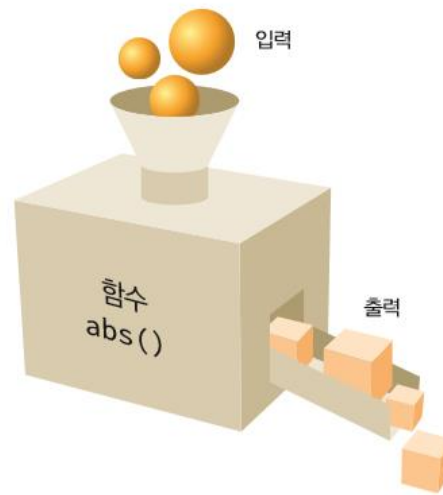


# 5장 함수

# 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?

비슷한 코드인데 하나로 합칠 수 있을까?



```
sum = 0;
for i in range(1, 11)
    sum += i;
```

```
sum = 0;
for i in range(1, 21)
    sum += i;
```

get\_sum(1, 10)

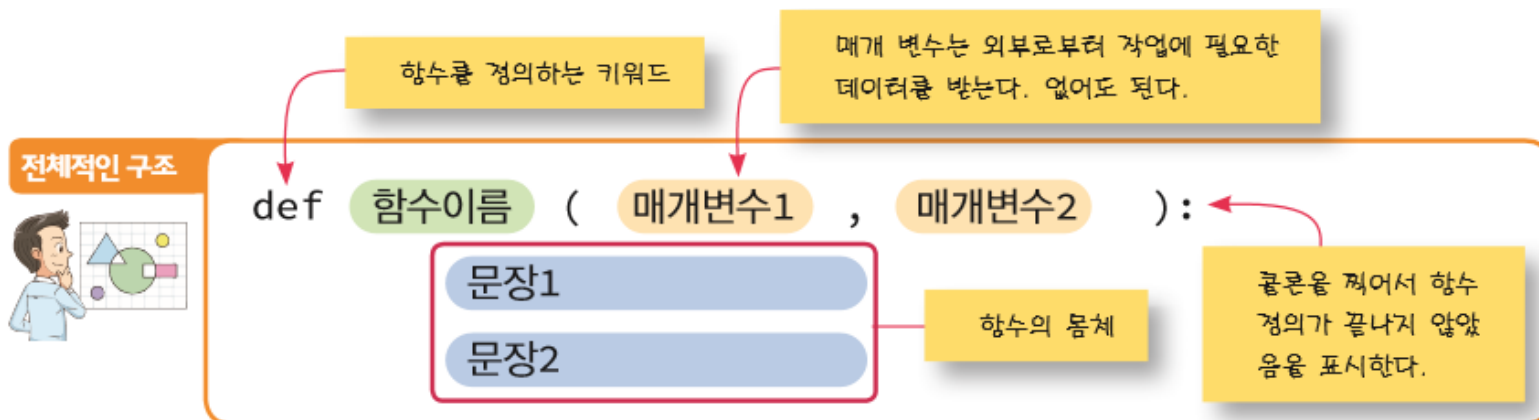
get\_sum(1, 20)

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

함수를 사용하면 됩니다.

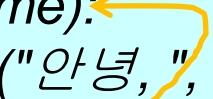


# 함수를 작성해보자.



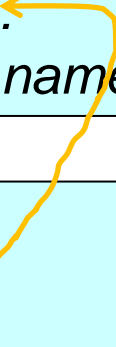
# 함수 작성의 예

```
>>>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+명사를 사용

`square(side)`

// 정수를 제공하는 함수

`compute_average(list)`

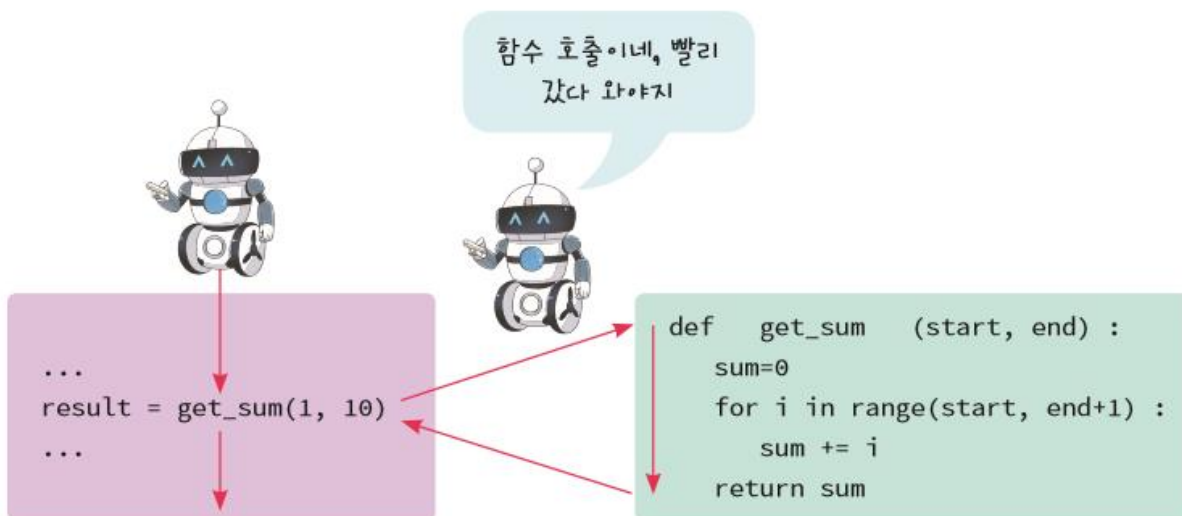
// 평균을 구하는 함수

`set_cursor_type(c)`

// 커서의 타입을 설정하는 함수

# 함수 호출

- 함수를 사용하려면 함수를 호출(call)하여야 한다.



# 함수는 여러 번 호출할 수 있다.



```
...  
x = get_sum(1, 10)  
...  
y = get_sum(1, 20)
```

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

# 예제

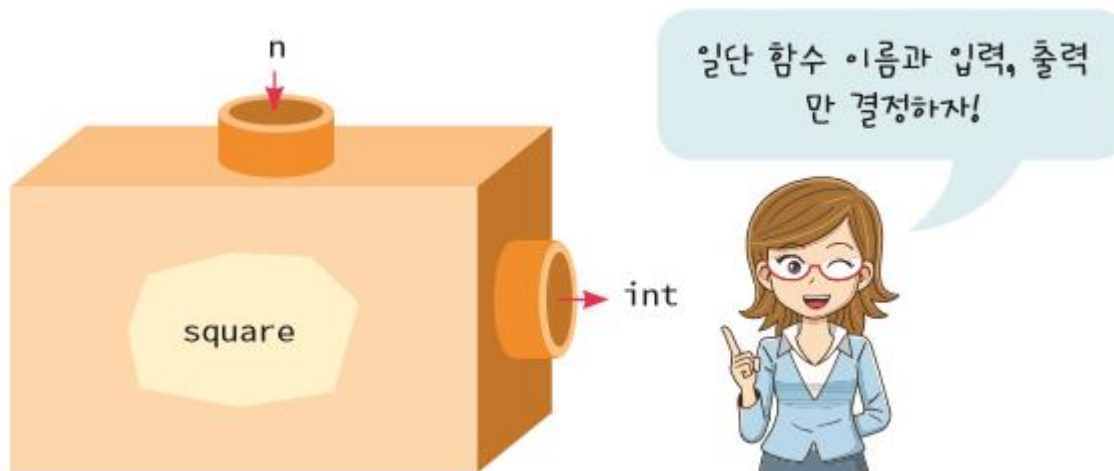
```
def get_sum(start, end) :  
    sum=0  
    for i in range(start, end+1) :  
        sum += i  
    return sum
```

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

```
55  
210
```

# 함수 작성의 예 #1

- 정수를 입력받아서 제공한 값을 반환하는 함수를 만들어 보자.



# 예제

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

100

## 함수 작성의 예 #2

- 두개의 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.

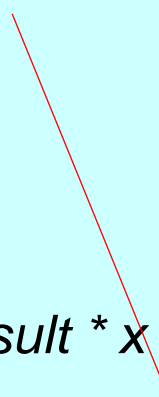
```
print(power(10, 2))
```

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

무엇이 문제인가?

# 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!

# Lab: 생일 축하 함수

- 생일 축하 메시지를 출력하는 함수 `happyBirthday()`를 작성해보자.

생일 축하 합니다!  
생일 축하 합니다!  
사랑하는 친구의 생일 축하 합니다!

# Solution

```
def happyBirthday():  
    print("생일 축하 합니다 !")  
    print("생일 축하 합니다 !")  
    print("사랑하는 친구의", end=" ")  
    print("생일 축하 합니다 !")  
  
happyBirthday()
```

# Lab: 온도 변환 함수

- 섭씨 온도를 화씨 온도로 변환하여 반환하는 함수 `FtoC()`를 작성하고 테스트하라.

화씨온도를 입력하시오: 32.0  
0.0

# Solution

```
# 함수가 여기서 정의된다.  
def FtoC(temp_f):  
    temp_c = (5.0 * (temp_f - 32.0)) / 9.0;  
    return temp_c;  
  
temp_f = float(input("화씨 온도를 입력하시오: "))  
  
# FtoC() 함수를 호출한다.  
print(FtoC(temp_f))
```



# Lab: 소수 찾기

- 여기서는 소수를 판별하는 함수 `is_prime()`을 작성하여 사용하여 보자.

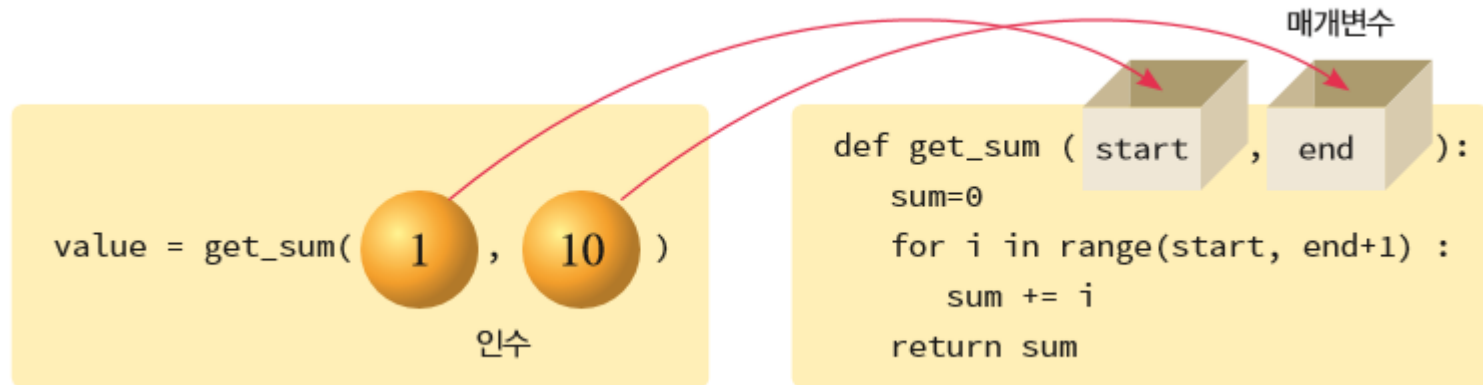
정수를 입력하시오: 101  
*True*

# Solution

```
def is_prime(n):  
    for i in range(2, n):  
        if ( n%i == 0 ):  
            return False  
    return True  
  
n = int(input("정수를 입력하시오: "))  
print(is_prime(n))
```

# 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.

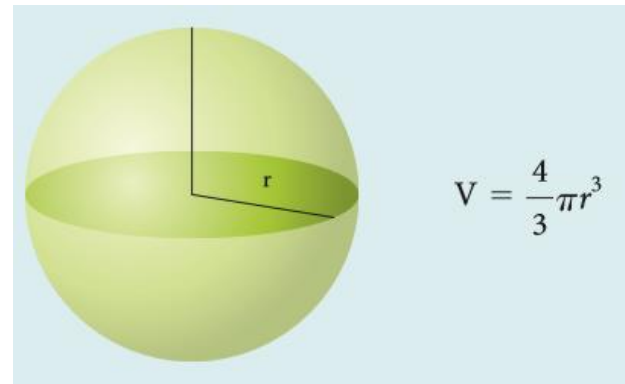
**value** = get\_sum( 1 , 10 )

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

# Lab: 소수 찾기

- 구의 부피를 계산하는 함수 `sphereVolume()`을 작성하여 보자. 반지름이  $r$ 인 구의 부피는 다음과 같다.

구의 반지름을 입력하시오: 10.0  
4188.790204786391



# Solution

```
import math

def sphereVolume(radius):
    volume = (4.0 / 3.0) * math.pi * radius * radius * radius
    return volume;

radius = float(input('구의 반지름을 입력하시오: '))
print(sphereVolume(radius))
```

# 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
```

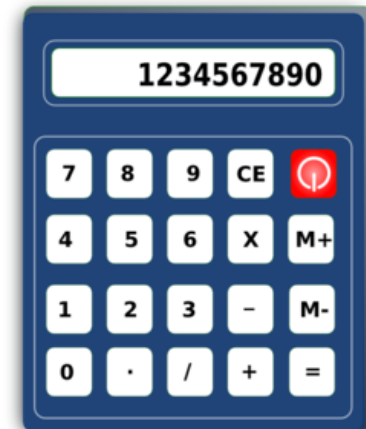
```
>>> calc(x=10, y=20, z=30)  
60
```

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

# Lab: 패스워드 생성기

- 사칙 연산을 수행하는 4개의 함수(add(), sub(), mul(), div())를 작성한다. 이들 함수를 이용하여  $10+20*30$ 을 계산하여 보자. 함수를 호출할 때 키워드 인수를 사용하여 호출해보자.

610



# Solution

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

```
def sub(a, b):  
    return a - b
```

```
def mul(a, b):  
    return a * b
```

```
def div(a, b):  
    return a / b
```

```
r1 = mul(a=20, b=30)  
r2 = add(a=10, b=r1)  
print(r2)
```

# Lab: 온도변환기

- 섭씨 온도를 화씨 온도로, 또 그 반대로 변환하는 프로그램을 작성하여 보자.

'c' 섭씨온도에서 화씨온도로 변환

'f' 화씨온도에서 섭씨온도로 변환

'q' 종료

메뉴에서 선택하세요.c

섭씨온도: 100

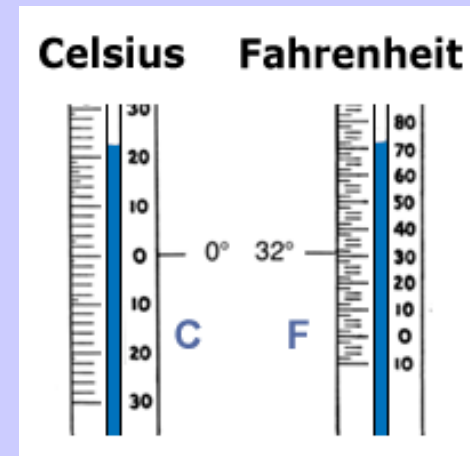
화씨온도: 212.0

'c' 섭씨온도에서 화씨온도로 변환

'f' 화씨온도에서 섭씨온도로 변환

'q' 종료

메뉴에서 선택하세요.



```
def printOptions():  
    print( " 'c' 섭씨온도에서 화씨온도로 변환")  
    print( " 'f' 화씨온도에서 섭씨온도로 변환")  
    print( " 'q' 종료")
```

```
def C2F(c_temp):  
    return 9.0 / 5.0 * c_temp + 32
```

```
def F2C(f_temp):  
    return (f_temp - 32.0) * 5.0 / 9.0
```

```
printOptions()  
choice = input("메뉴에서 선택하세요.")  
while choice != "q":  
    if choice == "c":  
        temp = float( input("섭씨온도: "))  
        print ("화씨온도:", C2F(temp))  
    elif choice == "f":  
        temp = float(input("화씨온도: "))  
        print ("섭씨온도:", F2C(temp))  
    printOptions()  
    choice = input("메뉴에서 선택하세요.")
```

# 참조값에 의한 인수 전달

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

# 값에 의한 호출

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

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



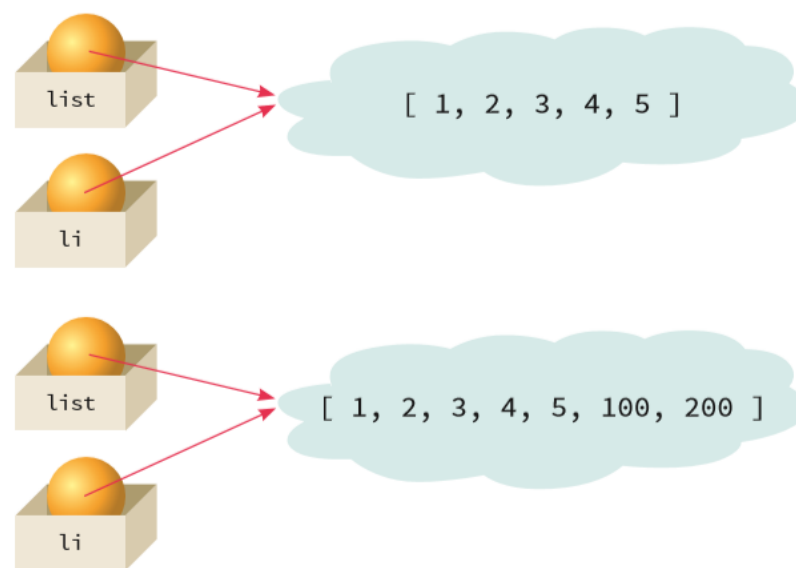
```
msg= Happy Birthday  
msg= Happy Birthday
```



# 참조에 의한 호출

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

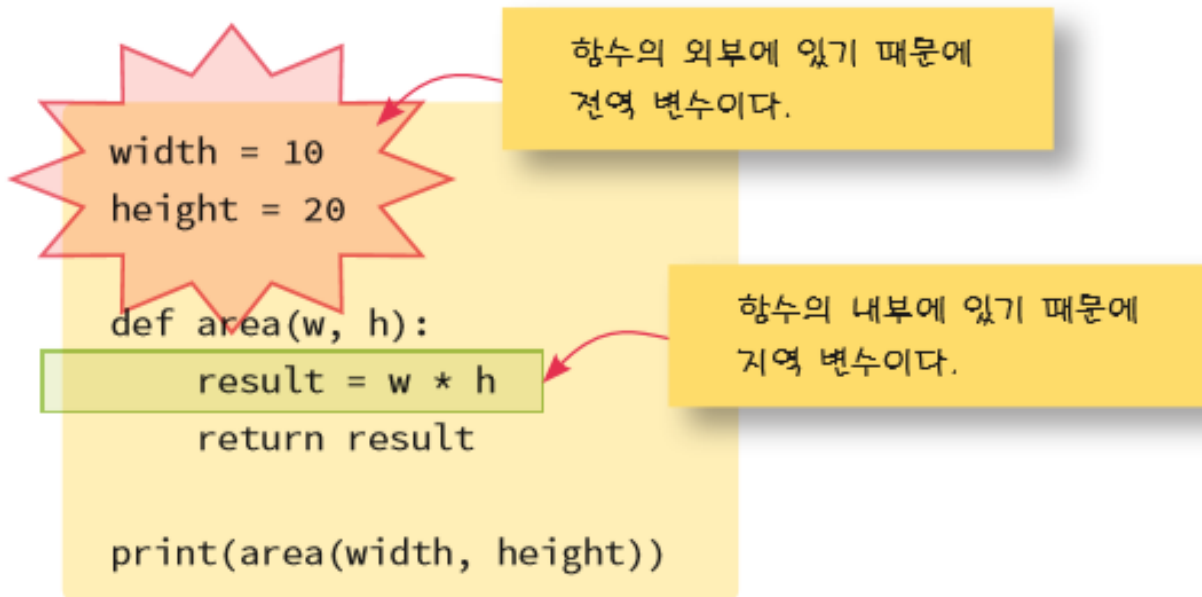
```
list = [1, 2, 3, 4, 5]  
print(list)  
modify2(list)  
print(list)
```



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

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

# 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!

# 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]

# Lab: 상수

- 파이를 전역 변수로 선언하고 이것을 이용하여서 원의 면적과 원의 둘레를 계산하는 함수를 작성해보자.

원의 반지름을 입력하시오: 10  
원의 면적: 314.159265358979  
원의 둘레: 62.8318530717958

# Solution

```
PI = 3.14159265358979 # 전역 상수
```

```
def main():
```

```
    radius = float(input('원의 반지름을 입력하시오: '))
```

```
    print('원의 면적:', circleArea(radius))
```

```
    print('원의 둘레:', circleCircumference(radius))
```

```
def circleArea(radius):
```

```
    return PI*radius*radius
```

```
def circleCircumference(radius):
```

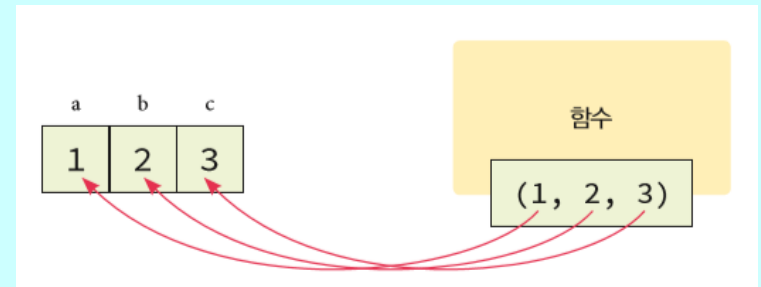
```
    return 2*PI*radius
```

```
main()
```

# 여러 개의 값 반환하기

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

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



1 2 3

# 무명 함수

- 무명 함수는 이름은 없고 몸체만 있는 함수이다. 파이썬에서 무명 함수는 `lambda` 키워드로 만들어진다.

전체적인 구조



`lambda` 인수1 , 인수2 : 수식

인수를 나타낸다.

몸체를 나타낸다.

# 무명 함수의 예

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

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

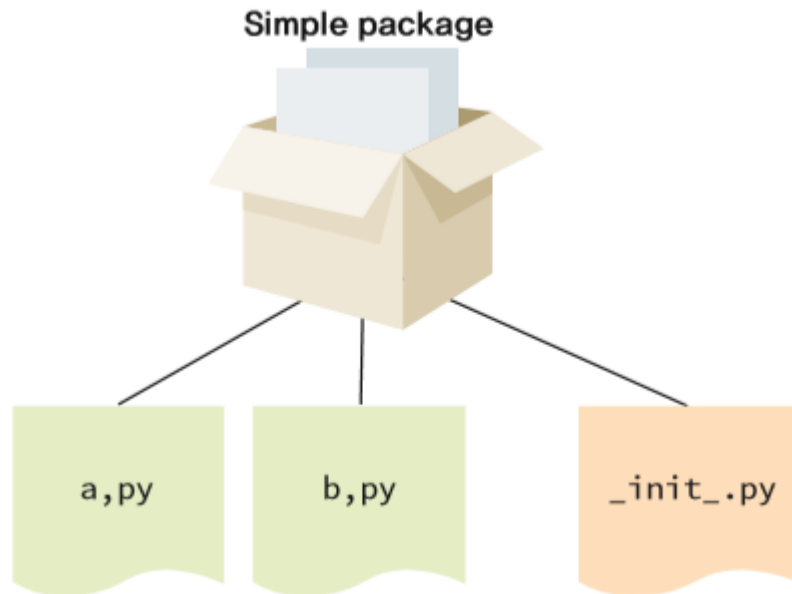
```
print( "정수의 합 :", sum( 20, 20 ))
```

```
정수의 합 : 30
```

```
정수의 합 : 40
```

# 모듈이란?

- 함수나 변수들을 모아 놓은 파일을 모듈(module)



# 모듈 작성

*fibonacci.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

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

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

# 함수를 사용한 프로그램 설계

1. 문제를 한 번에 해결하려고 하지 말고 더 작은 크기의 문제들로 분해한다. 문제가 충분히 작아질 때까지 계속해서 분해한다.
2. 문제가 충분히 작아졌으면 각각의 문제를 함수로 작성한다.
3. 이들 함수들을 조립하면 최종 프로그램이 완성된다.



# 예제

```
def readList():  
    nlist = []  
    flag = True;  
    while flag :  
        number = int(input("숫자를 입력하시오: "))  
        if number < 0:  
            flag = False  
        else :  
            nlist.append(number)  
    return nlist  
  
def processList(nlist):  
    nlist.sort()  
    return nlist  
  
def printList(nlist):  
    for i in nlist:  
        print("성적=", i)
```

# 예제

```
def main():  
    nlist = readList()  
    processList(nlist)  
    printList(nlist)  
  
if __name__ == "__main__":  
    main()
```

숫자를 입력하시오: 30  
숫자를 입력하시오: 50  
숫자를 입력하시오: 10  
숫자를 입력하시오: 90  
숫자를 입력하시오: 60  
숫자를 입력하시오: -1  
성적= 10  
성적= 30  
성적= 50  
성적= 60  
성적= 90

# 핵심 정리

- 함수는 동일한 코드를 재사용하기 위한 것이다. 함수는 `def`로 작성된다.
- 함수 안에서 선언되는 변수는 지역 변수이고 함수의 외부에서 선언되는 변수는 전역 변수이다.
- 함수나 변수들을 모아 놓은 파일을 모듈(module)이라고 한다.

# Q & A

