

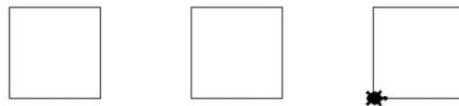
7. 함수

학습 내용

- 함수의 개념을 학습합니다.
- 함수를 작성하는 방법을 학습합니다.
- 함수를 호출하여 사용하는 방법을 학습합니다.
- 모듈의 개념을 학습합니다.

여기에서 만들 프로그램

- 터틀 그래픽에서 사각형을 그리는 함수를 정의하고 사용해 본다.

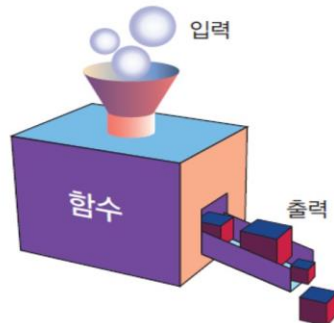


- 마우스로 클릭하는 곳에 사각형을 그리는 프로그램을 함수를 이용하여 작성해본다.



함수 (function)

- 함수는 일을 수행하는 코드 블록에 이름을 붙인 것이다.
- 함수는 입력을 받아서 출력을 내보내는 박스로 생각할 수 있다.



예제

1. $n!$ 구하는 함수
2. n 번째 피보나치 숫자 구하는 함수
3. 소수 판별 함수
4. 윤년 판단 함수

3

함수 작성하고 호출하기

- 함수 정의 →

```
def print_address():  
    print("서울 특별시 종로구 1번지")  
    print("파이썬 빌딩 7층")  
    print("홍길동")
```
- 함수 호출 →

```
print_address()
```

● 실행 결과

The screenshot shows a Python 3.6.5 Shell window with the following content:

```
Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
서울 특별시 종로구 1번지
파이썬 빌딩 7층
홍길동
>>> |
```

The status bar at the bottom right indicates 'Ln: 40 Col: 4'.

4

함수의 장점

- 한 번만 함수를 정의하면 언제든지 필요할 때 함수를 불러 실행시킬 수 있다.

●func1.py

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

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

```
Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
서울 특별시 종로구 1번지
파이썬 빌딩 7층
홍길동
서울 특별시 종로구 1번지
파이썬 빌딩 7층
홍길동
서울 특별시 종로구 1번지
파이썬 빌딩 7층
홍길동
>>>
```

5

함수에 입력 전달하기

●func2.py

함수에서 값(정보)을 받을 수 있다.
받는 변수를 **매개변수(Parameter)**라고 한다.

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

```
print_address("홍길동")
```

함수에 값(정보)을 전달할 수 있다.
이 값을 **인수(Argument)**라고 한다.



```
Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
서울 특별시 종로구 1번지
파이썬 빌딩 7층
홍길동
>>>
```

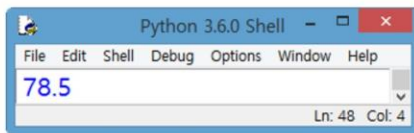
6

값 반환하기

● global.py, func3.py

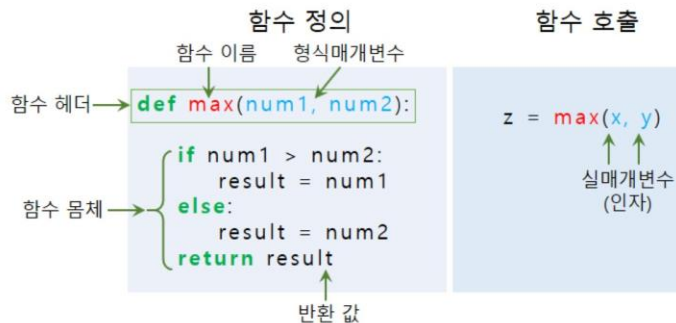
```
def calculate_area (radius):
    area = 3.14 * radius**2
    return area

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



7

함수 정의와 인자를 가진 함수의 호출



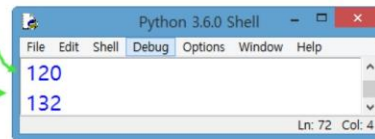
실습 - P01 : n! 계산 함수

```
n = int(input("숫자 입력 : "))
result = 1
for i in range(1, n+1):
    result *= i
print(result)
```

함수

```
def f(n):
    result = 1
    for i in range(1, n+1):
        result *= i
    return result

m = int(input("숫자 입력 : "))
print(f(m))
print(f(3) * 2 + f(5))
```



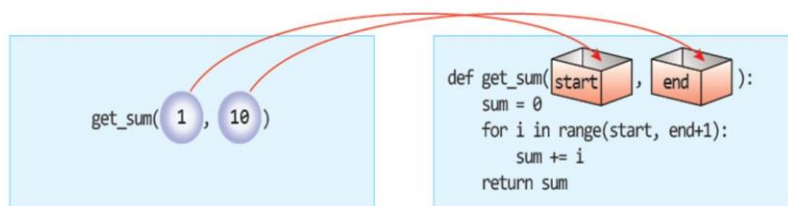
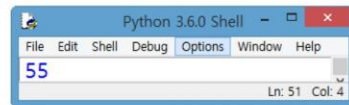
9

실습 - P02 : 함수에 여러 개의 입력 전달하기

●get_sum.py

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

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

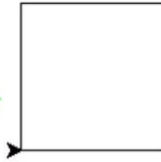


10

실습 - P03 : 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.



11

실습 - P03 : 사각형을 그리는 함수 작성하기

- lab1.py

```
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()
t.goto(-200, 0)
t.down()
square(100);
t.up()
t.goto(0, 0)
t.down()
square(100);
t.up()
t.goto(200, 0)
t.down()
square(100);
```



펜을 든다.
(-200, 0)으로 이동한다.
펜을 내린다.
square() 함수를 호출한다.

12

실습 - P04 : 사각형을 그리는 함수 작성하기

터틀을 이동하는 함수를
추가해 본다.

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

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

def move(x, y): # x, y 좌표로 이동
    t.up()
    t.goto(x, y)
    t.down()

move(-200, 0)
square(100)
move(0, 0)
square(100)
move(200, 0)
square(100)
```

13

실습 - P05

●square() 함수의 인수

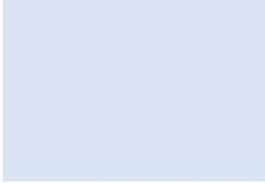
- x 좌표, y좌표
- length : 한 변의 길이

●square(x, y, length) 를 사용하여 원하는 위치에 사각형을 그리도록 프로그램을 변경하세요.

실습 - P05

```
import turtle

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

def square(x, y, length):
    

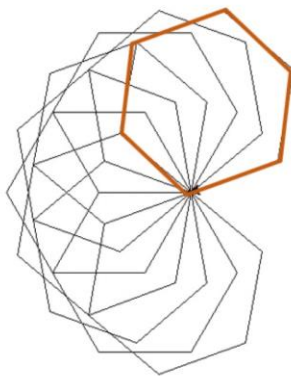
square(-200, 0, 100)
square(0, 0, 100)
square(200, 0, 100)
```

2019-04-28

© Chang Seung Kim - All rights reserved

15

실습 - P06 : n-각형을 그리는 함수 작성하기



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

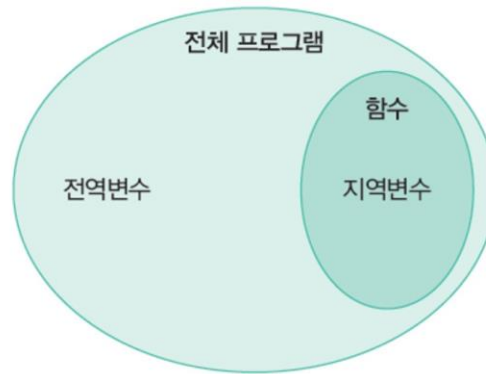
# n-각형을 그리는 함수를 정의한다.
def n_polygon(n, length):
    for i in range(n):
        t.forward(length)
        t.left(360//n) # 정수 나눗셈

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

16

변수의 종류

- 지역 변수: 함수 안에서 선언되는 변수
- 전역 변수: 함수 외부에서 선언되는 변수



17

지역 변수의 범위

- 지역 변수는 함수 안에서만 사용이 가능하다.

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

r = float(input("원의 반지름: "))
area = calculate_area(r)
print(result)
```

원의 반지름: 5.0

Traceback (most recent call last):

File "C:/Users/Myung/AppData/Local/Programs/Python/Python36-32/test.py", line 7, in <module>
 print(result)

NameError: name 'result' is not defined

18

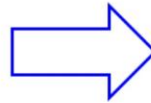
변수의 유효 범위

```
def f(n):
    print("f, ", n)
    n += 1
```

```
def g(n):
    n = 5
    print("g, ", n)
    n += 1
```

```
def h(n):
    print("h, ", n)
    n += 1
```

```
n = 1
print("main, ", n)
f(n)
print("main, ", n)
g(n)
print("main, ", n)
h(n)
print("main, ", n)
```



```
main, 1
f, 1
main, 1
g, 5
main, 1
h, 1
main, 1
```

19

전역 변수

- 어디서나 사용할 수 있다.

- global.py

```
def calculate_area():
    result = 3.14 * r**2
    return result
```

전역변수 r을 사용한다.

```
r = float(input("원의 반지름: "))
area = calculate_area()
print(area)
```

```
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
원의 반지름: 5.0
78.5
>>>
```

Ln: 14 Col: 4

- r, area 는 함수 외부에서 생성 → 전역변수

20

함수 안에서 전역 변수의 값 변경하기

●전역변수에 값을 저장하는 방법은?

●global1.py

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

area = 0
r = float(input("원의 반지름: "))
calculate_area(r)
print(area)
```

전역변수 area에 계산값을 저장하려고 했다!

왜 0이 나올까?

```
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
원의 반지름: 5.0
0
>>>
```

Ln: 18 Col: 4

함수 안에서 변수에 값을 저장하면
새로운 지역변수가 만들어진다.

21

실습 - P07 : 함수 안에서 전역 변수의 값 변경하기

●global을 사용하여 전역 변수에 값을 저장한다고 알려야 한다.

●global2.py

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

area = 0
r = float(input("원의 반지름: "))
calculate_area(r)
print(area)
```

```
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
원의 반지름: 5.0
78.5
>>>
```

Ln: 22 Col: 4

22

실습 - P08 : 디폴트 인수

●파이썬에서는 함수의 매개 변수가 기본값을 가질 수 있다.

- 이것을 디폴트 인수(default argument)라고 한다.
- default.py

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

greet("홍길동")
```



```
Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
안녕 홍길동, 별일없죠?
>>>
```

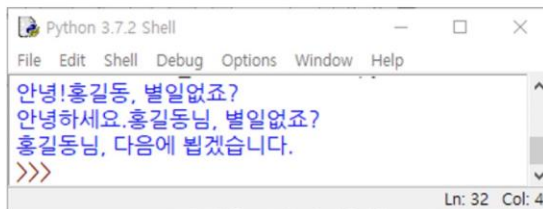
23

실습 - P08-1 : 디폴트 인수

●실습 P08 을 다음과 같이 함수를 호출하였을 때 출력되도록 수정하십시오

```
def greet(                                     ):

    greet("홍길동")
    greet("홍길동님", "안녕하세요.")
    greet("홍길동님", "", "다음에 뵙겠습니다.")
```



```
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
안녕!홍길동, 별일없죠?
안녕하세요.홍길동님, 별일없죠?
홍길동님, 다음에 뵙겠습니다.
>>>
```

- 주의 : 디폴트 인수 뒤에는 일반 인수가 올수 없다.

실습 - P09 : 키워드 인수

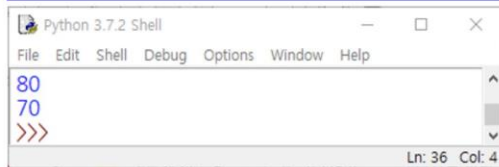
- 키워드 인수는 인수의 이름을 명시적으로 지정해서 전달하는 방법이다.

- keyword.py

```
def calc(x, y, z):
    return x+2*y+z

a = calc(y=20, x=10, z=30)
print(a)

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



```
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
80
70
>>>
Ln: 36 Col: 4
```

25

콜백 함수 (callback function)

- 콜백 함수

- 다른 함수에 인수로 전달되어 실행되는 함수
- 터틀 그래픽에서도 마우스가 클릭 되었을 때 호출되어, 그 이벤트를 처리하도록 콜백 함수를 정의하여 사용할 수 있다.

```
def drawit(x, y):
    ...
    ...
    ...
s = turtle.Screen()
s.onscreenclick(drawit)
```

x와 y는 마우스가 클릭 된 위치이다.

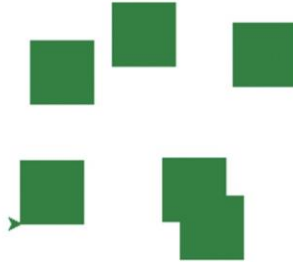
사용자가 화면에서 마우스를 클릭하면 처리하는 함수를 등록한다.

26

실습 - P10 : 클릭하는 곳에 사각형 그리기

● 사용자가 화면에서 마우스 버튼을 클릭한 경우, 클릭 된 위치에 사각형을 그리는 프로그램을 작성해 보자.

● 앞에서 작성한 square() 함수도 사용한다



27

실습 - P10 : 클릭하는 곳에 사각형 그리기

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

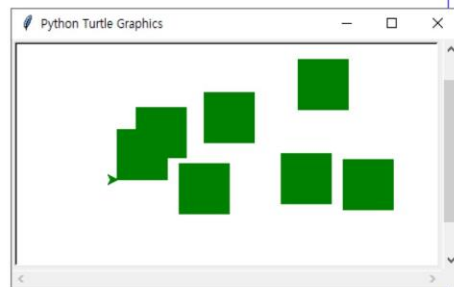
터틀 그래픽 모듈을 포함한다.

```
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()
s.onscreenclick(drawit)
```

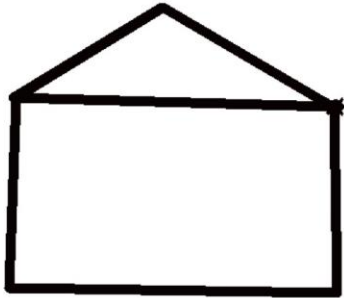
그림이 그려지는 화면을 얻는다.
마우스 클릭 이벤트 처리 함수를 등록한다.



28

실습 - P11 : 마우스로 그림 그리기

- 사용자가 마우스를 이동하여 그림을 그리는 프로그램을 작성하자.
 - draw() 안에 goto()를 넣어서 거북이를 클릭된 위치로 이동시키도록 하자.
 - 현재 위치에서 클릭된 위치까지 선이 그려 진다.



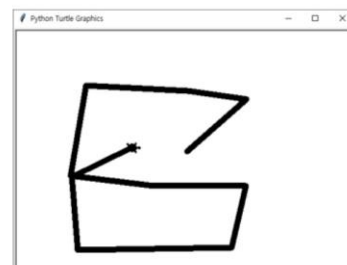
29

실습 - P11 : 마우스로 그림 그리기

```
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()
s.onscreenclick(draw)                        # 그림이 그려지는 화면을 얻는다.
                                              # 마우스 클릭 이벤트 처리 함수를 등록한다.
```



30

실습 - P11 : 마우스로 그림 그리기

```
import turtle

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

t = turtle.Turtle()
t.shape("turtle")
t.pensize(10)
s = turtle.Screen()
s.onscreenclick(draw)
```



```
import turtle

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

t = turtle.Turtle()
t.shape("turtle")
t.pensize(10)
s = turtle.Screen()
s.onscreenclick(t.goto)
```

31

실습 - P12 : 마우스로 그림 그리기

- 그리지 않고, 이동만 할 때, 키보드의 up 키를 누르고, 다시 그리기를 할 때는 down 키를 눌러 그리기

```
import turtle

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

t = turtle.Turtle()
t.shape("turtle")
t.pensize(10)

s = turtle.Screen()
s.onscreenclick(draw)

s.onkey(t.penup, "Up")
s.onkey(t.pendown, "Down")
s.listen()
```

마우스 클릭 이벤트 처리 함수를 등록한다.

키보드 이벤트 처리 함수를 등록한다.

키보드 이벤트 처리 함수를 등록한다.

키보드 이벤트를 기다린다.



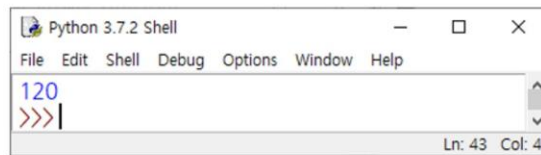
32

실습 - P13 : 재귀 함수 (recursive function)

● 자기 자신을 호출하는 함수

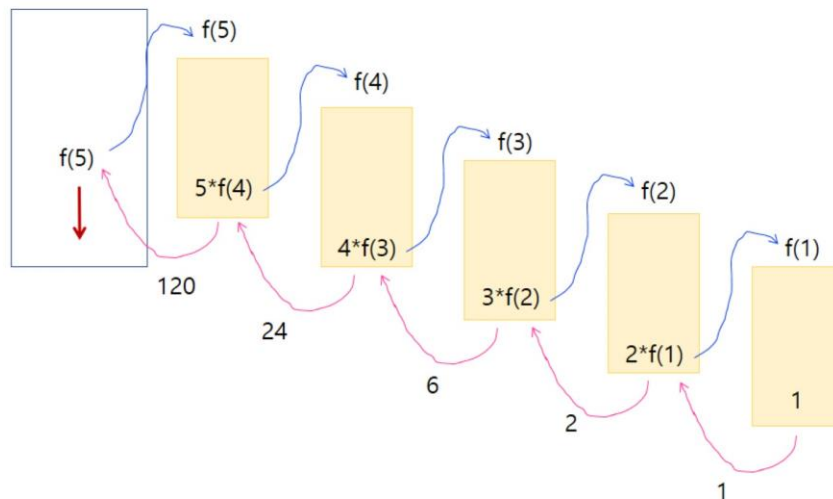
```
def f(n):
    if n == 1:
        return n
    else:
        return n * f(n-1)

print(f(5))
```



33

실습 - P13 : 재귀 함수 (recursive function)

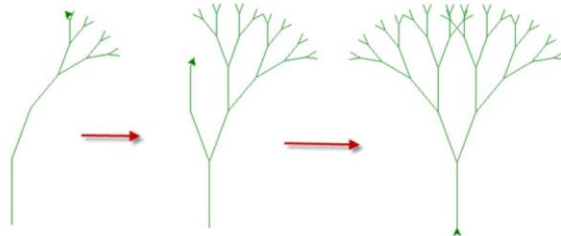


34

실습 - P14 : 나무 그리기

●순환적으로 나무를 그리는 프랙털(fractal) 프로그램을 작성해 보자.

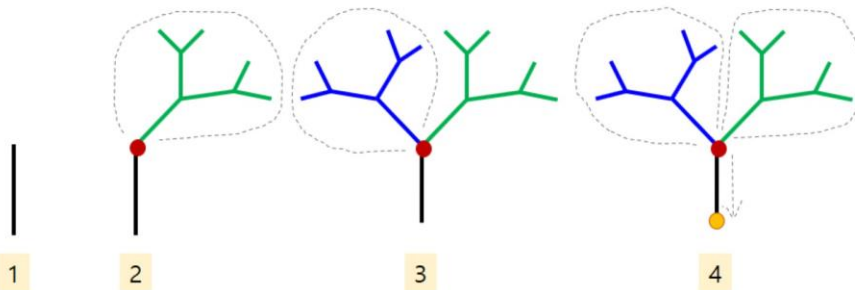
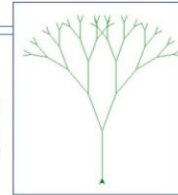
●순환 호출(recursion)을 사용하여 작성한다.



35

알고리즘 (algorithm)

1. 직선을 그린다.
2. 직선의 끝에서 녹색 가지를 다 그린 후 빨간점 위치로 돌아온다.
3. 그 자리에서 파란색 가지를 다 그린 후 빨간점 위치로 돌아온다.
4. 노란색 점 있는 곳으로 돌아온다.



36

실습 - P14 : 나무 그리기

```
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)
    else:
        # 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.

t = turtle.Turtle()
t.left(90)
# 거북이가 위쪽을 향하게 한다.

t.color("green")
# 선의 색을 녹색으로 한다.
t.speed(1)
# 속도를 제일 느리게 한다.
tree(90)
# 길이 90으로 tree()를 호출한다.
```

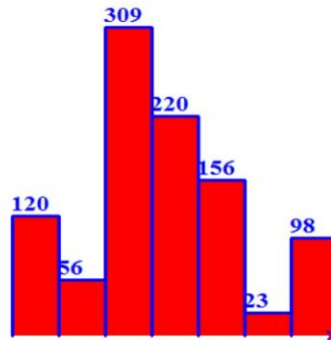
37

실습 - P15 : 풍성한 나무 그리기

- 실습-P14는 한 줄기에 가지가 2개이다.
- 더 풍성한 나무를 그리기 위하여 한 줄기에 가지를 3개 만든다.
 - 가지의 각도를 조정하여 더 넓게 퍼지게 한다.

실습 - P16 : 막대 그래프 그리기

●파이썬의 터틀 그래픽을 이용해서 막대 그래프를 그려 보자.



39

실습 - P16 : 막대 그래프 그리기

```
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()
```

폰트 사이즈

```
data = [120, 56, 309, 220, 156, 23, 98]
```

높이가 height인 막대 그래프를 그린다.

40

실습 - P16 : 막대 그래프 그리기

```
t = turtle.Turtle()
t.color("blue")
t.fillcolor("red")

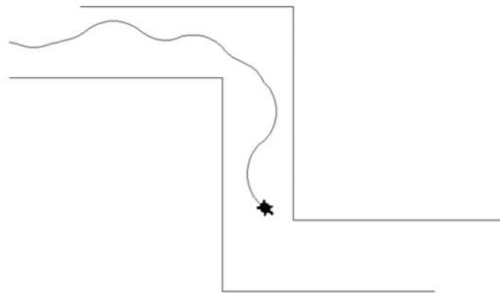
t.pensize(3)

for d in data:
    drawBar(d)
```

41

실습 - P17 : 터틀 메이즈 러너

- 화면에 미로를 만들고 거북이가 화살표를 이용하여 미로에 달지 않게 진행하는 프로그램을 작성해 보자.



42

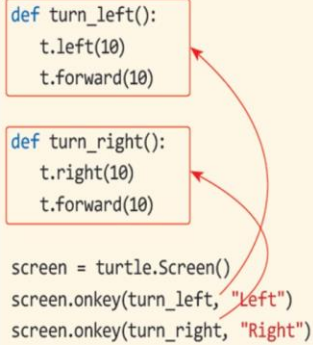
화살표 키 처리

- 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)

screen = turtle.Screen()
screen.onkey(turn_left, "Left")
screen.onkey(turn_right, "Right")
```



43

실습 - P17 : 터틀 메이즈 러너

```
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)

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)

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()
```

44

함수를 정의하는 위치

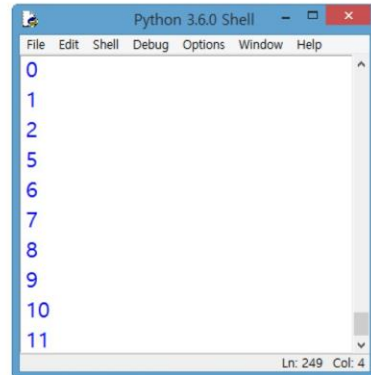
```
def f(n):
    for i in range(n):
        print(i)
```

f(3)

```
def g():
    global k
    for i in range(k):
        print(i+5)
```

k = 7

g()



45

코드 모듈화하기

● 함수

- 코드 중복을 줄여준다.
- 코드 재사용을 위한 목적으로 사용
- 코드를 모듈화하고 프로그램의 품질을 향상시키기 위해 사용

● 모듈(module) 파일

- 함수 정의를 파일 확장자 .py 를 갖는 모듈 파일에 넣을 수 있다.
- 모듈은 재사용을 위해 프로그램 내부로 임포트시킬 수 있다.
- 모듈 파일은 프로그램과 동일한 디렉터리에 있어야 한다.
- 하나의 모듈에 하나 이상의 함수를 가질 수 있다.
 - ◆ 모듈 내부의 각 함수는 서로 다른 이름을 가져야 한다.
- turtle, random, math 등은 파이썬 라이브러리에 정의된 모듈이다.

실습 - P18 : 두 수의 최대 공약수 구하기

●함수명 : gcd

- 매개변수 : num1, num2
- 반환 값 : 최대 공약수
- 저장 파일명 : GCDFunction.py

```
def gcd(num1, num2):
    gcd = 1 # gcd(최대공약수)의 초기값은 1
    k = 2   # 가능한 gcd

    while k <= num1 and k <= num2:
        if (num1 % k == 0 and num2 % k == 0):
            gcd = k # gcd의 값을 갱신
            k += 1

    return gcd
```

2019-04-28

© Chang Seung Kim - All rights reserved

47

실습 - P18 : 두 수의 최대 공약수 구하기

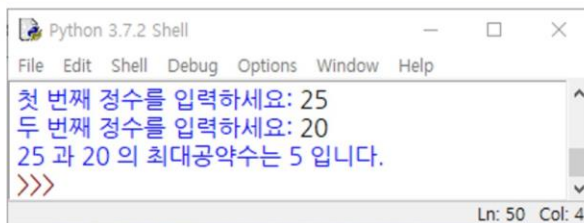
●GCDFunction.py 모듈 파일에 정의된 gcd 함수를 사용하기

- GCDFunction 모듈의 gcd 함수를 임포트하기

```
from GCDFunction import gcd

n1 = int(input("첫 번째 정수를 입력하세요: "))
n2 = int(input("두 번째 정수를 입력하세요: "))

print(n1, "과", n2, "의 최대공약수는", gcd(n1, n2), "입니다.")
```



2019-04-28

© Chang Seung Kim - All rights reserved

48

실습 - P19 : 두 수의 최대 공약수 구하기

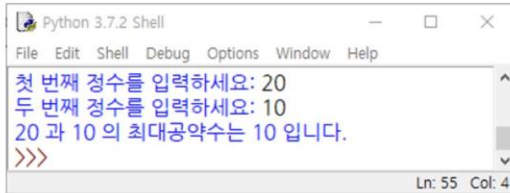
- GCDFunction.py 모듈 파일에 정의된 gcd 함수를 사용하기

- GCDFunction 모듈 전체를 임포트하기

```
import GCDFunction

n1 = int(input("첫 번째 정수를 입력하세요: "))
n2 = int(input("두 번째 정수를 입력하세요: "))

print(n1, "과", n2, "의 최대공약수는", GCDFunction.gcd(n1, n2), "입니다.")
```



2019-04-28

© Chang Seung Kim - All rights reserved

49

과제 : 학점 구하기

- 함수 getGrade() 는 점수를 입력하면 A+, A, B+, B, C+, C, D+, D, F 로 성적을 반환한다.
- 성적을 리스트 score 에 10개를 넣고, getGrade 함수를 사용하여 학점으로 변환한 후 grade 리스트에 저장한 후, 다음과 같이 출력하시오.

```
95 A+
80 B
99 A+
93 A
.....
35 F
59 F
66 D+
```

2019-04-28

© Chang Seung Kim - All rights reserved

50

과제 : 소수함수의 모듈 만들기

●모듈 파일명 : PrimeNumberFunction.py

- 함수명 : isPrime
- 매개변수 : number
- 반환값
 - ◆True : 소수인 경우
 - ◆False : 소수가 아닌 경우

●테스트 프로그램

- PrimeNumberFunction 모듈의 isPrime 을 사용하여 1부터 50까지의 숫자를 차례대로 소수인지 판별한다.
 - ◆반환값이 True 인 경우 : 숫자를 출력
 - ◆반환값이 False 인 경우 : 출력하지 않는다.
 - ◆결과
2, 3, 5, 7, 11, 13, 19, 23, 29, 31, 37, 41, 43, 47

2019-04-28

© Chang Seung Kim - All rights reserved

51

정리

- 함수는 일을 수행하는 코드 블록에 이름을 붙인 것이다
- 함수에 전달하는 값은 **인수(argument)**이고, 이는 **매개변수(parameter)**로 받는다.
- 함수에 인수를 여러 개 전달할 수 있고, 하나의 값을 반환할 수도 있다.
- 지역변수**는 함수 안에서 선언된 변수이고, **전역변수**는 함수 밖에서 선언된 변수이다.
global 키워드를 써서 함수 안에서 전역변수를 사용할 수 있다.
- default 인수**는 매개 변수에 초기값을 주는 것이다.
- 키워드 인수**는 함수의 각 파라미터의 이름을 지정하면서 값을 준다.
- 다른 함수에 인수로 전달되어 실행될 때 전달된 함수를 **콜백함수**라고 한다.
- 재귀 함수**는 함수의 정의 과정에서 자기 자신을 또 호출하는 함수이다.

52