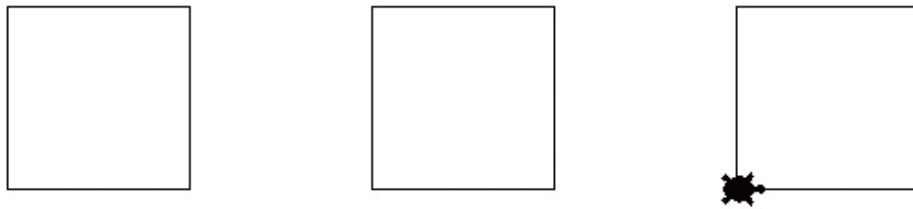




7장 코드를 함수로 모아
봅시다.

이번 장에서 만들 프로그램

(1) 터틀 그래픽에서 사각형을 그리는 함수를 정의하고 사용해본다.

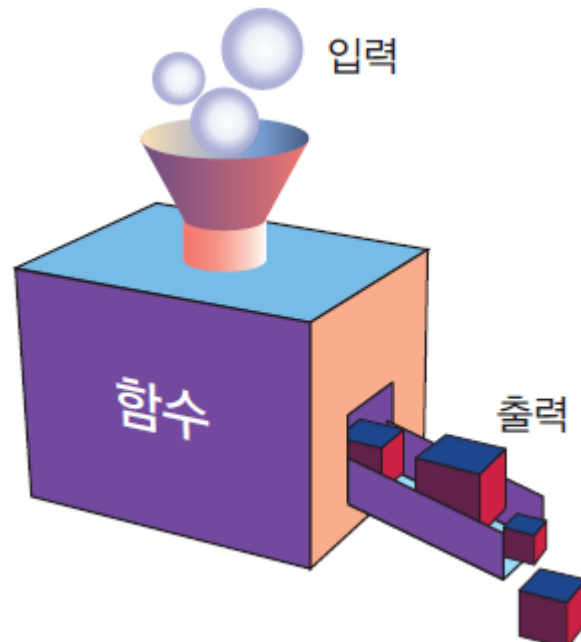


(2) 마우스로 클릭하는 곳에 사각형을 그리는 프로그램을 함수를 이용하여 작성해본다.



함수란?

- 함수는 코드의 묶음에 이름을 붙인 것
- 함수는 입력을 받아서 출력을 내보내는 박스로 생각할 수 있다.



함수 작성하고 호출하기

함수 정의

```
def print_address():  
    print("서울특별시 종로구 1번지")  
    print("파이썬 빌딩 7층")  
    print("홍길동")
```

함수 호출

```
print_address()
```

서울특별시 종로구 1번지
파이썬 빌딩 7층
홍길동

함수의 장점

- 한 번만 함수를 정의하면 언제든지 필요할 때면 함수를 불러서 일을 시킬 수 있다.

```
print_address()  
print_address()  
print_address()
```

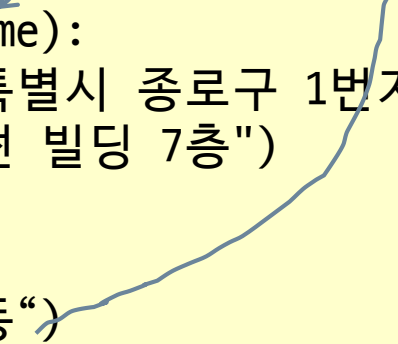
서울특별시 종로구 1번지
파이썬 빌딩 7층
홍길동
서울특별시 종로구 1번지
파이썬 빌딩 7층
홍길동
서울특별시 종로구 1번지
파이썬 빌딩 7층
홍길동

함수에 입력 전달하기

- 우리는 함수에 값(정보)을 전달할 수 있다. 이 값을 인수(argument)라고 한다.



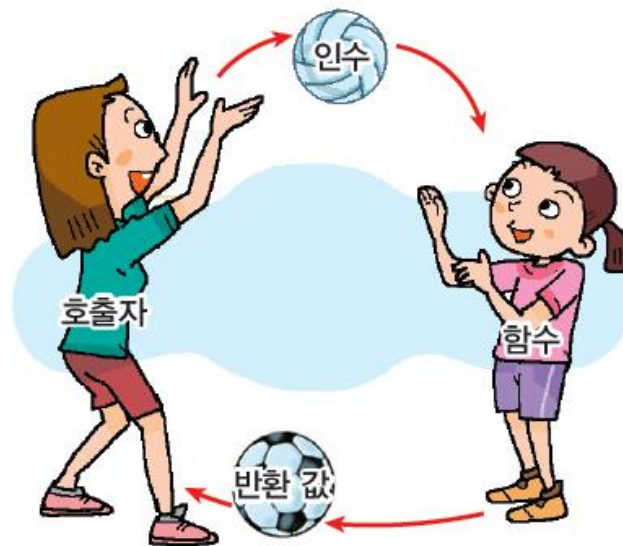
인수 전달



```
def print_address(name):  
    print("서울특별시 종로구 1번지")  
    print("파이썬 빌딩 7층")  
    print(name)  
  
print_address("홍길동")
```


값 반환하기

- 함수는 값을 반환할 수 있다.



값 반환

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

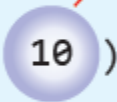




```
c_area = calculate_area(5.0)
```

함수에 여러 개의 입력 전달하기

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

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

get_sum( , )

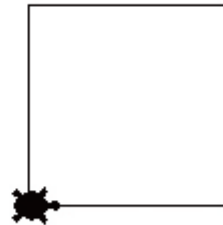
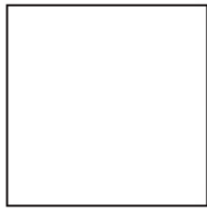
def get_sum( , ):
 sum = 0
 for i in range(start, end+1):
 sum += i
 return sum

Lab: 사각형을 그리는 함수 작성하기

- 정사각형을 그리는 함수는 다음과 같다.

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

- 위의 함수를 호출하여 3개의 정사각형을 그려 보자.



Solution

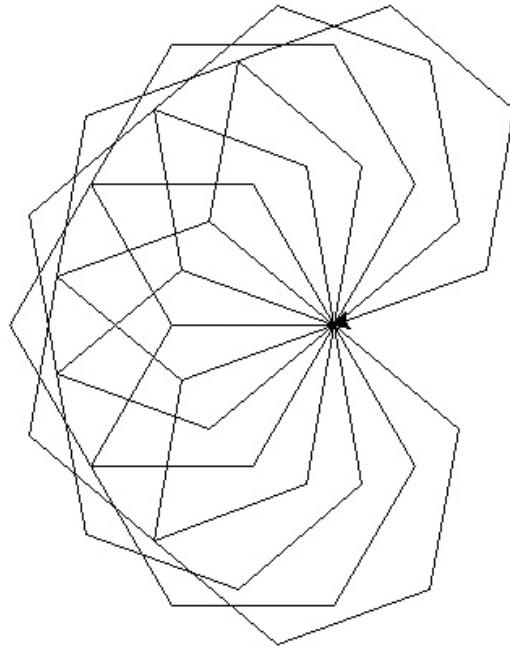
```
import turtle
t = turtle.Turtle()
t.shape("turtle")

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

t.up()           # 펜을 든다.
t.goto(-200, 0)  # (-200, 0)으로 이동한다.
t.down()         # 펜을 내린다.
square(100);     # square() 함수를 호출한다.
t.up()
t.goto(0, 0)
t.down()
square(100);
t.up()
t.goto(200, 0)
t.down()
square(100);
```

Lab: n -각형을 그리는 함수 작성하기

- n -각형을 그리는 함수를 작성하여 다음과 같은 그림을 그려 보자.



Solution

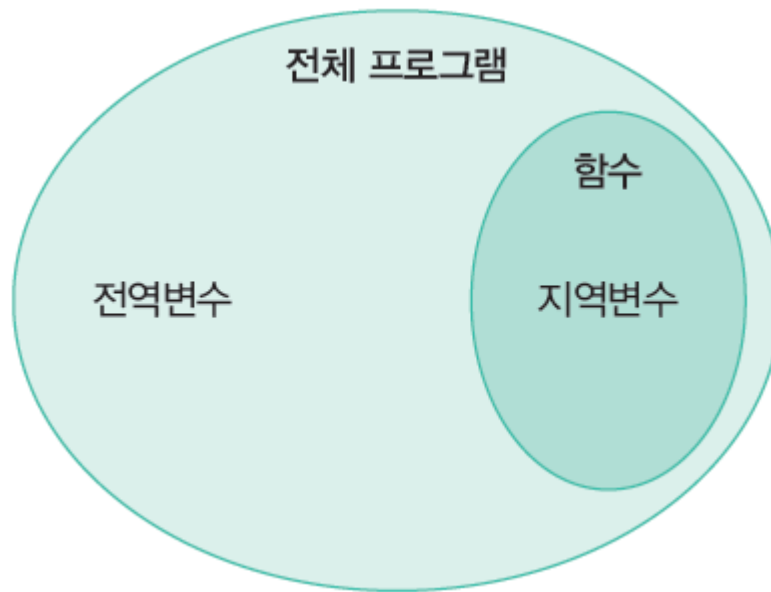
```
import turtle
t = turtle.Turtle()

# n-각형을 그리는 함수를 정의한다.
def n_polygon(n, length):
    for i in range(n):
        t.forward(length)
        t.left(360//n) # 정수 나눗셈은 //으로 한다.

for i in range(10):
    t.left(20)
    n_polygon(6, 100)
```

변수의 종류

- 지역 변수(local variable): 함수 안에서 선언되는 변수
- 전역 변수(global variable): 함수 외부에서 선언되는 변수



지역 변수의 범위

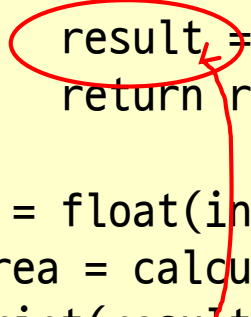
- 지역 변수는 함수 안에서만 사용이 가능하다.
- 아래의 코드에서 지역 변수를 찾아보자.

```
def calculate_area(radius):  
    result = 3.14 * radius**2  
    return result  
  
r = float(input("원의 반지름: "))  
area = calculate_area(r)  
print(result)
```

오류가 없을까?

지역 변수의 범위

```
def calculate_area (radius):  
    result = 3.14 * radius**2  
    return result  
  
r = float(input("원의 반지름: "))  
area = calculate_area(r)  
print(result)
```



원의 반지름: 10

Traceback (most recent call last):

File "C:\Users\sec\AppData\Local\Programs\Python\Python35-32\z.py", line 7, in <module>

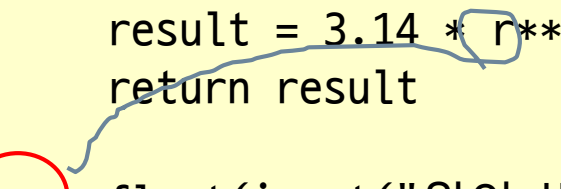
print(result)

NameError: name 'result' is not defined

전역 변수

- 전역 변수는 어디서나 사용할 수 있다.
- 아래의 코드에서 전역 변수를 찾아보자.

```
def calculate_area ():  
    result = 3.14 * r**2  
    return result  
  
r = float(input("원의 반지름: "))  
area = calculate_area()  
print(area)
```



함수 안에서 전역 변수 변경하기

```
def calculate_area (radius):
```

```
    area = 3.14 * radius**2 # 전역변수 area에 계산값을 저장하려고 했다!
```

```
    return
```

```
area = 0
```

```
r = float(input("원의 반지름: "))
```

```
calculate_area(r)
```

```
print(area)
```

원의 반지름: 10

0

>>>

왜 0이 나올까?

함수 안에서 전역 변수 변경하기

```
def calculate_area (radius):  
    area = 3.14 * radius**2 # 전역변수 area에 계산값을 저장하려고 했다!  
    return  
  
area = 0  
r = float(input("원의 반지름: "))  
calculate_area(r)  
print(area)
```

여기서 새로운 지역 변수 area가 생성된다.

함수 안에서 전역 변수 변경하기

- global을 사용하여 전역 변수에 값을 저장한다고 알려야 한다.

```
def calculate_area (radius):  
    global area  
    area = 3.14 * radius**2  
    return  
  
area = 0  
r = float(input("원의 반지름: "))  
calculate_area(r)  
print(area)
```

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

- 키워드 인수는 인수의 이름을 명시적으로 지정해서 전달하는 방법이다.

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

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

Lab: 클릭하는 곳에 사각형 그



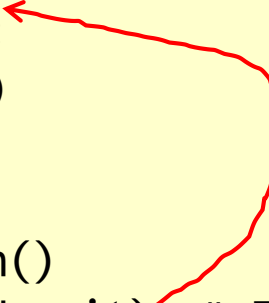
- 사용자가 화면에서 마우스 버튼을 클릭한 경우, 클릭 된 위치에 사각형을 그리는 프로그램을 작성해 보자. 앞에서 작성한 `square()` 함수도 사용한다



콜백 함수

- 이벤트가 발생했을 때, 이벤트를 처리하는 함수를 콜백 함수(callback function)라고 부른다.
- 터틀 그래픽에서도 마우스가 클릭 되었을 때 호출되는 콜백 함수를 등록할 수 있다.

```
def drawit(x, y):  
    t.penup()  
    ...  
    ...  
s = turtle.Screen() # 그림이 그려지는 화면을 얻는다.  
s.onscreenclick(drawit) # 마우스 클릭 이벤트 처리 함수를 등록한다.
```



Solution

```
import turtle # 터틀 그래픽 모듈을 포함한다.

def square(length):
    for i in range(4):
        t.forward(length)
        t.left(90)

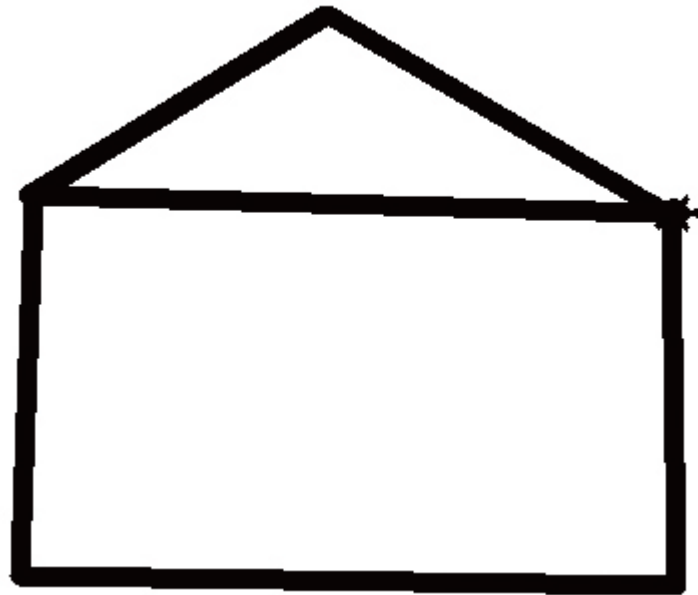
def drawit(x, y):
    t.penup()
    t.goto(x, y)
    t.pendown()
    t.begin_fill()
    t.color("green")
    square(50)
    t.end_fill()

s = turtle.Screen() # 그림이 그려지는 화면을 얻는다.
s.onscreenclick(drawit) # 마우스 클릭 이벤트 처리 함수를 등록한다.
```

Lab: 마우스로 그림 그리기



- 이번 실습에서는 `drawit()` 안에 `goto()`를 넣어서 거북이를 클릭된 위치로 이동시키도록 하자.
- 현재 위치에서 클릭된 위치까지 선이 그려 진다.



Solution

```
import turtle # 터틀 그래픽 모듈을 포함한다.

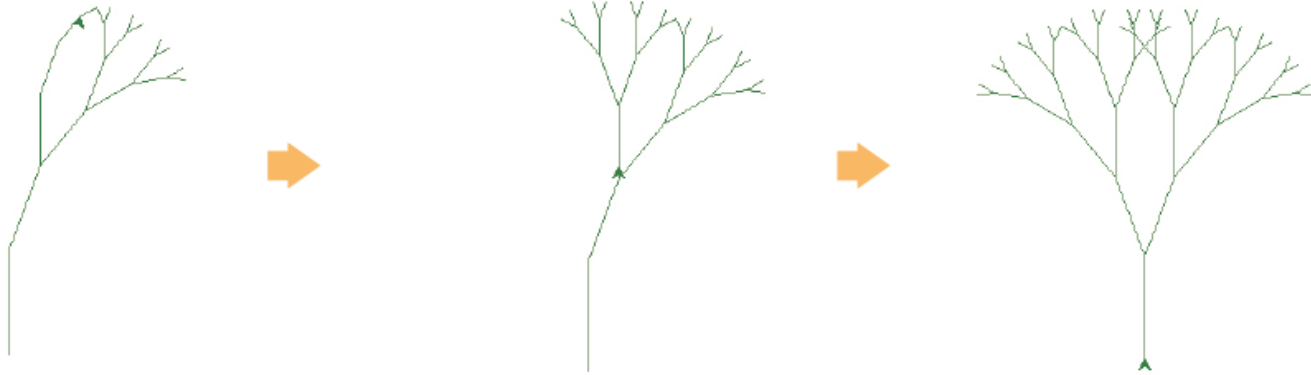
def draw(x, y):
    t.goto(x, y)

t = turtle.Turtle()
t.shape("turtle")
t.pensize(10)
s = turtle.Screen() # 그림이 그려지는 화면을 얻는다.
s.onscreenclick(draw) # 마우스 클릭 이벤트 처리 함수를 등록한다.
```

Lab: 나무 그리기

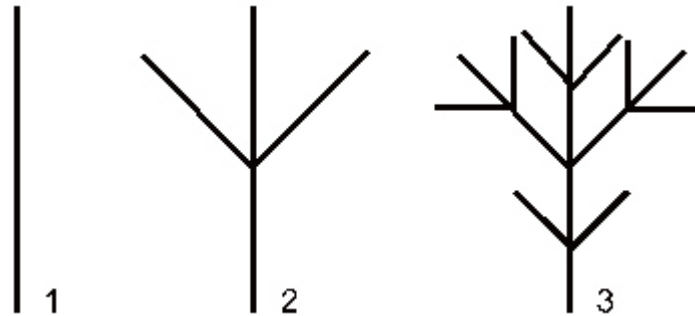


- 지금까지 학습한 내용을 바탕으로 순환적으로 나무를 그리는 프랙탈(fractal) 프로그램을 작성해 보자.
- 함수는 내부에서 다시 자기 자신을 호출할 수 있다. 이것을 순환호출(recursion)이라고 한다



알고리즘

1. 직선을 그린다.
2. 직선의 끝에서 특정한 각도로 2개의 가지를 그린다.
3. 충분한 나뭇가지가 생성될 때까지 각 가지의 끝에서 과정 2를 되풀이 한다.



Solution

```
import turtle

def tree(length):
    if length > 5:
        t.forward(length)
        t.right(20)
        tree(length-15)
        t.left(40)
        tree(length-15)
        t.right(20)
        t.backward(length)

    t = turtle.Turtle()
    t.left(90)
    t.color("green")
    t.speed(1)
    tree(90)
```

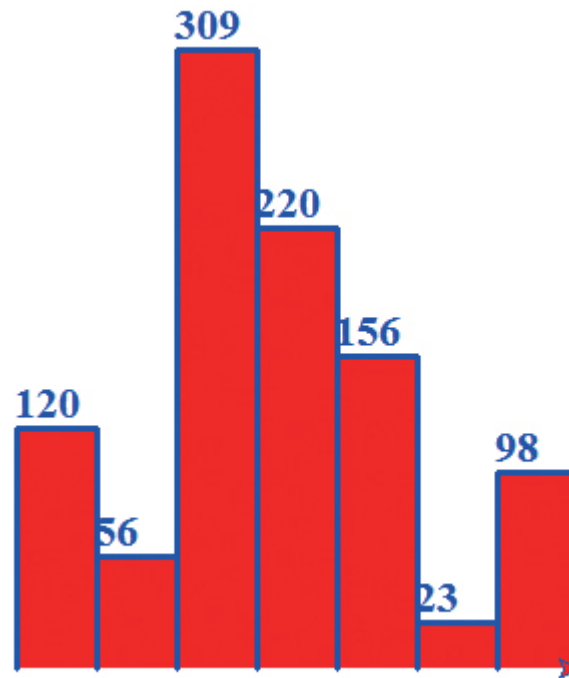
length가 5보다 크면 순환호출을 한다.
거북이가 length 만큼 선을 그린다.
오른쪽으로 20도 회전한다.
(length-15)를 인수로 tree()를 순환 호출한다.
왼쪽으로 40도 회전한다.
(length-15)를 인수로 tree()를 순환 호출한다.
오른쪽으로 20도 회전한다.
length만큼 뒤로 간다. 제자리로 돌아온다.

거북이가 위쪽을 향하게 한다.
선의 색을 녹색으로 한다.
속도를 제일 느리게 한다.
길이 90으로 tree()를 호출한다.

Lab: 막대 그래프 그리기



- 파이썬의 터틀 그래픽을 이용해서 막대 그래프를 그려보자.




```
import turtle

def drawBar(height):
    t.begin_fill()
    t.left(90)
    t.forward(height)
    t.write(str(height), font = ('Times New Roman', 16, 'bold'))
    t.right(90)

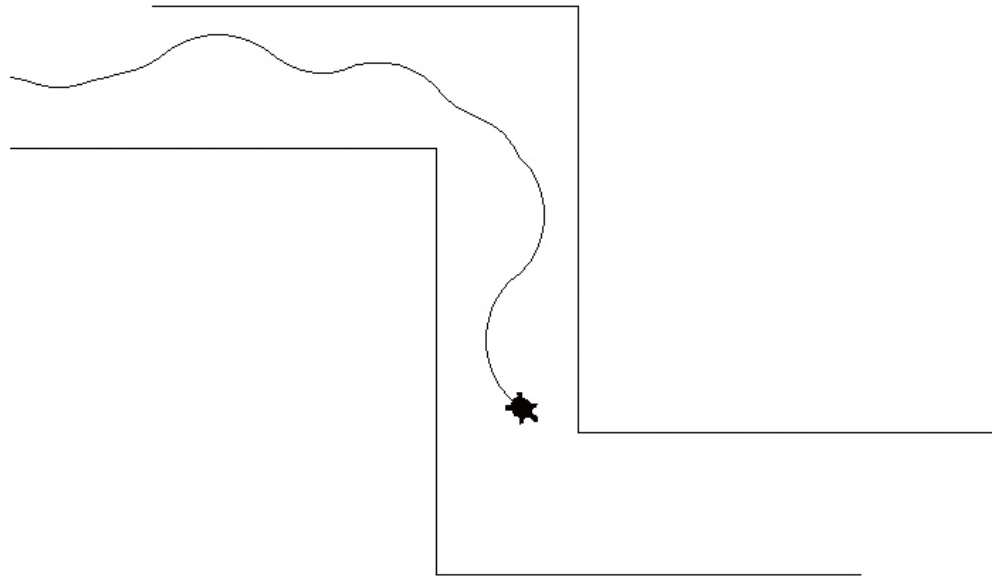
    t.forward(40)
    t.right(90)
    t.forward(height)
    t.left(90)
    t.end_fill()

data = [120, 56, 309, 220, 156, 23, 98]
t = turtle.Turtle()
t.color("blue")
t.fillcolor("red")
t.pensize(3)
for d in data:
    drawBar(d)
```

Lab: 터틀 메이즈 러너



- 화면에 미로를 만들고 거북이가 화살표를 이용하여 미로에 닫지 않게 진행하는 프로그램을 작성해보자.



화살표 키 처리

- 키보드에서 화살표 키가 눌리면 이벤트가 발생하고 이 이벤트를 처리하는 함수는 다음과 같이 등록한다.

```
def turn_left():  
    t.left(10)  
    t.forward(10)  
  
def turn_right():  
    t.right(10)  
    t.forward(10)  
  
t = turtle.Turtle()  
screen = turtle.Screen()  
screen.onkey(turn_left, "Left")  
screen.onkey(turn_right, "Right")
```

Solution

```
import random
import turtle

def draw_maze(x, y):
    for i in range(2):
        t.penup()
        if i==1 :
            t.goto(x+100, y+100)
        else:
            t.goto(x, y)
        t.pendown()
        t.forward(300)
        t.right(90)
        t.forward(300)
        t.left(90)
        t.forward(300)
```

Solution

```
def turn_left():  
    t.left(10)  
    t.forward(10)  
  
def turn_right():  
    t.right(10)  
    t.forward(10)  
  
t = turtle.Turtle()  
screen = turtle.Screen()  
t.shape("turtle")  
t.speed(0)
```

Solution

```
draw_maze(-300, 200)
screen.onkey(turn_left, "Left")
screen.onkey(turn_right, "Right")

t.penup();
t.goto(-300, 250)
t.speed(1)
t.pendown();
screen.listen()
screen.mainloop()
```

이번 장에서 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.



Function(Lambda)

40

□ 구문

- ▣ **lambda 인수 : 식**
- ▣ 한 줄 짜리 축약형 함수, 자동 return

□ (예1)

```
▣ g = lambda x, y : x * y  
    print(g(2,3))
```

□ (예2)

```
▣ print((lambda x, y : x + y)(2,3))
```


Q & A

