컴퓨터 및 프로그래밍입문(005) 5주차

한상곤(sangkon@pusan.ac.kr)

4. 함수

목차

학습목표

4.1 함수에 대해 이해하고 그 필요성을 설명할 수 있다.
4.2 내장 함수와 사용자 정의 함수를 이해하고 설명할 수 있다.
4.3 사용자 정의 함수를 `def` 문을 이용하여 정의하고 호출할 수 있다.
4.6 다중 반환문을 사용하여 여러 개의 값을 반환할 수 있다.
4.7 지역변수와 전역변수를 올바르게 사용할 수 있다.
4.4 호출되는 함수에 값을 전달하기 위하여 매개변수를 사용할 수 있다.
4.5 함수의 반환문에 대해 이해하고 그 필요성을 설명할 수 있다.
4.8 순서/키워드 매개변수를 이용하여 효율적으로 값을 전달할 수 있다.
4.9 ~~함수에 전달되는 가변 인자를 처리하는 방법을 익힌다.~~
4.14 파이썬의 다양한 내장 함수를 사용할 수 있다.

4.1 함수의 필요성

- 반복적으로 사용되는 코드 - 덩어리(혹은 블록block) 이라고 함
- 기능에 따라 미리 만들어진 블록은 필요할 때 호출function call함
- 파이썬에서 미리 만들어서 제공하는 함수는 인터프리터에 포함되어 배포되는데 이러한 함수를 내장함수built-in function 라고 함
  - 대표적으로 print()가 있음
- 사용자가 직접 필요한 함수를 만들 수 있음
  - 이러한 함수를 사용자 정의 함수user defined function라고 함

```
def print_star():                          # 별표 출력을 위한 함수 정의
    print('**************************')
print_star()                               # 별표 출력을 위한 함수 호출
```

4.2 함수와 매개변수

```
a = 30 # 전역 변수
b = 50 # 전역변수
def print_sum(a, b): # 매개변수를 가진 함수
    result = a + b    # 지역변수
    print(a, '과', b, '의 합은', result, '입니다.')

print_sum(10, 20)     # 두개의 인자
print_sum(100, 200)
```

4.4 반환값(a)

- 일반적으로 함수 내부는 블랙박스black box라고 가정
- 함수의 내부는 특정한 코드를 가지고 있으며 주어진 일을 수행하고 결과를 반환할 수 있음
- return 키워드를 사용하여 하나 이상의 값을 반환해 줄 수 있음

```
a = 30 # 전역 변수
b = 50 # 전역변수
def sum(a, b): # 매개변수를 가진 함수
    result = a + b    # 지역변수
    return result

print(sum(10, 20))     # 두개의 인자를 활용한 sum 함수 호출
print(sum(100, 200))
```

4.4 반환값(b)

- 두 개 이상의 값을 반환하는 반환문을 다중 반환문multiple return statement이라고 함
- 다중 반환문에서 쉼표로 구분되는 두 개의 값은 튜플 형으로 반환이 이루어짐

```
def get_circumference_and_area(r):
    return 3.14 * r, 3.14 * r * r
print(get_circumference_and_area(12))
```

4.5 전역변수(a)

- 함수 바깥에서 선언되거나 전체 영역에서 사용 가능한 변수

```
def print_sum():
    a = 100
    b = 200
    result = a + b
    print('print_sum() 내부 :', a, '과', b, '의 합은', result, '입니다.')
a = 10
b = 20
print_sum()
result = a + b
print('print_sum() 외부 :', a, '과', b, '의 합은', result, '입니다.')
```

4.5 전역변수(b)

```
def print_sum():
    global a, b             # a, b는 함수외부에서 선언된 a, b를 사용한다.
    a = 100
    b = 200
    result = a + b
    print('print_sum() 내부 :', a, '과', b, '의 합은', result, '입니다.')

a = 10
b = 20
print_sum()
result = a + b
print('print_sum() 외부 :', a, '과', b, '의 합은', result, '입니다.')
```

4.6 함수의 인자 전달 방식(a) - 디폴트 값을 가지는 매개변수

```
def print_star(n):     # 인자를 필요로 함
    for _ in range(n):
        print('**************************')
print_star() # Error

def print_star(n=1):     # 인자를 필요로 함
    for _ in range(n):
        print('**************************')
print_star()
```

4.6 함수의 인자 전달 방식(b) - 키워드

```
def func(shape, width=1, height=1, radius=1):
    if shape == 0 :  # shape 값이 0이면 사각형의 면적을 반환
        return width * height
    if shape == 1  : # shape 값이 1이면 원의 면적을 반환
        return 3.14 * radius ** 2

print("rect area =", func(0, width=10, height=2))
print("circle area =", func(1, radius=5))
```

4.6 함수의 인자 전달 방식(c) - 가변

```
def greet(*names):
    for name in names:
        print('안녕하세요', name, '씨')

greet('홍길동', '양만춘', '이순신') # 인자가 3개
greet('James', 'Thomas') # 인자가 2개
```

4.7 재귀함수

- 재귀함수recursion란 함수 내부에서 자기 자신을 호출하는 함수를 말함
- 절차적 기법으로 해결하기 어려운 문제를 직관적이고 간단하게 해결 가능

```
def factorial(n):       # n!의 재귀적 구현
    if n <= 1 :         # 종료조건이 반드시 필요하다
        return 1
    else :
        return n * factorial(n-1)     # n * (n-1)! 정의에 따른 구현

n = 5
print(f'{n}! = {factorial(n)}')
```

4.8 내장함수

abs()	dict()	help()	min()	setattr()
all()	dir()	hex()	next()	slice()
any()	divmod()	id()	object()	sorted()
ascii()	enumerate()	input()	oct()	staticmethod()
bin()	eval()	int()	open()	str()
bool()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	_import_()
complex()	hasattr()	max()	round()	
delattr()	hash()	memoryview()	set()	