함수

1. 함수

2. 변수의 유효범위

3. lambda

■ 함수

● 어떠한 값을 입력하면 정의된 절차에 따라 일을 수행한 후 결과물을 내는 것

- 수학에서의 함수 : y = x + 3
- 파이썬에서의 함수

```
def 함수명(매개변수):
    수행할 문장
    수행할 문장
    return 결과값
```

● 함수를 사용하는 이유

- 반복되는 코드의 수를 줄일 수 있음
- 누군가가 만들어 놓은 **기능을 손쉽게 활용**할 수 있음
- 팀 단위로 **협업 / 분업** 가능
- 함수의 코드만 수정하면 실제 사용되는 여러 부분들이 자동으로 수정되므로 유지보수가 용이함

■ 함수

● 두 수를 입력받아 합을 구하는 함수

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

x = 10
y = 5

result = plus(x, y) # result = plus(10, 5)
print(result)
```

- plus 함수로 정의된 변수 **a, b** → **매개변수(parameter)**, 입력값
- plus 함수를 부르면서 입력해주는 변수 **x, y** → **인자(arguments)**
- return a + b → 반환 / 출력 / 결과값

● 함수의 형태는 입력값과 결과값의 존재 여부에 따라서 4가지로 나누어짐

- 입력값 O, 결과값 O : 일반적으로 많이 사용되는 형태
- 입력값 O, 결과값 X
- 입력값 X, 결과값 O
- 입력값 X, 결과값 X

■ 함수

● 함수를 호출 할 때 매개변수를 지정하여 호출

```
def total(a, b, c):
    return a + b + c

def average(num, count):
    return num / count

num = total(a=90, b=80, c=70) # 매개변수 지정 후 값 입력
avg = average(count=3, num=num) # 순서 상관없이 입력 가능
print(num, avg)
```

● 입력되는 값의 개수에 따라 동적으로 처리

- 입력값을 모두 더하는 함수 (1, 2, 3 → 6 1, 2, 3, 4, 5 → 15)

```
def total(*args): # Tuple 형태로 전달
    total = 0
    for n in args:
        total += n
    return total

num = total(1, 2, 3)
print(num)
num = total(1, 2, 3, 4, 5)
print(num)
```

■ 함수

● 키워드 파라미터

- 매개변수명이 지정된 입력값을 개수에 따라 동적으로 처리

```
def func(**kwargs): # Dictionary 형태로 저장
    print(kwargs)

func(a=1, b=2, c=3)
```

- 인자를 key=value 형태로 넘겨주고 함수 내부에서는 Dictionary 형태로 활용

● 가변 인자 방식과 키워드 파라미터 방식을 함께 사용

```
def func(*args, **kwargs): # Tuple, Dictionary
    print(args, kwargs)

func('a', 'b', a=1, b=2, c=3)
```

■ 함수

● 매개변수 기본값 지정

- 매개변수의 마지막에 위치

```
def setInfo(name, age, isPass=True):
   print(name, age, isPass)

setInfo('ggoreb', 20)
setInfo('ggoreb', 20, False)
```

- 사용불가

```
def setInfo(name, isPass=True, age):
    print(name, age, isPass)
```

■ 함수

● 결과값의 여러가지 반환 형태

- 일반적인 형태

```
def funcA():
    return 'a'

retA = funcA()
print(retA)
```

- 2개 이상의 값 반환

```
def funcB():
    return 'a', 'b' # Tuple 형태로 반환

retB = funcB() # Tuple 형태로 저장
print(retB)

retB1, retB2 = funcB() # 각 변수에 따로 저장
print(retB1, retB2)
```

■ 함수

● 결과값의 여러가지 반환 형태

- 함수 반환

```
def a():
    print('a')

def b():
    return a

b()()
```

- return 키워드 단독 사용

```
def divide(a, b):
    if b == 0: # 나누는 수(제수) 가 0 인 경우
        print('0 으로 나눌 수 없습니다 .')
        return  # 강제 종료
    return a / b

result = divide(4, 0) # result = None
print(result)
```

■ 연습문제

● 하나의 숫자를 입력받아 홀인지 짝인지 판단하는 함수 작성

```
# 함수 작성 코드

# 함수 호출 예)
print(is_odd(3))
print(is_odd(4))
```

● 1 이상의 정수 N을 입력받은 후 1 ~ N 까지의 합을 구하는 함수 작성

```
# 함수 작성 코드

# 함수 호출 예)
print(total(10))
```

● 숫자 요소를 가지는 리스트를 입력받은 후 합과 평균을 구하고 튜플 형태로 반환하는 함수 작성

```
# 함수 작성 코드

# 함수 호출 예)
print(calculator([3, 7, 9, 11]))
```

■ 연습문제

● 아래와 같은 수열이 있을 때 N번째 수열의 값을 구하는 함수 작성

```
1
1+(1+2)
1+(1+2)+(1+2+3)
1+(1+2)+(1+2+3)+(1+2+3+4)
⋮
1+(1+2)+(1+2+3)+(1+2+3+4)+...+(1+...+n)
```

● 문자열의 가운데 글자를 반환하는 함수 작성

조건 1) functions와 같은 홀수 길이가 입력되면 가운데 한 글자 반환 → t

조건 2) python과 같은 짝수 길이가 입력되면 가운데 두 글자 반환 → th

■ 연습문제

● 길이가 N이고, 수박수박수박수…와 같은 패턴을 반환하는 함수 작성

입력 예)	출력 예)
n=3	수박수
n=4	수박수박
n=7	수박수박수박수
n=11	수박수박수박수박수박수

● 자연수 N이 주어지면 N의 각 자리 수 합을 구하는 함수 작성

입력 예)	출력 예)
n=124	7
n=79	16
n=12345	15
n=13579	25

■ 변수의 유효범위

● 함수 안에서 변수 선언 (지역변수)

```
def func(a):
    b = a + 2
    return b

print(a)
print(b)
```

변수 a와 b의 유효범위

함수 내부에 선언된 변수는 외부에서 사용불가

● 함수 밖에서 변수 선언 (전역변수)

```
outer_a = 0

def func(a):
    b = a + 2
    return b

outer_a = func(10)
print(outer_a)
```

변수 outer_a의 유효범위

■ 변수의 유효범위

● 함수 안에서 변수 선언 (전역변수)

- 전역변수를 함수 내부에서 사용

```
outer_a = 0

def func(a):
    outer_a = a + 2
    print('함수 내부', outer_a)

func(10)
print('함수 외부', outer_a)
```

전역변수 outer_a

지역변수 outer_a, 새로운 지역변수 생성

```
outer_a = 0

def func(a):
    global outer_a
    outer_a = a + 2
    print('함수 내부', outer_a)

func(10)
print('함수 외부', outer_a)
```

전역변수 outer_a

전역변수 outer_a 사용 설정

■ lambda

- 함수를 생성할 때 사용
- def 예약어와 동일한 역할
  - 함수를 간결하게 한줄로 만들때
  - 함수가 함수를 반환할때
- 기본 형태
  - lambda 인자 1, 인자 2, ... , 인자 n : 표현식

일반적인 함수

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

result = plus(10, 20)
print(result)
```

lambda

```
plus = lambda a, b: a + b
result = plus(10, 20)

print(result)
```

■ 연습문제

● 아래의 함수를 lambda와 List Comprehension을 사용하는 함수로 작성

```
def myfunc(numbers):
    result = []
    for number in numbers:
        if number > 5:
            result.append(number)
    return result

result = myfunc([2, 3, 4, 5, 6, 7, 8])
print(result)
```