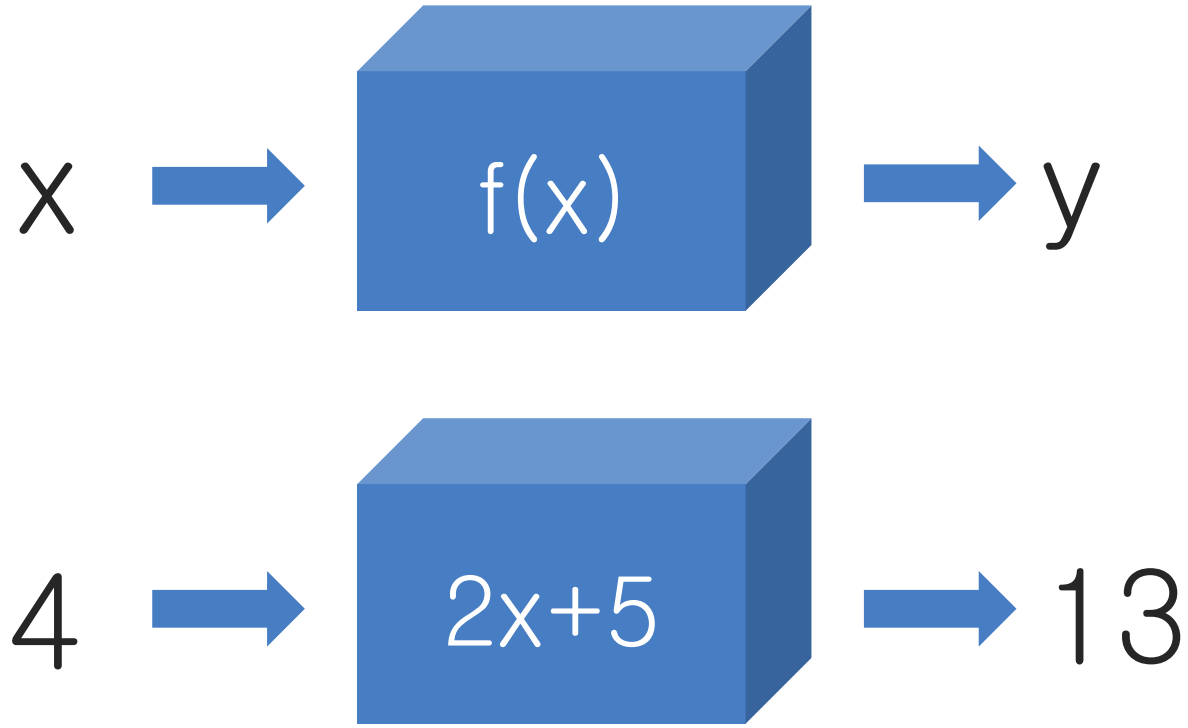
김 미 희 강사


x
f(x)
y
4
2x+5
13

함수란?

- 하나의 특별한 목적의 작업을 수행하기 위해 독립적으로 설계된 코드의 집합

함수를 사용하는 이유

- 반복적인 프로그래밍을 피할 수 있다.
- 모듈화로 인해 전체적인 코드의 가독성이 좋아진다.
- 프로그램에 문제가 발생하거나 기능의 변경이 필요할 때에도 손쉽게 유지보수가 가능하다.

python 함수(function) 구조

define : 정의를 내리다

```
def 함수명(매개변수):
    실행문장
    return 반환변수
```

(colon, 콜론)

들여쓰기 (Tab, Space*4)

함수호출

- 함수명(인수1, 인수2)

```
def 함수명(입력인수):
    실행문장
    return 반환변수
```

```
def number_sum(num1, num2):
    result = num1 + num2
    return result
```

```
number_sum(3, 10)
```

```
13
```

```
number = number_sum(3, 10)
number
```

```
13
```

두 수를 입력 받아서 뺀 결과를 **return**하는 함수를 정의하시오.

```
1  num1 = int(input('첫 번째 정수 입력>> '))
2  num2 = int(input('두 번째 정수 입력>> '))
3  # 빼기 기능을 하는 함수 두 인자값 입력하여 호출
4  result = number_sub(num1,num2)
5  print(result)
```

```
첫 번째 정수 입력>> 10
두 번째 정수 입력>> 3
7
```

두 수를 입력 받아서 원하는 연산을 수행하여 결과를 **return**하는 함수를 정의하시오.

```
첫 번째 정수 입력 >> 5
두 번째 정수 입력 >> 3
연산자 입력(+,-) >> +
결과 : 8
```

```
첫 번째 정수 입력 >> 5
두 번째 정수 입력 >> 3
연산자 입력(+,-) >> -
결과 : 2
```

python 함수(function) 예제

두 개의 숫자(**start, end**)를 매개변수로 받아 **start**숫자부터
end 숫자까지 **list**로 만들어 반환하는 **rangeList** 함수를 정의하세요.

```
print(rangeList(1,15))
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
```

```
print(rangeList(3,9))
```

```
[3, 4, 5, 6, 7, 8, 9]
```

매개변수로 받은 정수의 약수를 구하여 그 결과를 리스트로 반환해주는 **divisor( )**함수를 정의하시오.

```
print(divisor(15))
```

```
[1, 3, 5, 15]
```

```
print(divisor(20))
```

```
[1, 2, 4, 5, 10, 20]
```

```
print(divisor(36))
```

```
[1, 2, 3, 4, 6, 9, 12, 18, 36]
```

매개변수로 받은 정수의 약수의 합을 구하여 그 결과를 반환해 주는 **divisorSum( )**함수를 정의하시오.

```
print(divisorSum(8))
```

```
15
```

```
print(divisorSum(5))
```

```
6
```

```
print(divisorSum(10))
```

```
18
```

매개변수로 두 개의 숫자(**start, end**)를 받아 **start**부터 **end**까지의 숫자 중 완전수를 구하여 그 결과를 리스트로 반환하는 **perfectNumbers( )**함수를 정의하시오.

*** 완전수 : 자기 자신을 제외한 약수의 합이 자기 자신이 되는 수**

[114]:

```
print(perfectNum(1, 1000))
```

```
[6, 28, 496]
```

독스트링(docstring)

- 함수의 설명을 작성 **(Shift + <Tab>)**

```
def cal(num1, num2, op):
    """덧셈과 뺄셈을 계산하는 함수"""
    if op=='+':
        return num1+num2
    else:
        return num1-num2
```

```
cal()
```

```
Signature: cal(num1, num2, op)
Docstring: 덧셈과 뺄셈을 계산하는 함수
```

return 키워드는 한번만 사용 가능하다.

```
def add_sub(num1, num2):
    return num1+num2, num1-num2
```

```
add_sub(10, 7)
```

return 키워드는 한번만 사용 가능하다.

```
def add_sub(num1, num2):
    return num1+num2, num1-num2
```

```
result_add, result_sub = add_sub(10,7)
print(result_add)
print(result_sub)
```

```
17
3
```

기본값 설정(default parameters)

```
def power_of_N(num, power=2):
    return num**power
```

```
power_of_N()
```

```
Signature: power_of_N(num, power=2)
```

기본값 설정(default parameters)

```
def power_of_N(num, power=2):
    return num**power
```

```
power_of_N(3)
```

```
9
```

```
power_of_N(3,3)
```

```
27
```

```
power_of_N(3,power=5)
```

```
243
```

가변 매개변수(variable parameters)

- 함수 호출 시 몇 개의 인수가 전달될지 알 수 없다면, 사용자가 직접 매개변수의 개수를 정할 수 있도록 선언

```
def 함수명(*매개변수):
    실행문장
    return 반환변수
```

가변 매개변수(variable parameters)

- 전달된 모든 인수는 튜플(tuple)의 형태로 저장

```
def add(*args):
    print(args)
```

args → arguments

```
add(1,2,3)
```

```
(1, 2, 3)
```

```
add(1,2,3,4,5,6,7,8,9,10)
```

```
(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
```

가변 매개변수를 활용해 모든 숫자를 더해서 반환하는 함수를 작성하시오.

```
def add(*args):
    ?
```

```
add(1,2,3)
```

```
6
```

```
add(1,2,3,4,5,6,7,8,9,10)
```

```
55
```