', result1)
```

지역변수(local), 전역변수(global)

using Grobal variable

- ✓ global : 전역변수용으로 생성해서 사용(또는 메인코드의 변수 활용)하겠다고 선언
- ✓ 함수에서 메인코드의 전역변수를 사용하려면 사용하겠다고 선언해 주어야 함

```
def function_a():
    a = 10
    print('function_a :', a)


def function_b():
    global a    # 전역변수 a변수를 사용하겠다고 선언
    a = 20
    print('function_b :', a)


a = 30
function_a()    # 10
function_b()    # 20
function_a()    # 10
print('main a:', a) # 20
```

```
function_a : 10
function_b : 20
function_a : 10
main a: 20
```

함수의 반환값과 매개변수

■ 함수로 입력되는 매개변수가 가변적인 경우

✓ 1개에서 n개의 값이 다양하게 입력되는 경우

```
def cal_sum(*para):
    hap = 0
    for num in para:
        hap += num
    return hap


total = cal_sum(10)
print(total)
total = cal_sum(10, 20, 30, 40)
print(total)
```

연습문제

■ 여러 개의 정수를 한꺼번에 입력받아 합계를 출력하는 프로그램을 완성하시오

- ✓ 조건: 합계는 함수로 구현하도록 한다.
- ✓ 더할 수를 여러 개 입력 하시오 > 1 2 3 4 5
  15

```
def all_add_function(para):
    
input_numbers = input('더할 수를 여러 개 입력 하시오 > ').split()
total = all_add_function(input_numbers)
print(total)
```

Thank you