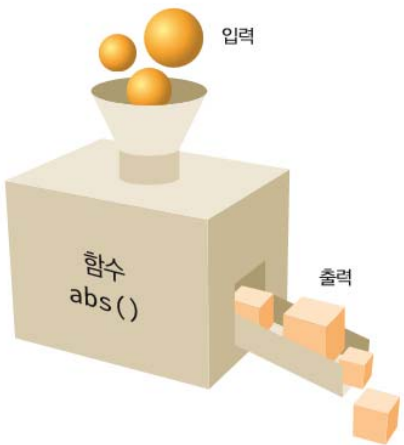
5장. 함 수

함수란?

- **함수(function)는** 특정 작업을 수행하는 명령어들의 모음에 이름을 붙인 것
- 함수는 작업에 필요한 데이터를 전달받을 수 있으며, 작업이 완료된 후에는 작업의 결과를 호출자에게 반환할 수 있다.



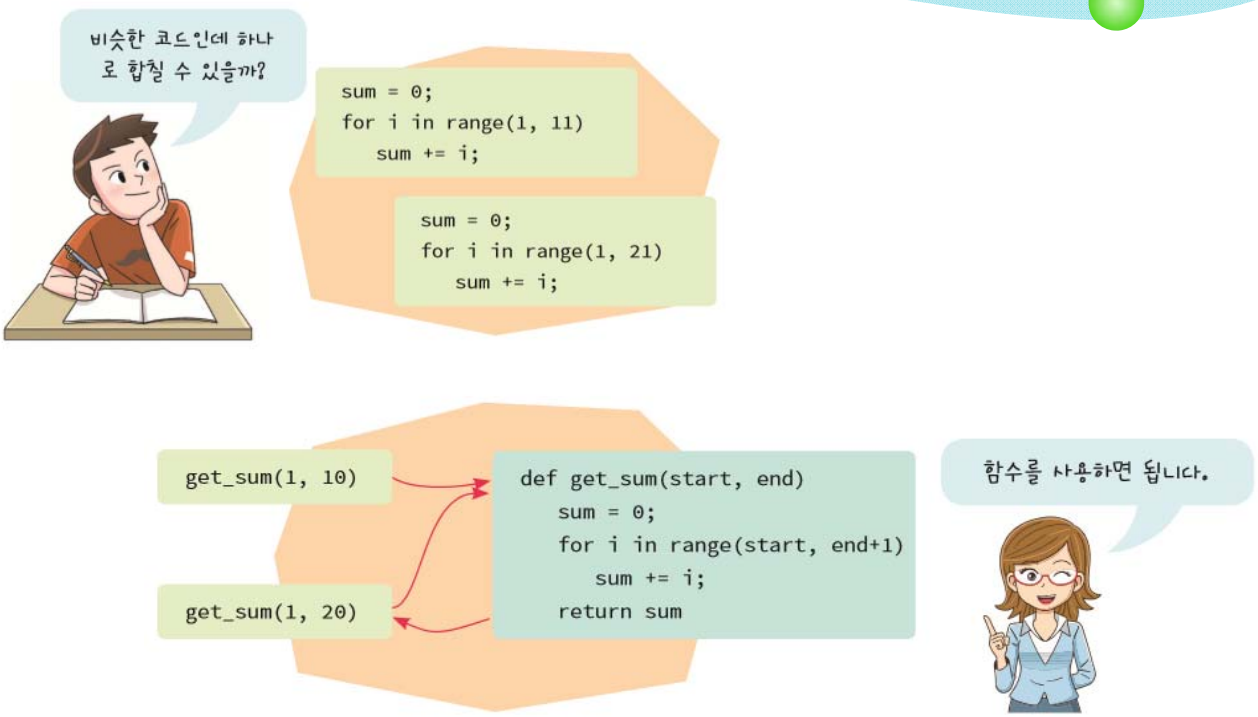
함수의 예

- print()
- input()
- abs(), …

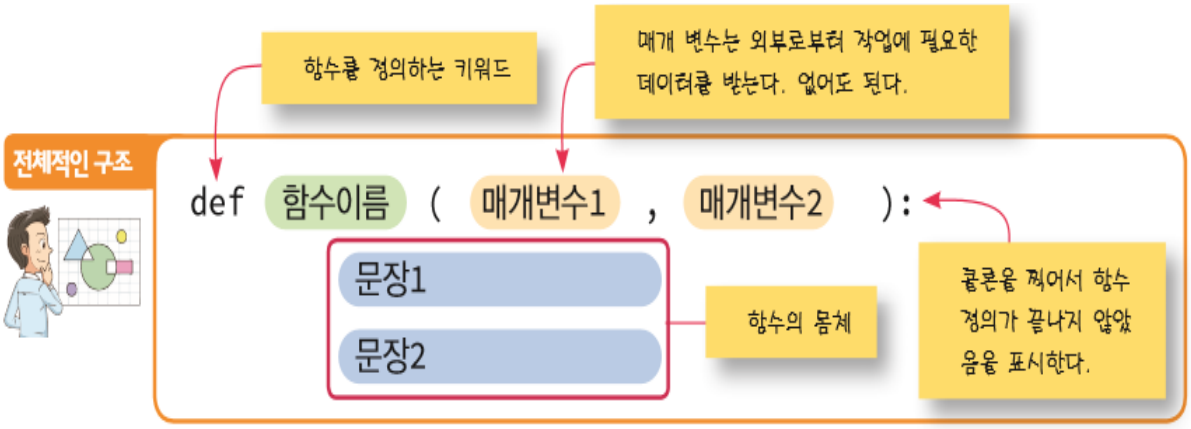
- 함수 안의 명령어들을 실행하려면 함수를 **호출**(call)하면 된다.

```
>>> value = abs(-100)
>>> value
100
```

함수는 왜 필요한가?



함수를 작성해보자.



함수 작성의 예

```
>>>def say_hello(name):
        print("안녕, ", name)

>>>say_hello("철수")
안녕, 철수

>>>def say_hello(name, msg):
        print("안녕, ", name, "야, ", msg)

>>>name="철수"
>>>msg="어서 집에 오너라"
>>>say_hello(name, msg)
안녕, 철수야, 어서 집에 오너라
```

값을 반환하는 함수

```
>>>def get_sum(start, end) :
        sum=0
        for i in range(start, end+1) :
            sum += i
        return sum
```

```
>>>value = get_sum(1, 10)
>>>print(value)
55
```

함수 사용의 장점

- 프로그램 안에서 중복된 코드를 제거한다.
- 복잡한 프로그래밍 작업을 더 간단한 작업들로 분해할 수 있다
- 함수는 한번 만들어지면 다른 프로그램에서도 재사용될 수 있다.
- 함수를 사용하면 가독성이 증대되고, 유지 관리도 쉬워진다.

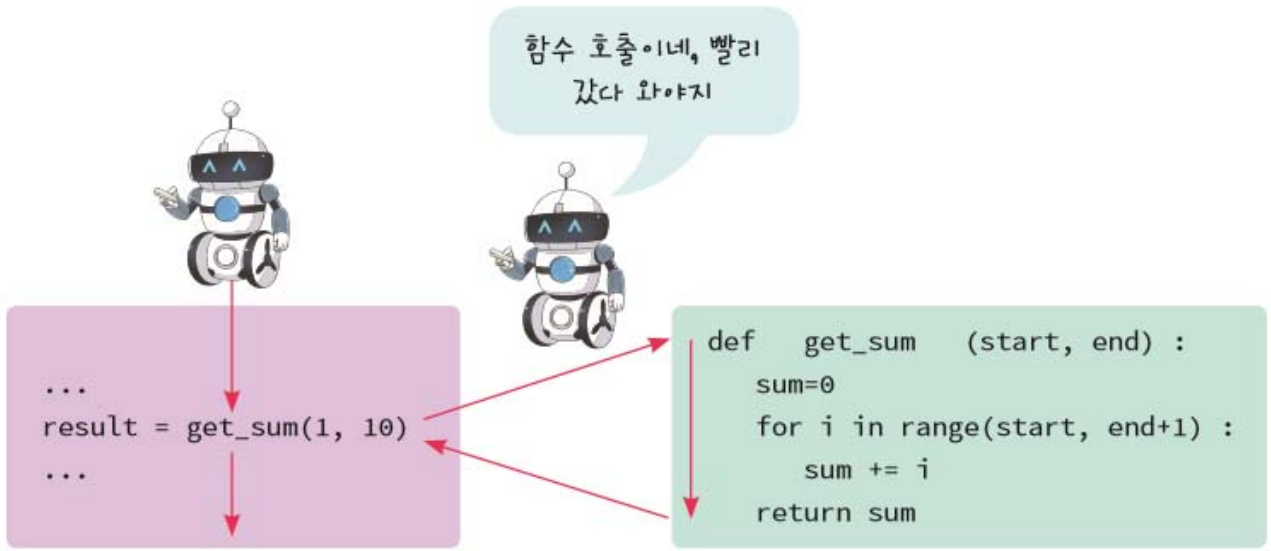
함수의 이름

- 함수의 목적을 설명하는 **동사** 또는 **동사+명사**를 사용

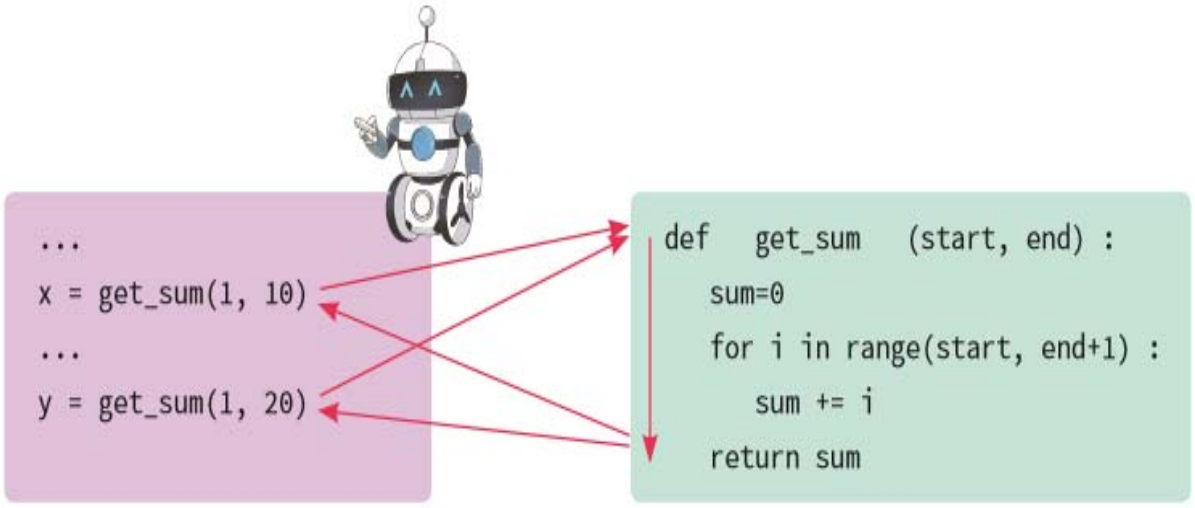
```
square(side)                # 정수를 제곱하는 함수
compute_average(list)       # 평균을 구하는 함수
set_cursor_type(c)          # 커서의 타입을 설정하는 함수
```

함수 호출

- 함수를 사용하려면 함수를 **호출**(call)하여야 한다.



함수는 여러 번 호출할 수 있다.



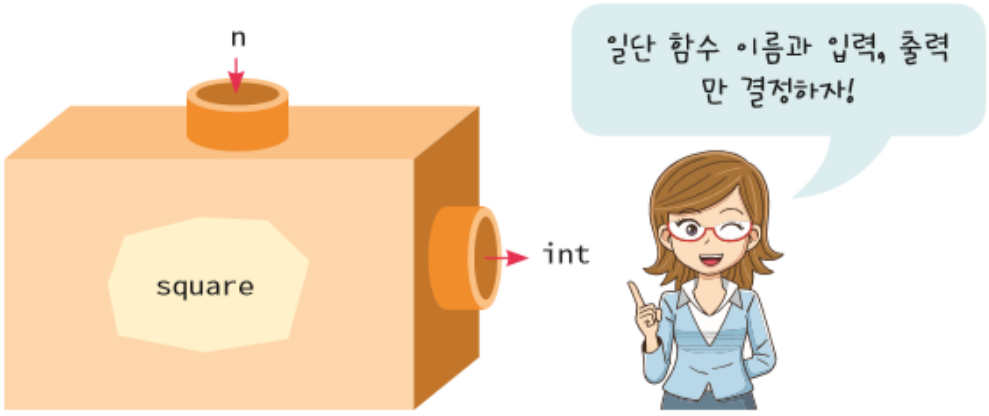
예제

```
def   get_sum(start, end) :
        sum=0
        for i in range(start, end+1) :
                sum += i
        return sum

print( get_sum(1, 10))
print( get_sum(1, 20))
```

```
55
210
```

함수 작성의 예 #1

- 정수를 입력 받아서 제곱한 값을 반환하는 함수를 만들어보자.



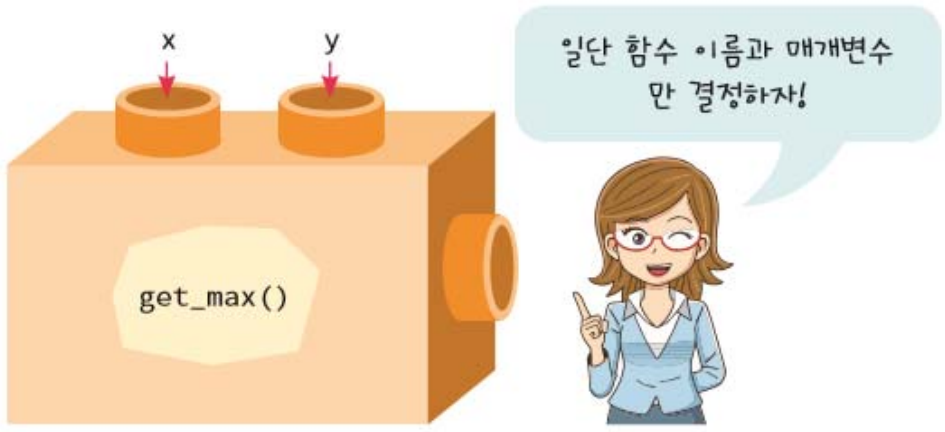
예제

```
def square(n):
        return(n*n)
print(square(10))
```

```
100
```

함수 작성의 예 #2

- 두 개의 정수가 주어지면 두수 중에서 더 큰 수를 찾아서 이것을 반환하는 함수를 만들어보자.



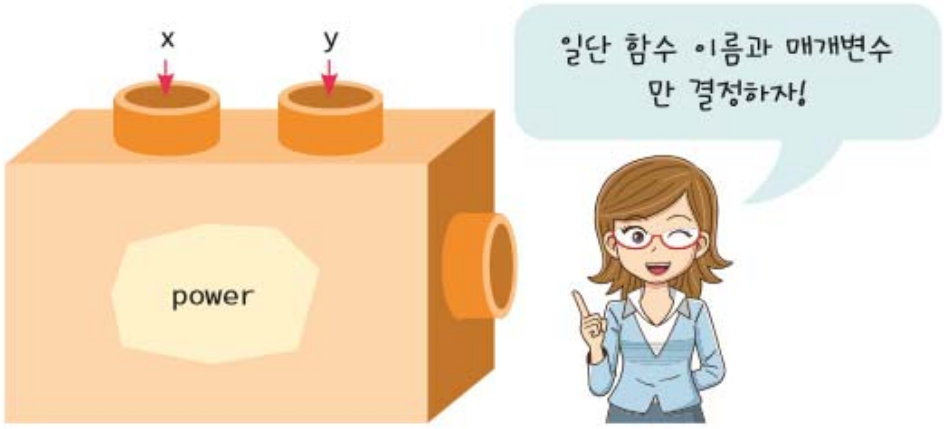
예제

```
def get_max(x, y):
        if( x > y ):
                return x
        else:
                return y

print(get_max(10, 20))
```

```
20
```

함수 작성의 예 #2

- 정수의 거듭제곱값을 계산하여 반환하는 함수를 작성하여 보자. (파이썬에는 ** 연산자가 있지만)



예제

```
def power(x, y):
        result = 1
        for i in range(y):
                result = result * x
        return result

print(power(10, 2))
```

```
100
```

함수를 이용할 때 주의할 점

- 파이썬 인터프리터는 함수가 정의되면 함수 안의 문장들은 즉시 실행하지 않는다.
- 함수 정의가 아닌 문장들은 즉시 실행하게 된다.

```
print(power(10, 2))

def power(x, y):
        result = 1
        for i in range(y):
                result = result * x
        return result
```

무엇이 문제인가?

```
Traceback (most recent call last):
  File "C:/Python34/test.py", line 1, in <module>
    print(power(10, 2))
NameError: name 'power' is not defined
```

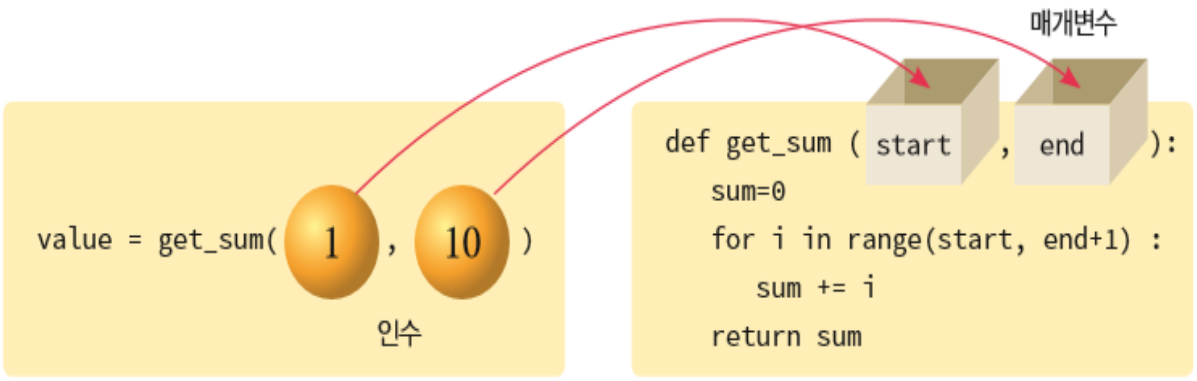
예제

```
def main():
        print(power(10, 2))

def power(x, y):
        result = 1
        for i in range(y):
                result = result * x
        return result

main()
```

이런 형태는 가능하다!

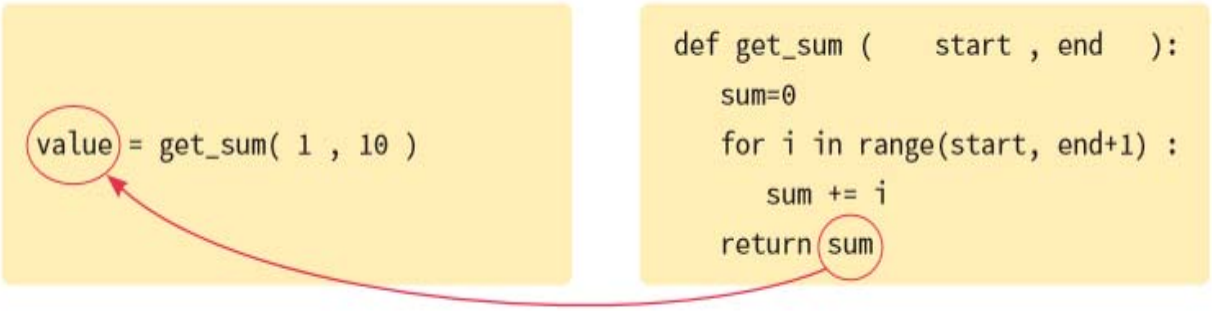
인수와 매개 변수

- 인수(argument)는 호출 프로그램에 의하여 함수에 실제로 전달되는 값이다.
- 매개 변수(parameter)는 이 값을 전달받는 변수이다.



함수가 값을 반환

- 반환값(return value)은 함수가 호출한 곳으로 반환하는 작업의 결과값이다.



Lab: 패스워드 생성기

- 일회용 패스워드 생성기를 이용하여서 3개의 패스워드를 생성하여 출력하는 프로그램을 작성해보자.

```
q546zv
1kvkss
b3vrmi
```

Solution

```
import random

def genPass():
        alphabet = "abcdefghijklmnopqrstuvwxyz0123456789"
        password = ""

        for i in range(6):
            index = random.randrange(len(alphabet))
            password = password + alphabet[index]
        return password

print(genPass())
print(genPass())
print(genPass())
```

디폴트 인수

- 파이썬에서는 함수의 매개변수가 기본값을 가질 수 있다. 이것을 **디폴트 인수(default argument)**라고 한다.

```
def greet(name, msg="별일없죠?"):
    print("안녕 ", name + ', ' + msg)

greet("영희")
```

```
안녕  영희, 별일없죠?
```

키워드 인수

- 인수들이 위치가 아니고 **키워드에 의하여 함수로 전달되는 방식**

```
>>> def calc(x, y, z):
    return x+y+z

>>> calc(10, 20, 30)
60

>>> calc(x=10, y=20, z=30)
60

>>> calc(y=20, x=10, z=30)
60
```

참조값에 의한 인수 전달

- 함수를 호출할 때, 변수를 전달하는 2가지 방법
  - 값에 의한 호출(call-by-value)
  - 참조에 의한 호출

값에 의한 호출 (1)

```
def modify(n):
        n = n + 1

k = 10
print("k=", k)
modify(k)
print("k=", k)
```

```
k= 10
k= 10
```

```
def modify(n):
        n = n + 1
        print ("n의 id:", id(n))

k = 10
print("k=", k)
print ("k의 id:", id(k))
modify(k)
print("k=", k)
print ("k의 id:", id(k))
```

id 함수는 참조값을 출력함

```
k= 10
k의 id: 1752224352
n의 id: 1752224368
k= 10
k의 id: 1752224352
```

값에 의한 호출 (2)

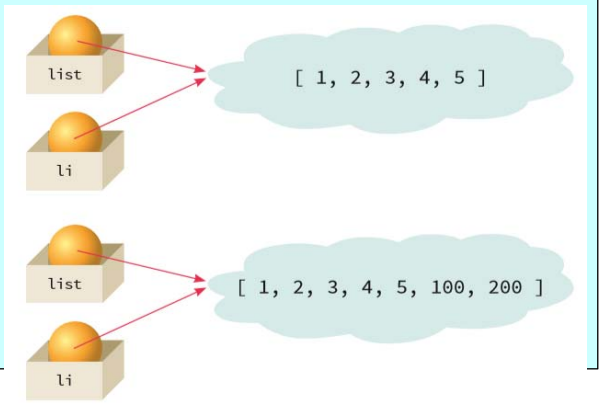
```
def modify1(s):
        s += "To You"

msg = "Happy Birthday"
print("msg=", msg)
modify1(msg)
print("msg=", msg)
```


```
msg= Happy Birthday
msg= Happy Birthday
```

숫자와 문자열은 변경 불가능한 객체여서
새로운 객체를 생성하여 변경하게 됨

참조에 의한 호출

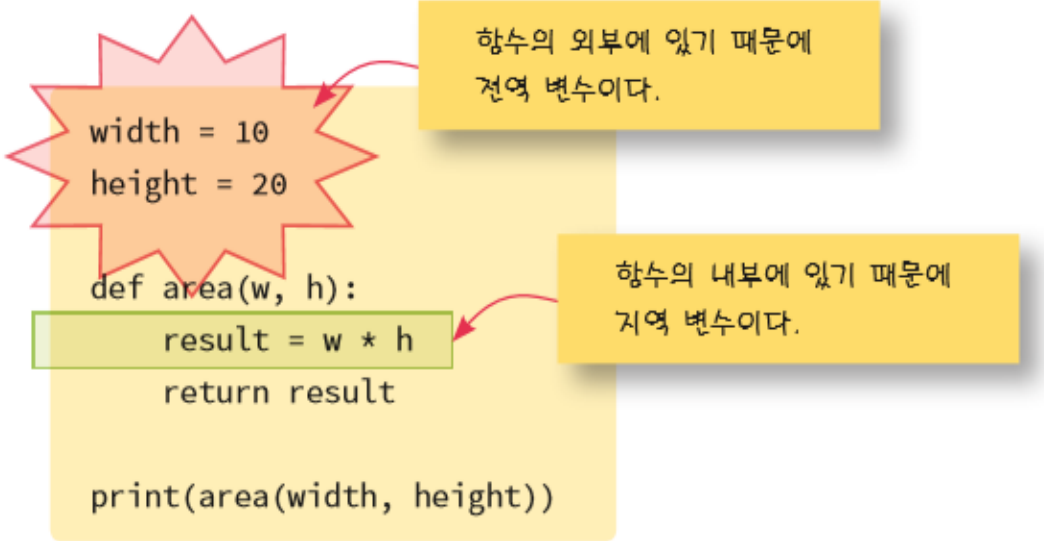
```
def modify2(li):
    li += [100, 200]

list = [1, 2, 3, 4, 5]
print(list)
modify2(list)
print(list)
```




```
[1, 2, 3, 4, 5]
[1, 2, 3, 4, 5, 100, 200]
```

리스트는 변경 가능한 객체여서 원본을 수정하게 됨

지역 변수와 전역 변수



지역 변수

```
def sub():
        s = "바나나가 좋음!"
        print(s)

sub()
```

```
바나나가 좋음!
```

전역 변수

```
def sub():
        print(s)

s = "사과가 좋음!"
sub()
```

```
사과가 좋음!
```

지역 변수와 전역 변수

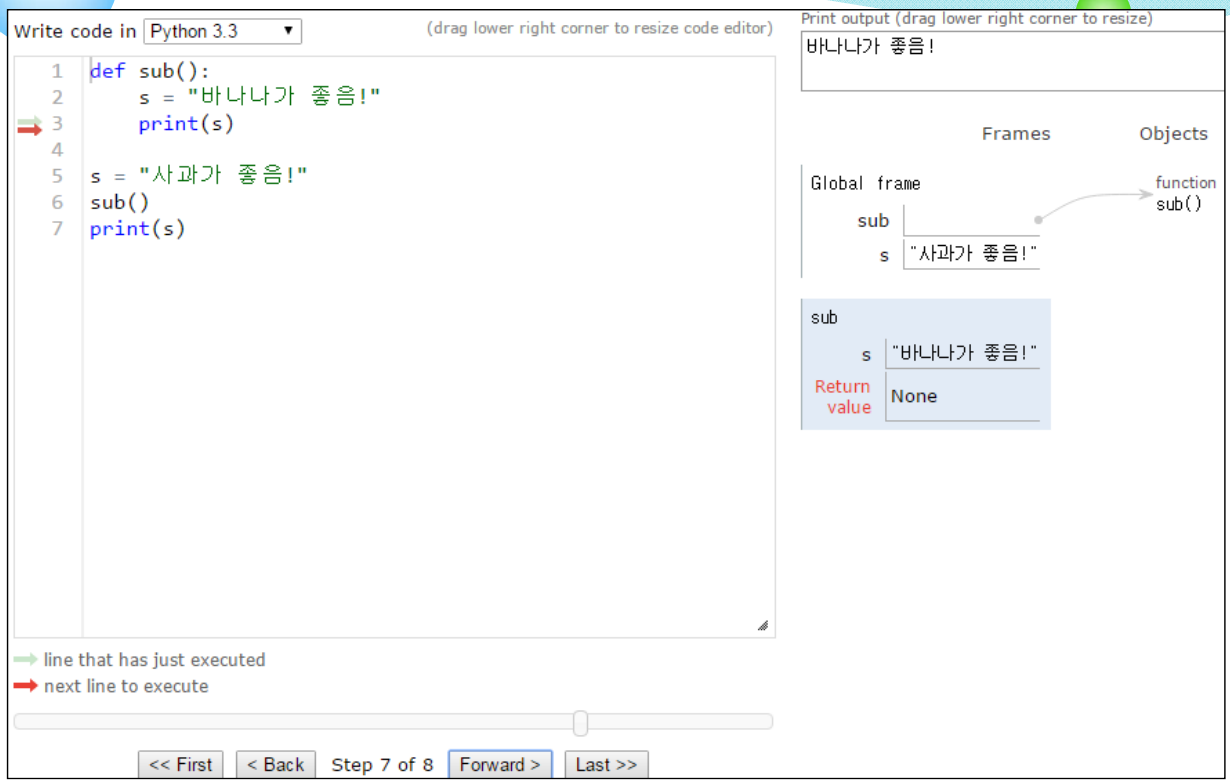
```
def sub():
        s = "바나나가 좋음!"
        print(s)

s = "사과가 좋음!"
sub()
print(s)
```

전역 변수를 사용하는 것이 아님!

```
바나나가 좋음!
사과가 좋음!
```

Python Tutor

- http://www.pythontutor.com/



전역 변수를 함수 안에서 사용하려면

```
def sub():
        global s

        print(s)
        s = "바나나가 좋음!"
        print(s)

s = "사과가 좋음!"
sub()
print(s)
```

```
사과가 좋음!
바나나가 좋음!
바나나가 좋음!
```

예제

```
def sub(x, y):
    global a

    a = 7
    x, y = y, x
    b = 3
    print(a, b, x, y)

a, b, x, y = 1, 2, 3, 4
sub(x, y)
print(a, b, x, y)
```

```
7 3 4 3
7 2 3 4
```

Lab: 매개변수 = 지역변수

- 다음 프로그램의 실행결과는 어떻게 될까?

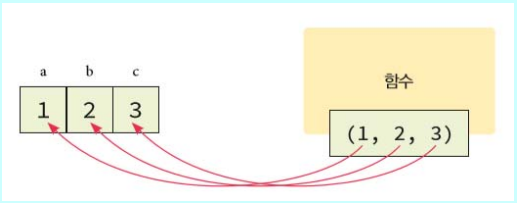
```
# 함수가 정의된다.
def sub( mylist ):
   # 리스트가 함수로 전달된다.
   mylist = [1, 2, 3, 4]  # 새로운 리스트가 매개변수로 할당된다.
   print ("함수 내부에서의 mylist: ", mylist)
   return

# 여기서 sub() 함수를 호출한다.
mylist = [10, 20, 30, 40];
sub( mylist );
print ("함수 외부에서의 mylist: ", mylist)
```

Solution

```
함수 내부에서의 mylist: [1, 2, 3, 4]
함수 외부에서의 mylist: [10, 20, 30, 40]
```

여러 개의 값 반환하기

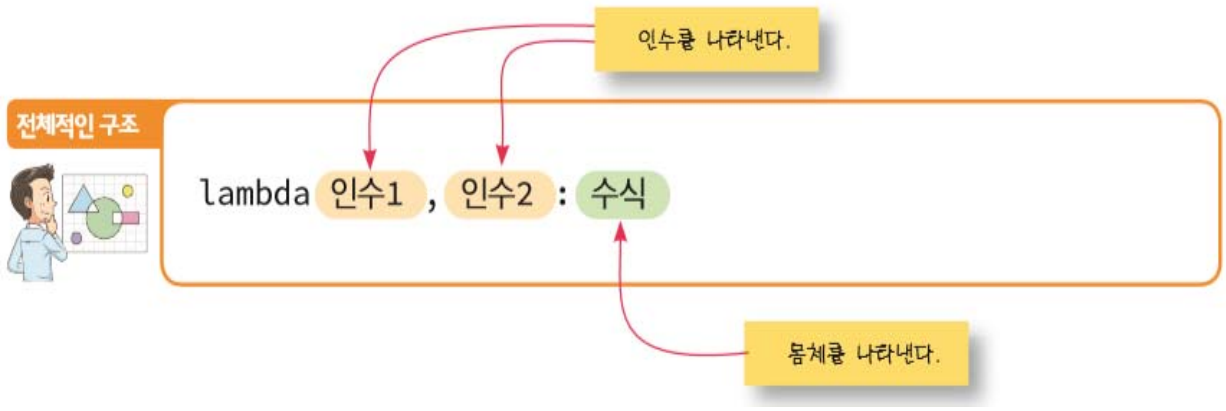
```
def sub():
        return 1, 2, 3

a, b, c = sub()
print(a, b, c)
```




```
1 2 3
```

무명 함수

- **무명 함수**는 이름은 없고 몸체만 있는 함수이다. 파이썬에서 무명 함수는 lambda 키워드로 만들어진다.



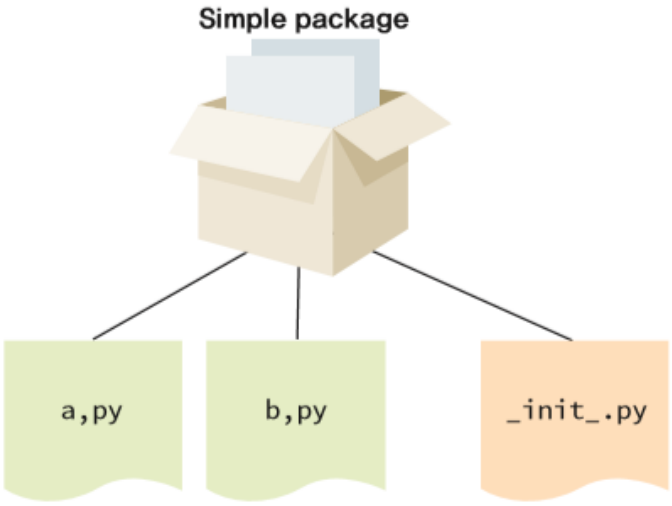
무명 함수의 예

```
sum = lambda x, y: x+y;

print( "정수의 합 : ", sum( 10, 20 ))
print( "정수의 합 : ", sum( 20, 20 ))
```

```
정수의 합 : 30
정수의 합 : 40
```

모듈이란?

- 함수나 변수들을 모아 놓은 파일을 **모듈**(module)



모듈 작성

fibo.py

```
# 피보나치 수열 모듈

def fib(n):       # 피보나치 수열을 화면에 출력한다.
    a, b = 0, 1
    while b < n:
        print(b, end=' ')
        a, b = b, a+b
    print()
```

모듈 사용

```
>>> import fibo

>>> fibo.fib(1000)
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987

>>> fibo.__name__
'fibo'
```

```
>>> from fibo import *
>>> fib(500)
1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

모듈을 스크립트로 실행하기

- 만약 파이썬 모듈을 다음과 같이 명령어 프롬프트를 이용하여 실행한다면

```
C:> python fibo.py <arguments>
```

```
if __name__ == "__main__":
    import sys
    fib(int(sys.argv[1]))
```

예 제

```
C:> python fibo.py 50
1 1 2 3 5 8 13 21 34
```

```
if __name__ == "__main__":
    import sys
    fib(int(sys.argv[1]))
```

핵심 정리

- **함수**는 동일한 코드를 재사용하기 위한 것이다. 함수는 def로 작성된다.
- 함수 안에서 선언되는 변수는 **지역 변수**이고 함수의 외부에서 선언되는 변수는 **전역 변수**이다.
- 함수나 변수들을 모아 놓은 파일을 **모듈(module)**이라고 한다.