

---

# 함수 정의와 변수

8주차\_02

한 동 대 학 교  
김 경 미 교수

# 사용자 정의 함수

## no parameter, no return value

---

```
def print_script() :
```

```
    print("Blessed are the poor in spirit,")  
    print("for theirs is the kingdom of heaven.")  
    print(" ")
```

```
print("result of first function call, ")  
print("=" * 25)
```

```
print_script()    # first function call
```

```
def repeat_script() :
```

```
    print_script()  
    print_script()
```

```
print("result of second function call, ")  
print("=" * 25)
```

```
repeat_script()    # second function call
```

```
>>>  
===== RESTART: E:/1_Works/2017Work/KMooC:  
=====  
result of first function call,  
=====  
Blessed are the poor in spirit,  
for theirs is the kingdom of heaven.  
  
result of second function call,  
=====  
Blessed are the poor in spirit,  
for theirs is the kingdom of heaven.  
  
Blessed are the poor in spirit,  
for theirs is the kingdom of heaven.  
  
>>> |
```

# 사용자 정의 함수

## parameters, no return value

---

```
def print_chr(times) :  
    for i in range(times) :  
        print("*")
```

```
print("result of first function call, ")  
print("=" * 25)  
print_chr(5)
```

```
def print_line(t) :  
    print("*" * t)
```

```
print("result of second function call, ")  
print("=" * 25)  
print_line(10)
```

```
>>>  
===== RESTART: E:/1_Works/2017  
=====  
result of first dunction call,  
=====  
*  
*  
*  
*  
*  
result of second function call,  
=====  
*****  
*****  
>>> |
```

# 사용자 정의 함수

## parameters, return value

---

```
def plus(num1, num2) :  
    return num1 + num2
```

```
result_plus = plus(2, 12)  
print("result of plus(2, 12) ")  
print(result_plus)
```

```
def mul(num1, num2) :  
    return num1 * num2
```

```
result_mul = mul(2, 12)  
print("result of mul(2, 12) ")  
print(result_mul)
```


```
>>>  
===== RESTART: E:/1_Works,  
=====  
result of plus(2, 12)  
14  
result of mul(2, 12)  
24  
>>>
```

# Parameters and Arguments 1

---

- 함수 내에서, 인수(argument)는 매개변수(parameter)라는 변수에 일대일로 매치된다

```
def print_twice( something ) :  
    print(something)  
    print(something)
```



A blue box labeled "Parameter" has a line pointing to the word "something" in the function definition.

```
print_twice( 'Love' )
```



A blue box labeled "Argument" has a line pointing to the string "'Love'" in the function call.

# Parameters and Arguments 2

---

```
>>> def print_twice( something ) :  
    print(something)  
    print(something)  
  
>>> print_twice( 'Love is real'
```

```
>>> def print_twice( something ) :  
    print(something)  
    print(something)  
  
>>> print_twice("Love is real")  
Love is real  
Love is real  
>>>
```

```
>>> sentence = 'YOU SHALL LOVE YOUR NEIGHBOR .'  
>>> print_twice(sentence)
```

```
>>> sentence = 'YOU SHALL LOVE YOUR NEIGHBOR .'  
>>> print_twice(sentence)  
YOU SHALL LOVE YOUR NEIGHBOR .  
YOU SHALL LOVE YOUR NEIGHBOR .  
>>>
```

# Parameters and Arguments 3

---

```
>>> def User_pow(number1, number2):
    result = 1
    if number2 == 0:
        result = 1
    else:
        for i in range(number2):
            result = result*number1
    print(result)
```

```
>>> User_pow(2,0)
1
>>> User_pow(2,3)
8
```

```
>>> def User_pow(number1, number2):
    result = 1
    if number2 == 0:
        result = 1
    else:
        for i in range(number2):
            result = result*number1
    print(result)
```

```
>>> User_pow(2,0)
1
>>> User_pow(2,3)
8
>>> |
```

# Parameters and Arguments 4

---

```
>>> def User_Max(number1, number2, number3):
    Max = number1
    if number2 > number1 and number2 > number3:
        Max = number2
    if number3 > number2 and number3 > number1:
        Max = number3

    print(Max)
```

```
>>> User_Max(1,2,3)
3
>>> User_Max(3,2,1)
3
>>> User_Max(1,3,2)
3
```

```
>>> def User_Max(number1, number2, number3):
    Max = number1
    if number2 > number1 and number2 > number3:
        Max = number2
    if number3 > number2 and number3 > number1:
        Max = number3

    print(Max)

>>> User_Max(1,2,3)
3
>>> User_Max(3,2,1)
3
>>> User_Max(1,3,2)
3
>>> |
```



# return statement 1

---

- 함수 바디(body)들은 한 개 이상의 return을 포함할 수 있다
  - 함수 바디의 어느 곳에든 위치할 수 있다
  - return문은 함수 호출의 실행을 종료하고
    - 결과 즉, 반환(return) 키워드 다음에 오는 표현의 값을 호출자(caller)에게 “반환” 한다
    - 만약 반환문에 return 문이 없다면
      - 제어 흐름이 함수 바디의 끝에 도착했을 때 함수가 종료된다
  - return되는 값이 여러 개인 경우, 그 값은 tuple 형식으로 전달된다

# return statement 2

---

```
# define function plus( a, b )
def plus( a, b ) :
    return( a+b )
```

```
i = plus(3, 5)
print("return value of function plus(3,5) = ", i)
```

```
# define function plus_list( listname )
num_list=[1,3,5,7,9]
def plus_list( listname ) :
    sum = 0
    for i in listname :
        sum = sum+i
    print (i, sum)
    return sum, num_list
```

```
pl=plus_list(num_list)    #function call
print("return value of funtion plus_list([1,3,5,7,9]) = ", pl)
```

```
>>>
return value of funtion plus(3,5) = 8
9 25
return value of funtion plus_list([1,3,5,7,9]) = (25, [1, 3, 5, 7, 9])
>>> |
```

# return statement 3

---

```
def fahrenheit(celsius) :  
    # returns the temperature in degrees Fahrenheit  
    return (celsius * 9 / 5) + 32  
  
for t in (22.6, 25.8, -10, 0.0) :  
    print("celcius = ",t, "farenheit = ", fahrenheit(t))
```

```
>>>  
===== RESTART: D:/1_Works/2017Work/  
==아이템 수 = 5  
celcius = 22.6 farenheit = 72.68  
celcius = 25.8 farenheit = 78.44  
celcius = -10 farenheit = 14.0  
celcius = 0.0 farenheit = 32.0  
>>> |
```

# 연습문제 1

---

- 2개의 파라미터를 지정한다
- 지정한 파라미터 두개의 곱을 리턴하는 함수를 작성
- 이 함수를 호출하기 전에,
  - 사용자에게 2개의 숫자를 입력 받는다
  - 위에 만든 함수를 호출한다
  - 결과값을 받아서 출력한다

# 연습문제 1 코드

---

```
def Multi(number1, number2):  
    result = number1 * number2  
    return result
```

# 예제 1

---

```
# define sum(n1, n2, n3) function

def sum(i, j, k) :
    return i + j + k

for i in range(10) :
    print(i, " + ", i+1, " + ", i+3, " = ", sum(i, i+1, i+3))

print(" = " * 30)

a = int(input(" enter first integer = " ))
b = int(input(" enter second integer = " ))
c = int(input(" enter third integer = " ))
print(a, " + " , b, " + " , c, " = " , sum(a, b, c))

print(" = " * 30)
```

# 예제 2

```
# define ex function ; exchange list element
def exchange(listN,i,j) :
    temp = listN[i]
    listN[i] = listN[j]
    listN[j] = temp

nums=[1,3,5,7,9,11]
fr=['apple', 'banana', 'bluberry', 'lemon', 'melon']

print("*"*50)
print(nums)
print(fr)

exchange(nums,0,1)
exchange(fr,0,1)
print("exchange index 0,1 = ", nums)
print(fr)
# continue to..
```

```
# ~~~
exchange(nums,1,4)
exchange(fr,1,4)

print("exchange index 1,4 = ",
nums)
print(fr)

print("*"*50)
```

```
>>>
*****
[1, 3, 5, 7, 9, 11]
['apple', 'banana', 'bluberry', 'lemon', 'melon']
exchange index 0,1 = [3, 1, 5, 7, 9, 11]
['banana', 'apple', 'bluberry', 'lemon', 'melon']
exchange index 1,4 = [3, 9, 5, 7, 1, 11]
['banana', 'melon', 'bluberry', 'lemon', 'apple']
*****
```

# 지역 변수와 전역 변수

---

- **전역 변수(Global Variable)**

- 지금까지 사용한 형태의 변수
- 프로그램 시작 시에 값을 지정하여, 계속 활용 가능하다

- **지역 변수(Local Variable)**

- Variable with local scope
  - 변수가 선언된 함수나 블록 내에서만 사용하는 변수
- Variable with global scope
  - 변수를 함수나 블록 내에서 값을 지정하지만, 어디서나 사용 가능하게 정의
  - `global count`



# 전역 변수(Global Variables)

---

- 파이썬 인터프리터 셸에서 값을 지정하는 변수는 글로벌 변수이다
- 어디에서나 사용 가능한 변수이다

```
>>> count = 10
```

```
>>> count = count + 1
```

```
>>> count
```

```
11
```

# 지역변수(Variables with Local Scope)

```
def fun(count):           # count: local scope
    a = 5                 # a: local scope
    count = count + a
    return count

count = 0                 # count: global scope

print('result = ', fun(count))
print(count)             # count: global scope
```

count

0

Module scope

count

6

a

5

Function fun()

```
>>> def fun(count):
      a=5
      count = count + a
      return count

>>> count = 0
>>> print('result = ', fun(count))
result = 5
>>> print(count)
0
>>>
```

# 전역 변수(Global Variables inside a function)

---

```
def fun(count):           #count: local scope
    global a
    a = 5                 #global a is changed
    count = count + a
    return count
```

```
a = 0                    #count: global scope
print('result = ', fun(count))
```

```
print(a)                 #count: global scope
```

```
>>> def fun(count):           #count: local scope
      global a
      a = 5                 #global a is changed
      count = count + a
      return count
```

```
>>> a = 0
>>> print('result = ', fun(count))
result = 5
>>> print(a)
5
>>> |
```

# 요약

---

- 함수 정의를 연습한다
- 파라미터와 리턴값을 이해한다
- 원하는 함수를 기술하는 과정을 익힌다
- 지역 변수와 전역 변수의 차이점을 이해한다

---

# 감사합니다

8주차\_02 함수 정의와 변수