python

스마트인재개발원
Smart Human Resources Development

손 지 영 강사

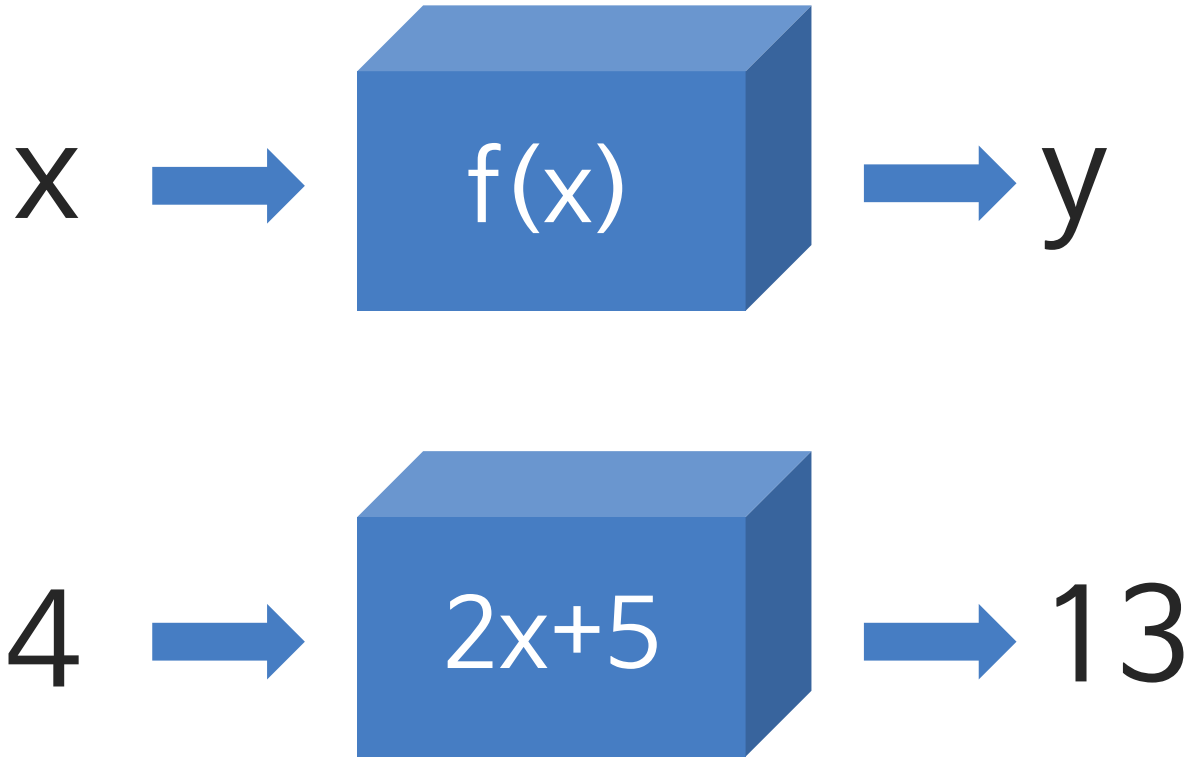
python

학습목표

- 함수에 대해 알 수 있다.
- 함수를 정의하고 활용 할 수 있다.

Smart Media
스마트미디어인재개발원

python 함수(function) 란?



함수란?

- 하나의 특별한 목적의 작업을 수행하기 위해 독립적으로 설계된 코드의 집합

함수를 사용하는 이유

- 반복적인 프로그래밍 피하기
- 모듈화로 인해 전체적인 코드의 가독성 극대화
- 프로그램에 문제가 발생하거나 기능의 변경이 필요할 때에도 손쉽게 유지보수

python 우리가 이미 사용해본 함수

print() : 출력하는 기능

type() : 데이터 타입 반환하는 기능

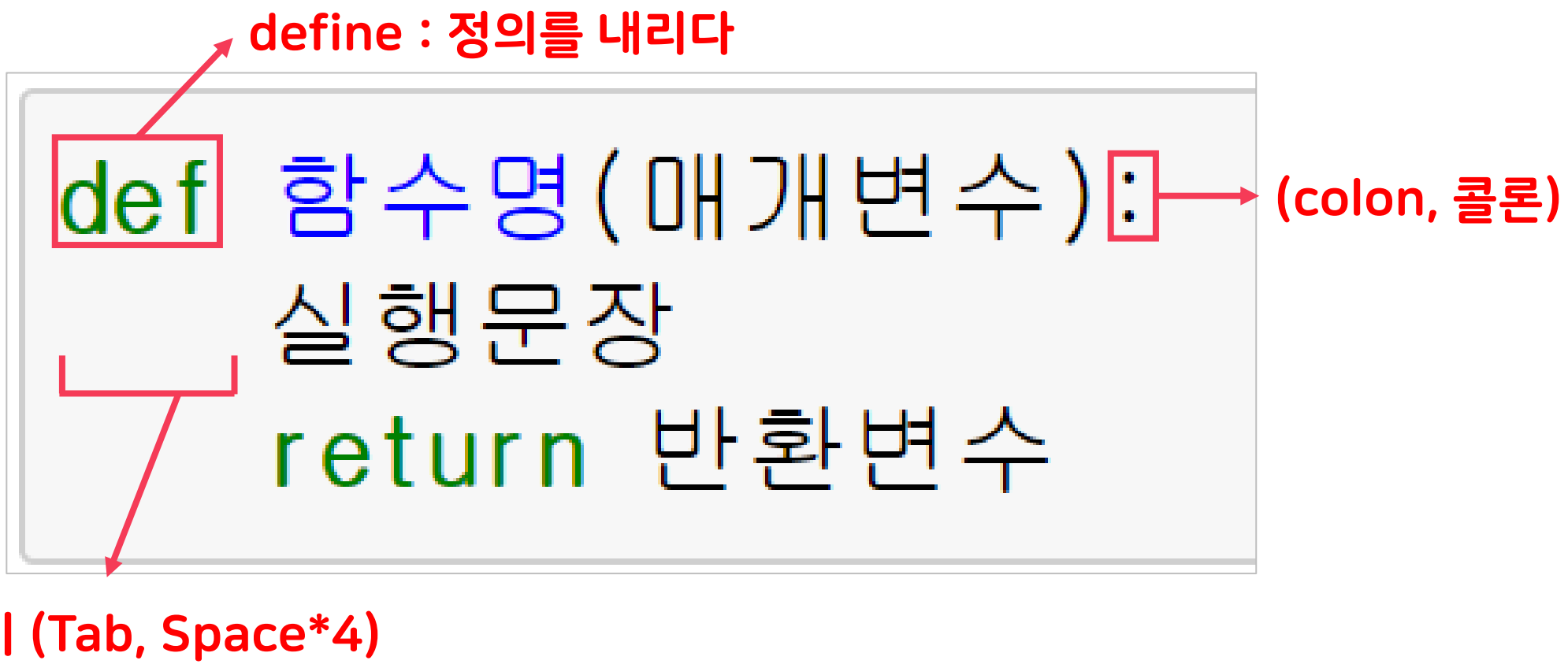
len() : 길이 확인하는 기능

range() : 입력된 범위에서 일련의 숫자열을 생성하는 기능

replace() : 문자열을 대체하는 기능

sort() : 정렬하는 기능

python 함수(function) 구조



python 함수(function) 사용

함수호출

- 함수명(인수1, 인수2)

```
def 함수명(매개변수):
    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 return하는 함수를 정의하시오.

함수 호출

```
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
```

문자열을 입력 받아 'ㅋ'을 제거하고 return하는 함수를 정의하시오.

함수 호출

```
s = input("문자열 입력 >> ")
result = s_replace(s)
result
```

```
문자열 입력 >> ㅋ을 모두 지워주세요ㅋㅋㅋㅋㅋㅋ

'을 모두 지워주세요'
```

숫자 1개를 입력 받을 때 약수를 출력하는 함수를 정의하시오.

함수 호출

```
divisor(10)
```

```
1 2 5 10
```

```
divisor(32)
```

```
1 2 4 8 16 32
```

```
divisor(100)
```

```
1 2 4 5 10 20 25 50 100
```

python 함수(function) 예제4

Smart Media
스마트미디어인재개발원

두 수를 입력 받아서 원하는 연산을 수행하여 결과를 return하는 함수를 정의하시오.

함수 호출

```
num1 = int(input("첫 번째 정수 입력 >> "))
num2 = int(input("두 번째 정수 입력 >> "))
op = input("연산자 입력(+,-) >> ")
result = cal(num1, num2, op)
print("결과 : {}".format(result))
```

결과

```
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
```

독스트링(docstring)

- 함수의 기능 설명 작성
- 정의된 함수 내에서 Shift + Tab 누르면 설명 확인 가능
- 함수 내부 첫 줄에 개행을 포함한 긴 줄 문자열기호(""" """)로 작성

```
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
```

python 함수(function) 독스트링(docstring) 예제

이름이 입력되면 인사하는 함수를 정의하면서 기능에 대한 설명도 추가해보기

함수 호출

```
greeting('지영')
```

```
'지영 안녕 ~ '
```

shift + tab

```
greeting('지영')
```

```
Signature: greeting(name)
Docstring:
이름이 입력되면 인사를 하는 기능
name type: str
return type : str
```

python 함수(function) 타입 힌트(Type Hints)

데이터(형)에 관한 주석(annotation)을 붙이는 기능

3.5ver typing 모듈 지원
복잡한 타입 어노테이션을
추가할 때 사용하는 모듈

기존 주석 방식

```
def greeting(name): # name : str, return: str
    return name + ' 안녕 ~ '
```

type hinting

```
def greeting(name: str) -> str:
    return name + ' 안녕 ~ '
```

확인

```
print(greeting.__annotations__)
```

```
{'name': <class 'str'>, 'return': <class 'str'>}
```

Return 함수의 반환값은 언제나 하나

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

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

python 함수(function) return 키워드

return 함수의 반환값은 언제나 하나

```
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)
```

```
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)
```

python 함수(function) 가변 매개변수(*) 예제

가변 매개변수를 활용해 모든 숫자를 더해서 반환하는 함수를 작성하시오.
(sum() 내장함수이용)

- sum 함수 사용: `sum((1,1,1))`

함수 호출

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

```
6
```

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

```
55
```

가변 매개변수(variable parameters)

- 딕셔너리 형태로 함수 내부에서 처리하고 싶을 때

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

가변 매개변수(variable parameters)

- 딕셔너리 형태로 함수 내부에서 처리하고 싶을 때

```
def print_map(**kwargs):
    print(kwargs)
    for key,value in kwargs.items():
        print(key,"/",value)
```

kwargs → keyword arguments

```
print_map(하나=1)
```

```
{'하나': 1}
하나 / 1
```

```
print_map(one=1, two=2)
```

```
{'one': 1, 'two': 2}
one / 1
two / 2
```

네임스페이스(namespace)

- 프로그래밍 언어에서 특정한 객체(Object)를 이름(Name)에 따라 구분할 수 있는 범위
- 빌트인 네임스페이스, 전역 네임스페이스, 지역 네임스페이스

```
x = 9 # x는 전역 변수
print('함수 밖',x)
def temp():
    x = 10 # x는 temp의 지역 변수
    print('함수 안:',x)
temp()
```

```
함수 밖 9
함수 안: 10
```

```
# global 키워드
x = 9 # x는 전역 변수
print('함수 밖',x)
def temp():
    global x  # x를 전역 변수로 만듬
    x = 10
    print('함수 안:',x)
temp()
print('함수 밖:', x)
```

수고하셨습니다 !

Smart Media
스마트미디어인재개발원