기초 PYTHON 프로그래밍

15. 함수 – 2

1. 전역 변수와 지역 변수
2. 함수의 인수
3. 람다 함수 (lambda)

서강대학교
SOGANG UNIVERSITY

1. 전역 변수와 지역 변수

서강대학교 SOGANG UNIVERSITY

- 전역 변수(global variable) : 프로그램 전체에서 사용 가능
- 지역 변수(local variable) : 함수 내에서만 사용 가능
- scoping rule : 변수를 찾을 때 지역 변수 → 전역 변수 순서로 찾는다.

```
def test():
    print(a)

# main

a = 100   # 전역변수
print(a)
test()
print(a)
```

```
100
100
100
```

```
def test():
    b = 20   # 지역변수
    print(a,b)

# main

a = 100   # 전역변수
print(a)
test()
print(a)

print(b)   # 에러
```

```
100
100 20
100
```

```
def test():
    a = 200  # 지역변수
    print(a)

# main

a = 100   # 전역변수
print(a)
test()
print(a)
```

```
100
200
100
```

1. 전역 변수와 지역 변수

서강대학교 SOGANG UNIVERSITY

- global 선언
  - 전역 변수를 함수 내에서 바꾸고자 하면, global 선언이 필요함

```
def test():
    a = 200

# main

a = 100
test()
print(a)
```

```
100
```

```
def test():
    global a
    a = 200

# main

a = 100
test()
print(a)
```

```
200
```

1. 전역 변수와 지역 변수

서강대학교 SOGANG UNIVERSITY

- global 선언
  - 함수에서 만든 지역 변수를 전역 변수로 사용하고자 한다면 global 선언이 필요함

```
def test():
    a = 100
    print(a)

# main

test()
print(a)
```

에러 발생

```
def test():
    global a
    a = 100
    print(a)

# main

test()
print(a)
```

```
100
100
```

1. 전역 변수와 지역 변수

서강대학교 SOGANG UNIVERSITY

- 함수 매개변수는 지역 변수이다.

```
def test(a):
    a += 200
    print(a)

# main

a = 100
test(a)
print(a)
```

지역 변수 a

전역 변수 a

```
300
100
```

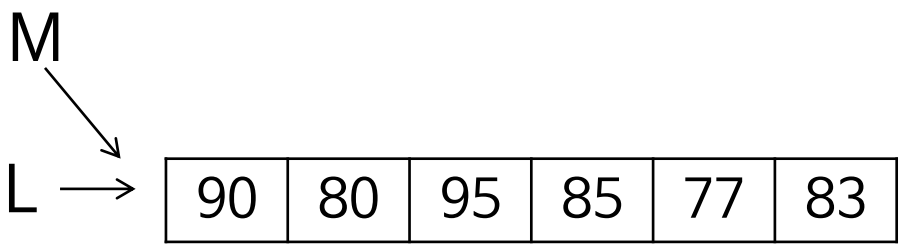
함수 test를 호출할 때 100을 인자로 넘기면 **지역 변수 a**가 생성되고 값으로 100을 갖는다.

1. 전역 변수와 지역 변수

서강대학교 SOGANG UNIVERSITY

- mutable 객체가 인수인 경우 (리스트, 집합, 사전)
  - 리스트 인수

```
def test(M):
    M[2] += 2
    M[4] += 10

# main

L = [90,80,95,85,77,83]
test(L)
print(L)
```



만약에 L 복사본이 생겨서 원본 L을 바꾸고 싶지 않다면 다음과 같이 test 함수에 인수를 넘겨야 한다.

test(L[:]) 또는 test(L[::])

```
[90, 80, 97, 85, 87, 83]
```

2. 함수의 인수

서강대학교 SOGANG UNIVERSITY

- 기본값(default value)이 있는 인수
  - 함수를 호출할 때 인수를 넘겨주지 않아도 인수가 자신의 기본값을 취하도록 하는 기능.

```
def inc(a, step=1):
    return a + step

b = inc(10)
print(b)    # 11

c = inc(10, 50)
print(c)    # 60
```

- 첫 번째 매개 변수 a 에는 반드시 인수를 넘겨야 한다.
- 두 번째 매개 변수인 step에는 인수를 넘기면 그 값이 step에 저장되고, 인수를 넘기지 않으면 1이 step 값으로 이용된다.

2. 함수의 인수

서강대학교 SOGANG UNIVERSITY

- 함수를 정의할 때 인수의 기본값 사용시 주의점
  - 기본값이 있는 인수가 먼저 올 수 없다.

```
def inc(step=1, a):
    return a + step
```

에러

  - 인수가 여러 개인 경우의 예

```
def test(a, b=1, c=2):          # OK
        return a + b + c

def test2(a, b, c=10):          # OK
        return a + b + c

def test3(a, b=1, c):           # 에러
        return a + b + c
```

```
>>> L = [4,3,5,1,2]
>>> L.sort()
>>> print(L)
[1, 2, 3, 4, 5]
>>> M = [3,5,1,2,4]
>>> M.sort(reverse=True)
>>> print(M)
[5, 4, 3, 2, 1]
>>> a,b = 10, 20
>>> print(a,b)
10 20
>>> print(a,b,sep=':')
10:20
>>> print(a,end=' ^^ '); print(b)
10 ^^ 20
```

2. 함수의 인수

서강대학교 SOGANG UNIVERSITY

- 키워드 인수
  - 함수 호출 시에 인수 이름(매개변수)과 값을 같이 전달하기

```
def area(x, y):
    return x * y
```

```
a = area(10,5)
b = area(x=4, y=9)      # 매개변수와 값을 같이 적어 준다
c = area(y=5, x=10)     # 매개변수와 값을 같이 적으면 순서 상관없다
d = area(10, y=5)       # OK (일반 인수 뒤에 키워드 인수 올 수 있다)
e = area(x=10, 5)       # 에러
```

2. 함수의 인수

서강대학교 SOGANG UNIVERSITY

- 키워드 인수
  - 키워드 인수의 위치는 일반 인수 이후이다.

```
def volume(x,y,z):
    return x * y * z
```

일반 인수 : positional argument

```
volume(1, 3, 5)          # 15
volume(y=7, z=5, x=2)    # 70
volume(z=2, x=4, y=5)    # 40
volume(5, z=10, y=2)     # 100
volume(5, x=2, z=20)     # 에러 (x에 두 개의 값이 할당된다는 에러)
volume(z=10, 20, x=2)    # 에러 (일반 인수가 키워드 인수 뒤에 오면 안됨)
```

2. 함수의 인수

서강대학교 SOGANG UNIVERSITY

◆ 가변 인수 사용하기

- 정해지지 않은 수의 인수를 함수에 전달하기
- 함수를 정의할 때 인수 목록에 반드시 넘겨야 하는 고정 인수를 우선 나열하고, 나머지를 마지막에 튜플 형식으로 한꺼번에 받는다.

```
def friends(*name):
    for n in name:
        print(n.title())

friends('alice', 'paul')
print('------')
friends('cindy', 'sally', 'david', 'tom')
```

```
Alice
Paul
------
Cindy
Sally
David
Tom
```

```
def arg(a, b, *c):
    print(a,b,c)
    print(sum(c))

# main
arg(10,20)
print('----------')
arg(7,8,9)
print('----------')
arg(1,2,3,4,5)
```

```
10 20 ()
0
----------
7 8 (9,)
9
----------
1 2 (3, 4, 5)
12
```

2. 함수의 인수

서강대학교 SOGANG UNIVERSITY

- 정의되지 않은 키워드 인수 처리하기
  - ** 형식으로 기술
  - 전달받는 형식은 사전이다. 즉, 키는 키워드(변수명)가 되고, 값은 키워드 인수로 전달되는 값이 된다.

```
def name_age(**lists):
    print(lists)

name_age(Alice=10, Paul=12)
print('----------------------------')
name_age(Cindy=5, David=7, Tom=10)
```

```
{'Alice': 10, 'Paul': 12}
----------------------------
{'David': 7, 'Tom': 10, 'Cindy': 5}
```

3. 람다 함수 (lambda)

서강대학교 SOGANG UNIVERSITY

- 람다 함수의 정의
  - 이름없는 한 줄짜리 함수이다.
  - 람다 함수는 return 문을 사용하지 않는다.
  - 람다 함수의 몸체는 문이 아닌 하나의 식이다.
  - 람다 함수는 함수를 함수 인자로 넘길 때 유용하다.

lambda 〈인수들〉 : 〈반환할 식〉

```
def add(x,y):
    return x+y
```

→

```
lambda x, y: x + y
```

3. 람다 함수 (lambda)

서강대학교 SOGANG UNIVERSITY

- map 내장 함수

map(함수명 , 함수에 대한 인수 집합)

시퀀스 자료형 (리스트, 튜플, 문자열)

```
>>> names = ['Alice', 'Paul', 'Bob', 'Robert']
>>> A = map(len, names)
>>> print(A)
<map object at 0x036091D0>
>>> list(A)
[5, 4, 3, 6]
```

3. 람다 함수 (lambda)

서강대학교 SOGANG UNIVERSITY

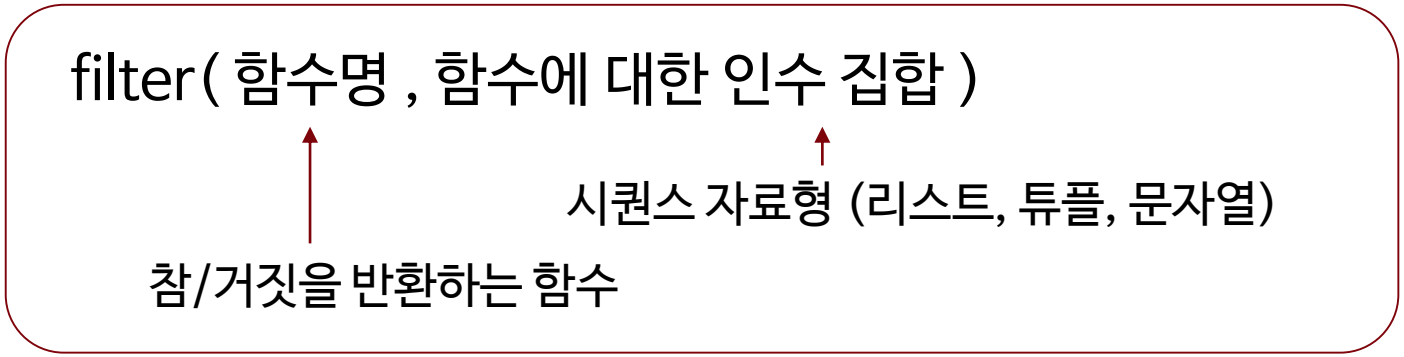
- map 내장 함수 예

```
>>> def f(x):
        return x * x
>>> X = [1, 2, 3, 4, 5]
>>> list(map( f, X ))        # 함수 f 에 X의 값들을 하나씩 적용한다
[1, 4, 9, 16, 25]

>>> X = [1, 2, 3, 4, 5]
>>> Y = map( lambda a:a * a, X )
>>> list(Y)
[1, 4, 9, 16, 25]

# range(10)의 원소를 식 x²+4x+5 에 대입한 결과를 내 준다
>>> Y = map( lambda x: x * x + 4 * x + 5, range(10) )
>>> list(Y)
[5, 10, 17, 26, 37, 50, 65, 82, 101, 122]
```

3. 람다 함수 (lambda)

서강대학교 SOGANG UNIVERSITY

- filter 내장 함수




```
>>> X = [1,3,5,7,9]
>>> result = filter( lambda x:x>5, X )
>>> print(result)
<filter object at 0x03FAD0D0>
>>> list(result)
[7, 9]

>>> list(filter( lambda x:x%2==1, range(11)))
[1, 3, 5, 7, 9]
```