

새내기 파이썬



6장 함수



함수 목표

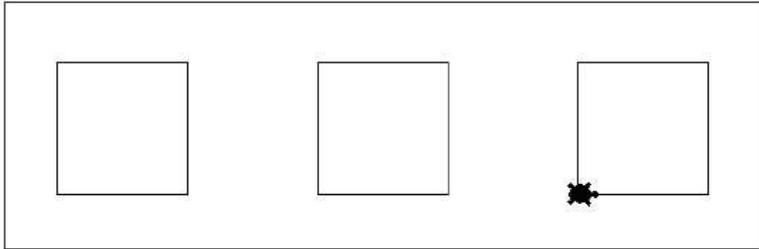
- 반복되는 코드를 함수로 정의할 수 있나요?
- 함수에 데이터를 전달하여 어떤 작업을 시킬 수 있나요?
- 유용한 함수들을 작성하여 사용할 수 있나요?
- 함수로부터 반환되는 값을 받을 수 있나요?
- 함수를 사용하는 장점에 대하여 설명할 수 있나요?





이번장에서 만든 프로그램

20cm 피자 2개의 면적: 2512.0
30cm 피자 1개의 면적: 2826.0



핀 번호를 입력하시오: 1234

1 - 잔액 보기 2- 인출 3- 저금 4- 종료

번호를 선택하시오: 1

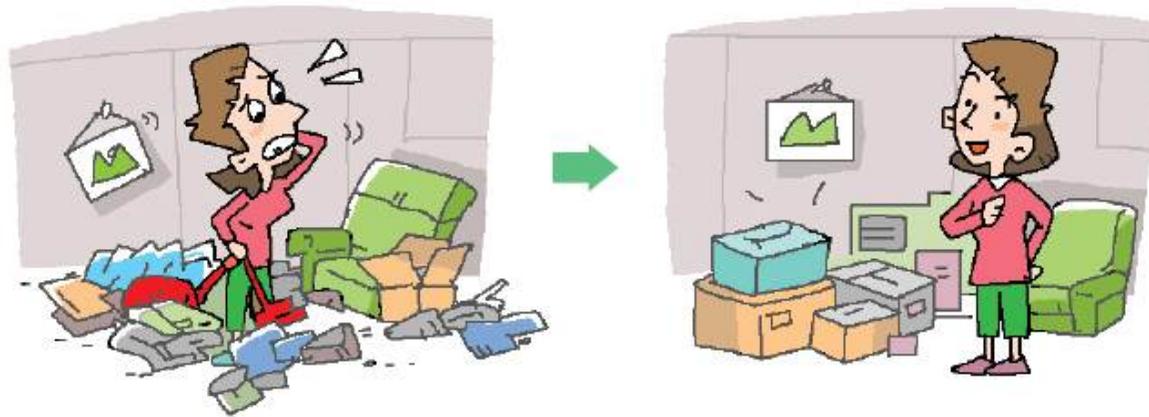
잔액은 0원입니다.

....



코드를 묶는 방법

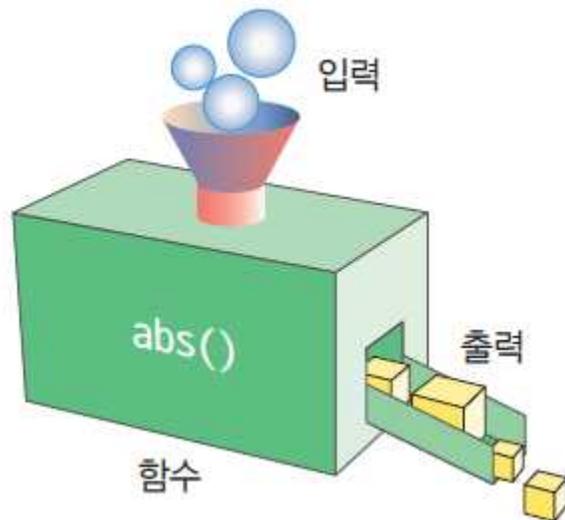
- 관련 있는 코드들을 묶어서 전체 프로그램을 조직화할 필요가 있다.
- 파이썬에서 코드를 묶는 3가지의 방법
 - 함수(function)는 우리가 반복적으로 사용하는 코드를 묶은 것으로 프로그램의 빌딩 블록과 같다.
 - 객체(object)는 서로 관련 있는 변수와 함수를 묶는 방법이다.
 - 모듈(module)은 함수나 객체들을 소스 파일 안에 모은 것이다





함수

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



함수는 입력을 받아서 처리한 후에 출력하는 상자와 같습니다.





함수의 필요성

- 프로그램을 작성하다 보면 동일한 처리를 반복해야 하는 경우가 많이 발생한다.
- 이런 경우에는, 이미 작성한 코드를 재사용하여 사용할 수 있으면 정말 좋을 것이다.



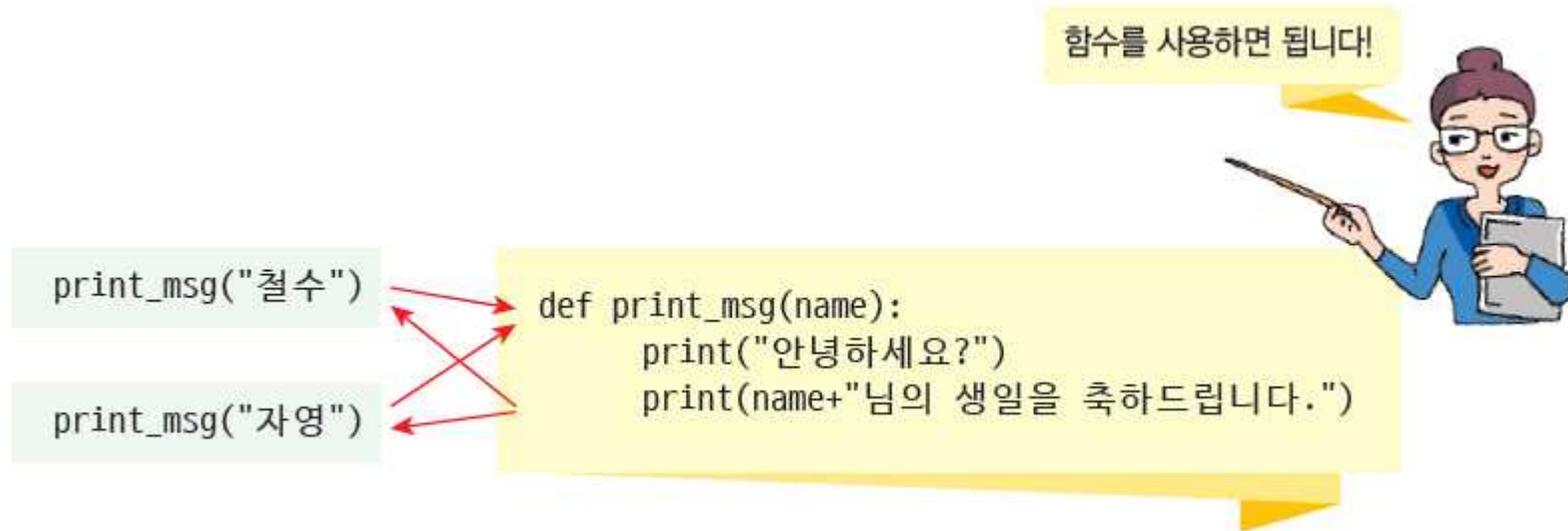
```
print("안녕하세요?")  
print("철수님의 생일을 축하드립니다.")
```

```
print("안녕하세요?")  
print("자영님의 생일을 축하드립니다.")
```



함수를 사용하는 경우

- 함수를 이용하면 우리가 여러 번 반복해야 되는 처리 단계를 하나로 모아서 필요할 때 언제든지 호출하여 사용할 수 있다.





진가저거 중난침침

1. 코드를 복사하는 것보다 함수를 사용하는 것이 좋은 이유는 무엇인가?
2. 함수를 사용하려면 함수를 _____ 하여야 한다.





함수 작성하고 호출하기

Syntax: 함수 정의

Syntax

```
def 함수이름(매개변수1, 매개변수2, ...):  
    명령문1  
    명령문2
```

함수 이름

매개 변수

함수 헤더

def get_area(radius) :

함수 몸체

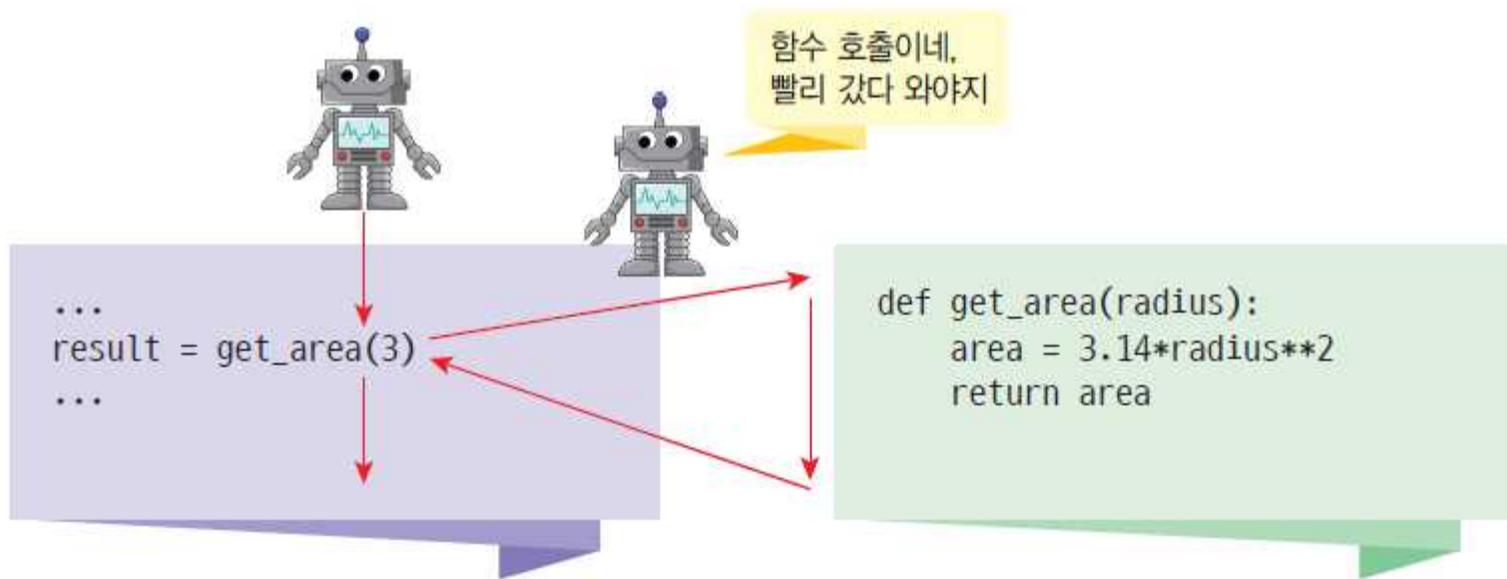
```
    area = 3.14*radius**2  
    return area
```

return 문장은 함수를 종료시키고 결과를 반환한다.



함수 호출

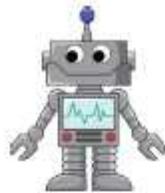
- 함수 호출(function call)이란 `get_area()`과 같이 함수의 이름을 써주는 것이다. 함수가 호출되면 함수 안에 있는 문장들이 실행되며 실행이 끝나면 호출한 위치로 되돌아간다.





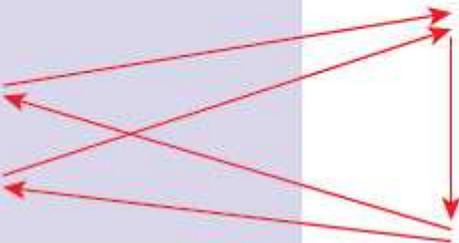
함수는 몇 번이고 호출될 수 있다.

- 함수는 일단 작성되면 몇 번이라도 호출이 가능하다.



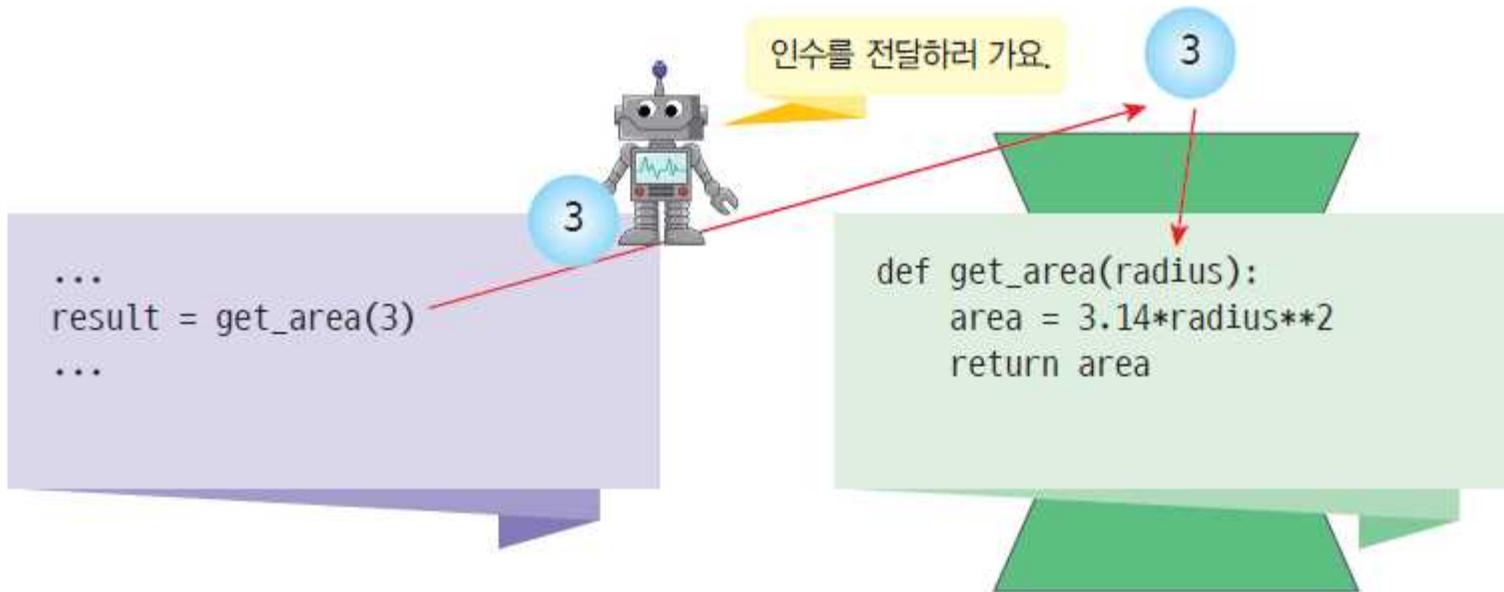
```
...  
x = get_area(3)  
...  
y = get_area(20)
```

```
def get_area(radius):  
    area = 3.14*radius**2  
    return area
```





인수





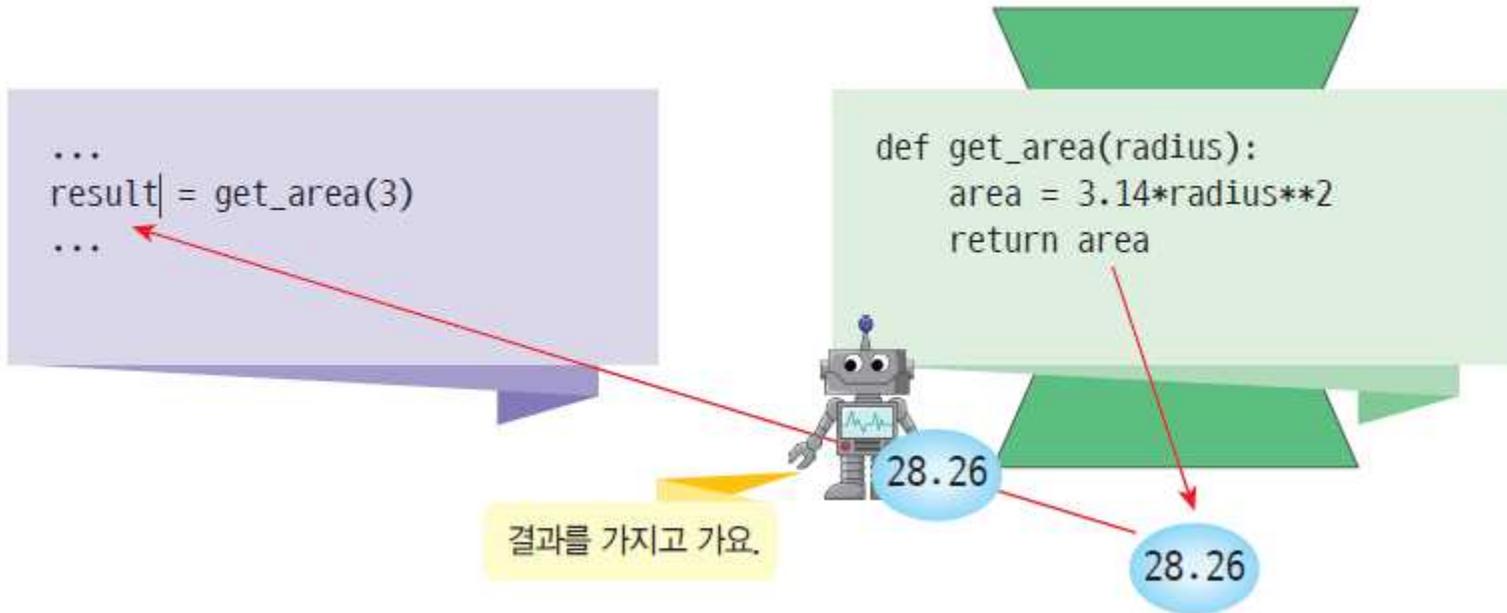
예제

```
def get_area(radius):  
    area = 3.14*radius**2  
    return area  
  
result = get_area(3)  
print("반지름이 3인 원의 면적=", result)
```

반지름이 3인 원의 면적= 28.26



값 반환하기





서로 다른 인수로 호출될 수 있다.

```
def get_area(radius):  
    area = 3.14*radius**2  
    return area
```

```
result1 = get_area(3)  
result2 = get_area(20)
```

```
print("반지름이 3인 원의 면적=", result1)  
print("반지름이 20인 원의 면적=", result2)
```

```
반지름이 3인 원의 면적= 28.26  
반지름이 20인 원의 면적= 1256.0
```



여러 개의 값 반환하기

- 파이썬에서는 함수가 여러 개의 값을 반환할 수 있다. 다음과 같은 형식을 사용한다. 이것은 7장에서 학습하는 튜플을 통하여 이루어진다.

```
def get_input():  
    return 2, 3
```

```
x, y = get_input()           # x는 2이고 y는 3이다.
```



함수의 몸체는 나중에 작성할 수 있다.

- 파이썬에서 함수의 헤더만 결정하고 몸체는 나중에 작성하고 싶은 경우에는 `pass` 키워드를 사용할 수 있다.

```
def sub():  
    pass
```



직접 공간점점

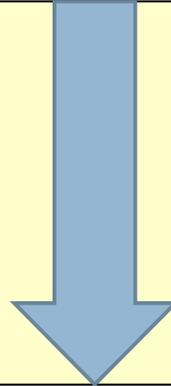
1. 함수에 전달되는 값을 무엇이라고 하는가?
2. 함수 안에서 전달되는 값을 받는 변수를 무엇이라고 하는가?
3. 사용자로부터 2개의 정수를 받아서 반환하는 함수를 작성해보자.
4. 값을 반환하는 키워드는 무엇인가?
5. x 와 y 를 받아서 $x+y$, $x-y$ 값을 반환하는 함수 `addsub(x, y)`를 정의해보자.





여러 함수가 있는 프로그램

```
def get_area(radius):  
    area = 3.14*radius**2  
    return area  
  
result = get_area(3)  
print("반지름이 3인 원의 면적=", result)
```



인터프리터가 프로그램을 읽는 순서

OK



함수의 순서

- 파이썬은 인터프리트 언어이기 때문에 함수의 순서가 중요하다.

```
result = get_area(3)
print("반지름이 3인 원의 면적=", result)

def get_area(radius):
    area = 3.14*radius**2
    return area
```





함수의 순서

- 그러나 함수 내에서는 아직 정의되지 않은 함수를 호출할 수는 있다.

```
def main():  
    result1 = get_area(3)  
    print("반지름이 3인 원의 면적=", result1)
```

```
def get_area(radius):  
    area = 3.14*radius**2  
    return area
```

```
main()
```

함수 안에서는 정의되지 않은 다른 함수를 호출하여도 된다.



직각삼각형 넓이 구하기

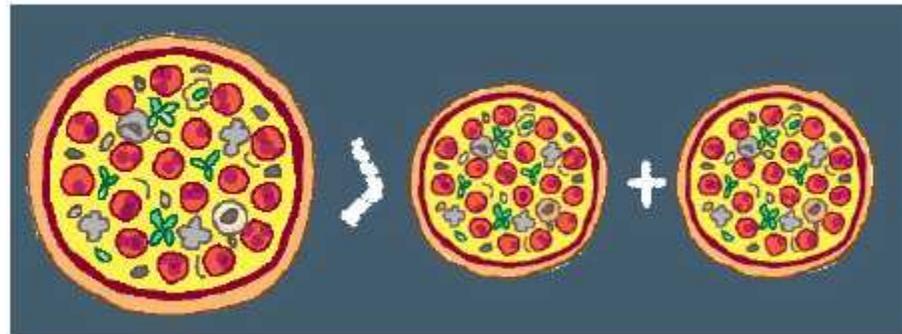
1. 주어진 사각형의 면적을 계산하는 함수 `get_rect_area(w, h)`를 정의해보자.
여기서 `w`는 너비, `h`는 높이이다.
2. `main()` 함수를 정의하고 `main()` 함수 안에서 `get_rect_area()`를 호출해보자.





Lab: 피자 크기 비교

- 원의 반지름을 받아서 피자의 면적을 계산하는 함수를 작성해서 사용해보자.



20cm 피자 2개의 면적: 2512.0
30cm 피자 1개의 면적: 2826.0



Solution:

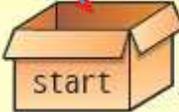
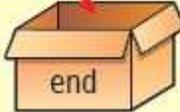
```
def main() :  
    print("20cm 피자 2개의 면적:", get_area(20)+get_area(20))  
    print("30cm 피자 1개의 면적:", get_area(30))  
  
## 원의 면적을 계산한다.  
# @param radius 원의 반지름  
# @return 원의 면적  
#  
def get_area(radius) :  
    if radius > 0 :  
        area = 3.14*radius**2  
    else :  
        area = 0  
    return area  
  
main()
```



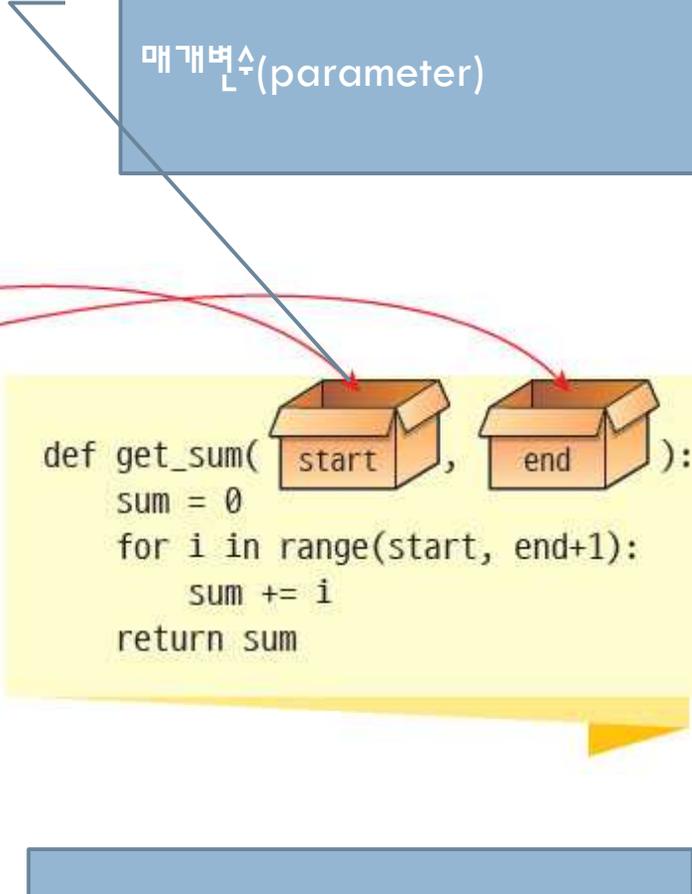
매개변수 전달

매개변수(parameter)

```
value = get_sum(1, 10)
```

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

인자(argument)





서로 다른 인수로 호출될 수 있다.

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

```
# 1과 10이 get_sum()의 인수가 된다.  
x = get_sum(1, 10)
```

```
# 1과 20이 get_sum()의 인수가 된다.  
y = get_sum(1, 20);
```



매개 변수를 변경한다고 해서 인수가 변경되지 않는다.

```
def set_radius(radius):  
    radius = 100  
    return
```

```
r = 20  
set_radius(r)  
print(r)
```

20



디폴트 인수

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

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

greet("영희")
```

```
안녕 영희, 별일업죠?
```



키워드 인수

- 키워드 인수(**keyword argument**)는 키워드 인수는 인수의 이름을 명시적으로 지정해서 값을 매개 변수로 전달하는 방법이다.

```
def sub(x, y, z):  
    print("x=", x, "y=", y, "z=", z)
```

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

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

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

```
>>> sub(x=10, 20, 30)  
sub(x=10, 20, 30)  
^
```

```
SyntaxError: positional argument follows keyword argument
```



Example: 패턴 출력 함수 만들기

- 기호를 가지고 다음과 같은 패턴을 출력하는 함수 `printPattern()`을 작성하고 테스트해보자.

```
>>> printPattern(5, "#")
#####
#####
#####
#####
#####
```



Example: 패턴 출력 함수 만들기

```
import random

def printPattern(rows=5, char="#"):
    for _ in range(rows):
        for _ in range(5):
            print(char, end="")
        print()

printPattern(5, "A")
printPattern(3)
```

```
AAAAA
AAAAA
AAAAA
AAAAA
AAAAA
#####
#####
#####
```



가변 인수

- 파이썬에서는 가변 인수도 허용한다.

```
def varfunc(*args):  
    print(args)
```

```
print("하나의 값으로 호출")  
varfunc(10)
```

```
print("여러 개의 값으로 호출")  
varfunc(10, 20, 30)
```

```
하나의 값으로 호출  
(10,)  
여러 개의 값으로 호출  
(10, 20, 30)
```



예제

```
def add(*numbers) :  
    sum = 0  
    for n in numbers:  
        sum = sum + n  
    return sum  
  
print(add(10, 20))  
print(add(10, 20, 30))
```

```
30  
60
```



예제

- 매개 변수 이름 앞에 이중 별표(**)를 사용하여 가변 길이 키워드 인수를 나타낸다.
- 인수는 딕셔너리 형태로 전달된다

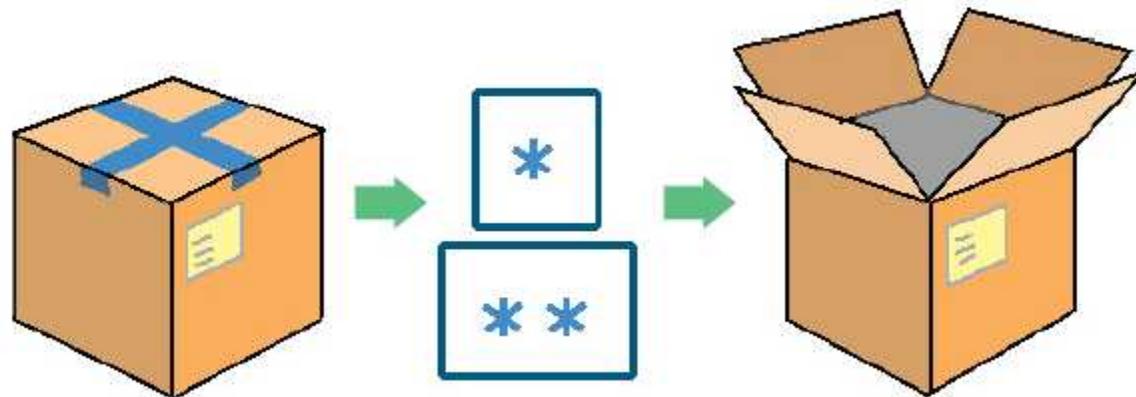
```
def myfunc(**kwargs):  
    result = ""  
    for arg in kwargs.values():  
        result += arg  
    return result  
  
print(myfunc(a="Hi!", b="Mr.", c="Kim"))
```

Hi!Mr.Kim



* 연산자로 언패킹하기

- 단일 별표 연산자 *는 파이썬이 제공하는 모든 반복 가능한 객체 (iterable)을 언패킹할 수 있고 이중 별표 연산자 **는 딕셔너리 객체를 언패킹할 수 있다.





예제

```
>>> alist = [ 1 , 2 , 3 ]  
>>> print(*alist)  
1 2 3
```

```
>>> alist = [ 1 , 2 , 3 ]  
>>> print(alist)  
[1, 2, 3]
```



예제

```
def sum(a, b, c):  
    print(a + b + c)
```

```
alist = [1, 2, 3]  
sum(*alist)
```

6



중간점검

1. 인수와 매개 변수는 다시 한번 설명해보자.
2. 디폴트 인수란 무엇인가? 예를 들어보자.
3. 키워드 인수란 무엇인가? 예를 들어보자.
4. 매개 변수 앞에 * 기호가 있다면 무슨 의미인가?





람다 함수

- 람다 함수(lambda function)는 작은 익명 함수이다.

lambda 인수 : 수식

함수의
인수이다.

함수의 몸체에
해당한다.

```
# 정상적인 파이썬 함수
def func1(x):
    return x + 10

# 람다 함수
func2 = lambda x : x + 10

result = func1(2)
print(result)
result = func2(2)
print(result)
```



람다함수의 예

```
func = lambda x, y : x + y
```

```
result = func(2, 3)
```

```
print(result)
```

```
myList = [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

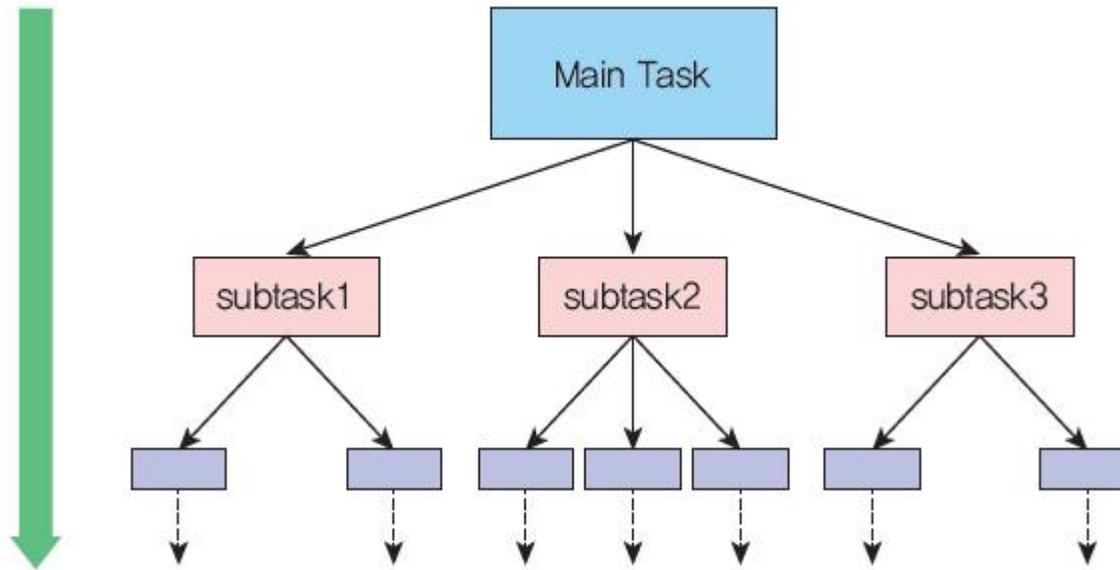
```
result = map(lambda x: x**2, myList)    # [1, 4, 9, 16, 25, 36, 49, 64, 81]
```



함수를 사용하는 이유

- 소스 코드의 중복성을 없애준다.
- 한번 제작된 함수는 다른 프로그램을 제작할 때도 사용이 가능하다.
- 복잡한 문제를 단순한 부분으로 분해할 수 있다.

구조화 프로그래밍





함수를 사용하는 이유

```
# 숫자들의 리스트를 키보드에서 읽어 들이는 함수
def read_list() :
    ...

# 숫자들의 리스트를 크기순으로 정렬하는 함수
def sort_list() :
    ...

# 숫자들의 리스트를 화면에 출력하는 함수
def print_list() :
    ...

def main() :
    ...
    read_list()
    sort_list()
    print_list()
    ...

main()
```



장가저거 공간점점

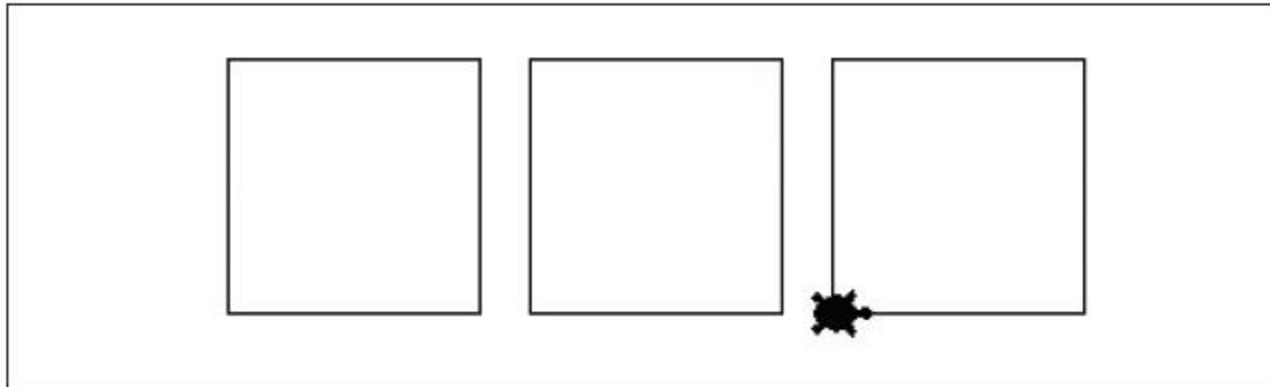
1. 함수 사용의 장점을 이야기해보자.
2. 구조화 프로그래밍이란 무엇인가?





Lab: 사각형을 그리는 함수 작성하기

- 터틀 그래픽에서 정사각형을 그리는 함수를 작성해보자.





Solution:

```
import turtle
t = turtle.Turtle()
t.shape("turtle")

def square(length):          # length는 한변의 길이
    t.down()
    for i in range(4):
        t.forward(length)
        t.left(90)
    t.up()

square(100);                 # square() 함수를 호출한다.
t.forward(120)
square(100);
t.forward(120)
square(100);

turtle.done()
```



Lab: 구조화 프로그래밍 실습

- 온도를 변환해주는 프로그램을 작성해보자.

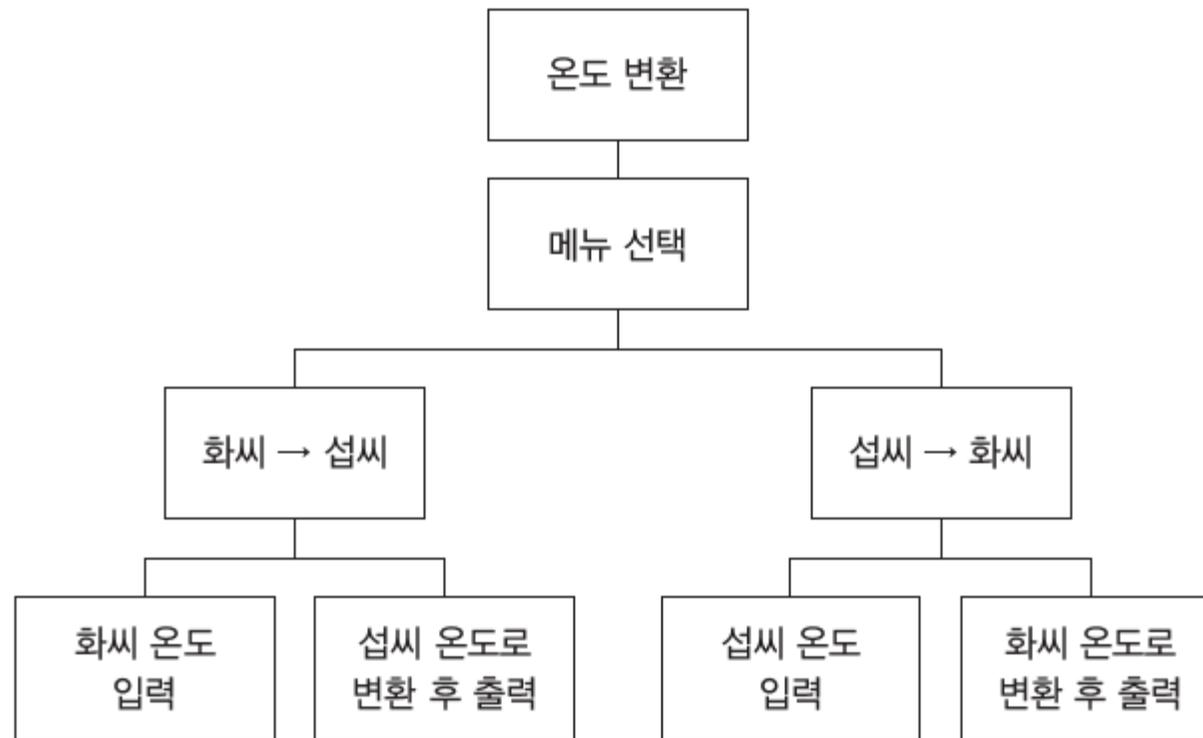
```
1. 섭씨 온도->화씨 온도
2. 화씨 온도->섭씨 온도
3. 종료
메뉴를 선택하세요: 1
```

```
섭씨 온도를 입력하세요: 37.0
화씨 온도 = 98.6
```

```
1. 섭씨 온도->화씨 온도
2. 화씨 온도->섭씨 온도
3. 종료
메뉴를 선택하세요: 3
```



Solution:





Solution:

```
def menu() :  
    print("1. 섭씨 온도->화씨 온도")  
    print("2. 화씨 온도->섭씨 온도")  
    print("3. 종료")  
    selection = int(input("메뉴를 선택하세요: "))  
    return selection
```

```
def ctof(c) :  
    temp = c*9.0/5.0 + 32  
    return temp
```

```
def ftoc(f) :  
    temp = (f-32.0)*5.0/9.0  
    return temp
```

```
def input_f() :  
    f = float(input("화씨 온도를 입력하세요: "))  
    return f
```



Solution:

```
def input_c() :
    c = float(input("섭씨 온도를 입력하시오: "))
    return c

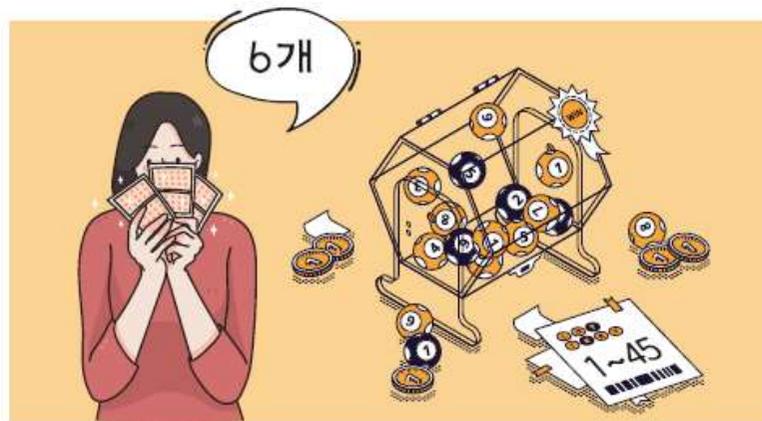
def main() :
    while True:
        index = menu()
        if index == 1 :
            t = input_c()
            t2 = ctof(t)
            print("화씨 온도 = ", t2, "\n")
        elif index == 2 :
            t = input_f()
            t2 = ftoc(t)
            print("섭씨 온도 = ", t2, "\n")
        else :
            break
    main()
```



Lab: 로또 번호 생성하는 함수 작성

- 하나의 예제로 로또 번호를 생성하는 함수를 작성하고 테스트해보자. 로또 번호는 1부터 45까지의 숫자 6개로 이루어진다. 숫자가 중복되면 안 된다.

[36, 43, 26, 45, 24, 11]





Solution:

```
import random

def getLotto() :
    myList = [] # 공백 리스트 생성
    while len(myList) != 6: # 6개의 숫자가 생성될 때까지 반복
        number = random.randint(1, 45) # 1부터 45 사이의 난수 생성
        if number not in myList: # 리스트에 숫자가 없으면
            myList.append(number) # 리스트에 난수를 추가
    return myList # 리스트를 반환한다.

lottoList = getLotto()
print(lottoList)
```



지역 변수와 전역 변수



전역 변수: 함수
외부에서 정의된 변수



지역 변수: 함수
내부에서 정의된 변수





지역 변수

```
def func():  
    x = 100  
    print(x)
```

```
func()
```

지역 변수

100

함수 안에서 생성되는 변수를 지역 변수(local variable)라고 한다. 지역 변수는 함수가 종료되면 사라지게 된다.



전역 변수

```
gx = 100

def func1():
    print(gx)          # 전역 변수 사용 가능

def func2():
    print(gx)          # 전역 변수 사용 가능

func1()
func2()
```

함수의 외부에서 생성된 변수는 프로그램의 어디서나 사용할 수 있다. 이러한 변수를 전역 변수(global variable)라고 한다. 전역 변수는 프로그램이 실행되는 동안, 계속해서 존재한다.



함수 안에서 전역 변수 변경하기

```
gx = 100
```

```
def func1():  
    print("func1() :", gx)
```

```
def func2():  
    gx = 200  
    print("func2() :", gx)
```

```
myfunc1()  
myfunc2()  
print("외부: ", gx)
```

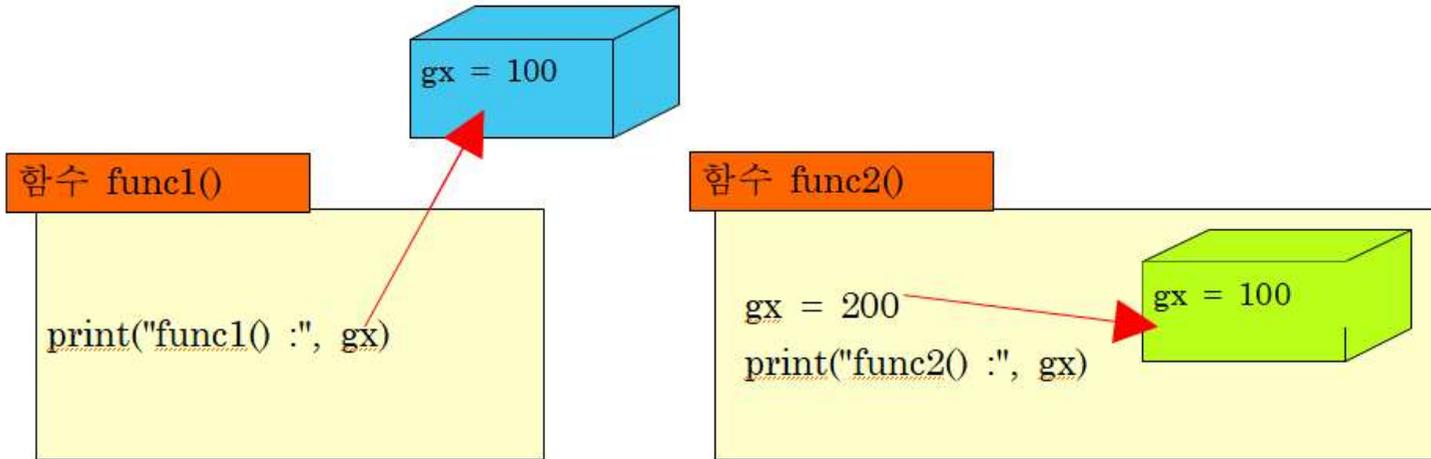
함수 안에서 변수에 값을
저장하면 새로운 지역 변수가
생성된다.

여기서 지역 변수 gx가 생성된다.
지역 변수 gx를 사용한다.

```
func1() : 100  
func2() : 200  
외부: 100
```



함수 안에서 전역 변수 변경하기





함수 안에서 전역 변수 변경하기

```
gx = 100

def func1() :
    print("func1() :", gx)

def func2() :
    global gx
    gx = 200
    print("func2() :", gx)

myfunc1()
myfunc2()
print("외부: ", gx)
```

```
# 전역 변수 gx를 사용하겠음
# 전역 변수 gx가 200으로 변경
```

```
func1() : 100
func2() : 200
외부: 200
```



중간점검 평가점거

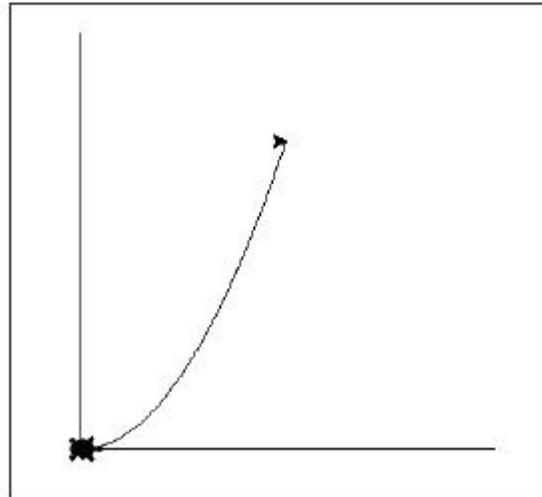
1. 지역 변수와 전역 변수는 어떻게 다른가?
2. 함수 안에서 전역 변수의 값을 읽을 수 있는가?
3. 함수 안에서 전역 변수의 값을 변경하면 어떻게 되는가?





Lab: 함수 그리기

- 함수 $f(x) = x^2 + 1$ 을 계산하는 함수를 작성하고 이 함수를 이용하여 화면에 $f(x)$ 를 그려보자.





Solution:

```
import turtle
t = turtle.Turtle()
t.shape("turtle")
t.speed(0)

def f(x):
    return x**2+1

t.goto(200, 0)
t.goto(0, 0)
t.goto(0, 200)
t.goto(0, 0)

for x in range(150):
    t.goto(x, int(0.01*f(x)))

turtle.done()
```



Mini Project: ATM 구현하기

핀 번호를 입력하시오: 1234

1 - 잔액 보기 2- 인출 3- 저금 4- 종료

번호를 선택하시오: 1

잔액은 0원입니다.

1 - 잔액 보기 2- 인출 3- 저금 4- 종료

번호를 선택하시오: 3

금액을 입력하시오: 10000

잔액은 10000원입니다.

...





이번장에서 배운 것

- 함수가 무엇인지를 학습하였다.
- 인수와 매개변수가 무엇인지를 학습하였다.
- 어떻게 함수로 인수를 전달할 수 있는지를 학습하였다.
- 여러 개의 인수를 함수로 전달하는 방법을 학습하였다.
- 함수가 값을 반환하는 방법을 학습하였다.
- 지역변수와 전역변수의 차이점에 대하여 학습하였다.
- global 키워드를 사용하여 함수 안에서 전역변수를 사용하는 방법을 학습하였다.





Q & A

