Basic Python

Lecture 7. 함수

BAE SEKANG(배세강)

Contents

Functions

♣ 함수 정의

- 호출을 할 때마다 내부의 코드을 실행하는 한 단위
- 함수명은 함수의 목적이 직관적으로 보이도록 짓는 것이 좋다.

```
def add(a, b):
    return a + b

a = add(3, 5)
print(f"The sum is: {a}")
```

Functions

♣ 함수 호출

```
def my_function(x):
    return x * 2

print(my_function(5))

def words_print(x):
    print(x)

# print()도 함수이다.
print(7)
```

Functions

[Practice 1]

두 숫자 중 더 큰 수를 반환하는 함수를 구현한다.

[Practice 2]

문자열을 뒤집어 반환하는 함수를 구현한다.

Functions

♣ pass

- 함수 내부를 비워놓으면 오류가 발생
- pass는 실행 시 그냥 지나가는 역할

```
# 아직 내부의 기능을 구현하지 않은 함수이다.
def temp():
    pass
```

Functions

♣ 반환값

- return 키워드를 활용
- print와 return의 차이

```
def func1():
    return "Function 1"

def func2():
    print("Function 2")

def func3():
    print("Function 3-1")
    return "Function 3-2"
```

Arguments

♣ 인자(Argument)와 매개변수(Parameter)

- Argument: 함수를 호출할 때 사용하는 일련의 값들
- Parameter: 함수의 외부에서 받아들이는 값

```
def double(x):      # x가 Parameter
    return x * 2

print(double(10)) # 10이 Argument

def add(a, b):      # a, b가 Parameter
    return a + b

print(add(3, 5))    # 3, 5가 Argument
```

Arguments

♣ 기본 인자(Default arguments)

- 함수 선언 시 기본 인자를 설정할 수 있다.
- Parameter를 받지 않을 때, 기본 인자를 사용한다.

```
def double(x = 0):
    return x * 2

print(double())        # 0

print(double(5))       # 10
```

Arguments

♣ 키워드 인자(Keyword arguments)

- Argument를 넘겨줄 때, 변수명을 지정해서 넘겨준다.

```
def func(name, age):
    print(f"Name: {name}, Age: {age}")

# name과 age의 값을 지정해서 넘긴다.
func(name="Alice", age=30)
```

Functions

[Practice 3]

문자열을 받아, 문자의 길이를 반환하는 함수를 구현한다. 아무 것도 인자로 받지 않을 때는 0을 반환한다.

Arguments

♣ 가변 인자(Arbitrary Arguments) - *args

- 인자의 개수가 정해져 있지 않다. (가변적)
- *을 앞에 붙여서 사용한다. (변수명은 꼭 args가 아니어도 된다.)

```
def nums(*args):
    print(args)
    print(type(args))
    return list(args)

print(nums(1, 2, 3))
```

```
>> (1, 2, 3)
>> <class 'tuple'>

>> [1, 2, 3]
```

Arguments

♣ 가변 인자(Arbitrary Arguments)

- 일반적인 Argument와 함께 사용할 수 있다.

```
def introduce(name, *hobbies):
    print(name)
    print(hobbies)

introduce("Alice", "reading", "music")
```

Arguments

♣ 가변 키워드 인자(Arbitrary Keyword Arguments) - **kwargs

- 키워드 인자의 개수가 정해져 있지 않다. (가변적)
- **을 앞에 붙여서 사용한다. (변수명은 꼭 kwargs가 아니어도 된다.)

```
# kwargs는 내부적으로 {'name': 'Alice', 'age': 25}와 같은 딕셔너리가 된다.

def print_user_info(**kwargs):
    for key, value in kwargs.items():
        print(f"{key}: {value}")

print_user_info(name="Alice", age=25, city="Seoul")
```

Recursions

♣ 재귀 함수

- 함수 내부에서 자기 자신을 호출하는 함수
- Base Case와 Recursive Case가 있다.
- 너무 깊은 재귀 호출은 RecursionError를 발생시킨다.

```
# n!을 재귀적으로 구현한 함수
def factorial(n):
    if n <= 1:
        return 1
    return n * factorial(n - 1)

print(factorial(5))
```

Recursions

[Practice 4]

자연수 n을 입력 받을 때, 피보나치 수열의 n번 째 항을 출력한다.
(피보나치 수열은 1, 1, 2, 3, 5, 8, 13, 21, …이며, 앞선 두 항을 더한 값이 현재 항의 값이 된다.)

[입력 1]
7

[출력 1]
13

Recursions

[Practice 4] – Answer: 재귀함수를 활용한 풀이

```
def fibo(n):
    if n == 1 or n == 2:
        return 1
    return fibo(n-1) + fibo(n-2)

n = int(input())
print(fibo(n))
```

Recursions

[Practice 4] – Answer: 재귀함수를 활용하지 않은 풀이

```
n = int(input())

a, b = 1, 1

for _ in range(n - 1):
    a, b = b, a + b

print(a)
```

THANK YOU!

Any Question?