

## 점선 그리기

### 1. 문제

아래 그림과 같이 길이를 입력 받고, 점선을 긋는 함수를 만들어보자.

#### 실행결과



#### 조건

- 1) 길이 10pt 에 점을 하나씩 생성하도록 한다.
- 2) 점은 dot() 함수를 이용한다.
- 3) 함수명 : Dotforward
- 4) 파라미터 : 한변의 길이
- 5) 호출 예:  
Dotforward (100)

```
import turtle
```

```
aaa = turtle.Turtle()
```

```
def Dotforward(x_length):
```

```
    pt_cnt = int(x_length/10) #int
```

```
    for jj in range(pt_cnt):
```

1 / 23

```
aaa.dot(5)
aaa.penup()
aaa.forward(x_length/pt_cnt) #float
aaa.pendown()
```

```
Dotforward (100)
```

2 / 23

### 2. 문제

아래 그림과 같이 길이를 입력 받고, 점선을 긋는 함수를 만들어보자.

#### 실행결과



#### 조건


- 1) 길이 10pt 에 점을 하나씩 생성하도록 한다.
- 2) 점은 실선을 짧게 표현한다
- 3) 함수명 : Dotforward2
- 4) 파라미터 : 한변의 길이
- 5) 호출 예:  
Dotforward2 (150)

3 / 23

### 3. 문제

아래 그림과 같이 길이를 입력 받고, 점선(일점 채선)을 긋는 함수를 만들어보자.

#### 실행결과



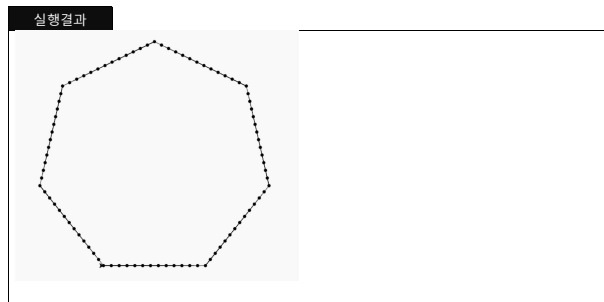
#### 조건

- 1) 길이 30pt 에 점을 하나씩 생성하도록 한다.
- 2) 점은 실선을 짧게 표현한다
- 3) 함수명 : Dotforward3
- 4) 파라미터 : 한변의 길이
- 5) 호출 예:  
Dotforward3 (150)

4 / 23

#### 4. 문제

다음 그림과 같이 다각형 선을 점으로 표현하는 함수 (DotPolygon)를 만드시오.



#### 조건

- 1) 다각형의 한변을 그리는 Dotforward 함수를 만들고, 사용하시오
- 2) 함수명 : DotPolygon
- 3) 파라미터 : 한변의 길이
- 4) 호출 예:  
DotPolygon (7,80)

```
import turtle

aaa = turtle.Turtle()

def MyPolygon(cnt,length):

    for kk in range(cnt):

        aaa.forward(length)

        aaa.left(360/cnt)

def Dotforward(x_length):

    pt_cnt = int(x_length/10) #int
    for jj in range(pt_cnt):

        aaa.dot(5)

        aaa.penup()

        aaa.forward(x_length/pt_cnt) #float
        aaa.pendown()

def DotPoylgon(cnt,length):

    for kk in range(cnt):

        Dotforward(length)

        aaa.left(360/cnt)

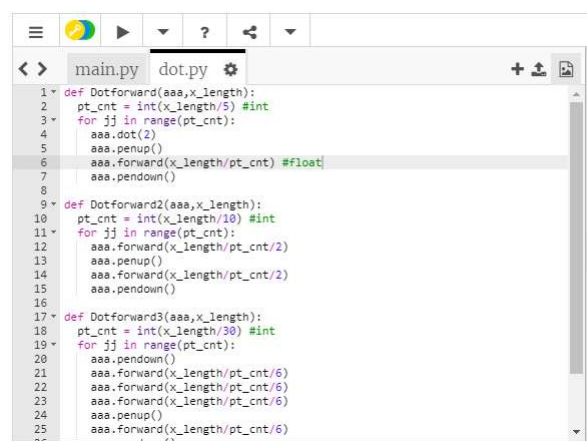
MyPolygon (7,80)

DotPolygon (7,80)
```

### 파일 분리

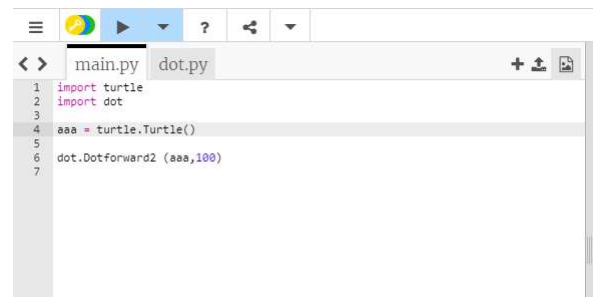
앞서 작업한 여러가지 점선을 그리는 함수를 하나의 파일로 묶어서 관리할 수 있다.

먼저, 아래 그림처럼 함수를 dot.py 라는 이름으로 저장한다.



다음은 dot.py 를 이용하고자 하는 main.py 라는 파일 안에서 dot.py 를 import 문장을 통해 호출 한다.

또한 dot.py 에 있는 함수를 사용할 때는 dot 이라는 패키지명(파일명)을 이용해서 호출 가능하다.



```
Main.py

import turtle

import dot

aaa = turtle.Turtle()

dot.Dotforward2(aaa,100)

dot.py

def Dotforward(aaa,x_length):

    pt_cnt = int(x_length/5) #int
    for jj in range(pt_cnt):

        aaa.dot(2)

        aaa.penup()

        aaa.forward(x_length/pt_cnt) #float
        aaa.pendown()
```

```
def Dotforward2(aaa,x_length):
    pt_cnt = int(x_length/10) #int
    for jj in range(pt_cnt):
        aaa.forward(x_length/pt_cnt/2)
        aaa.penup()
        aaa.forward(x_length/pt_cnt/2)
        aaa.pendown()
```

```
def Dotforward3(aaa,x_length):
    pt_cnt = int(x_length/30) #int
    for jj in range(pt_cnt):
        aaa.pendown()
        aaa.forward(x_length/pt_cnt/6)
        aaa.forward(x_length/pt_cnt/6)
        aaa.forward(x_length/pt_cnt/6)
        aaa.penup()
        aaa.forward(x_length/pt_cnt/6)
        aaa.pendown()
        aaa.forward(x_length/pt_cnt/6)
        aaa.penup()
        aaa.forward(x_length/pt_cnt/6)
```

## 좌표 이용

아래 코드는 두개의 터틀을 이용해서 좌표를 이용해서 이동시키는 경우와 forward 이동 함수를 이용해서 움직이는 두가지 경우를 비교한 것이다.

```
import turtle
aaa = turtle.Turtle()
aaa2 = turtle.Turtle()

aaa.pencolor('red')

aaa.forward(100)
aaa2.goto(100,0)
```



## 5. 문제

터틀을 특정 위치로 이동시키는 함수를 만드시오.

### 조건

- 이동중에 선을 긋지 않도록 한다.
- 함수명 : MyGoto
- 파라미터 : x좌표, y좌표
- 호출 예:  
MyGoto (-100,100)

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

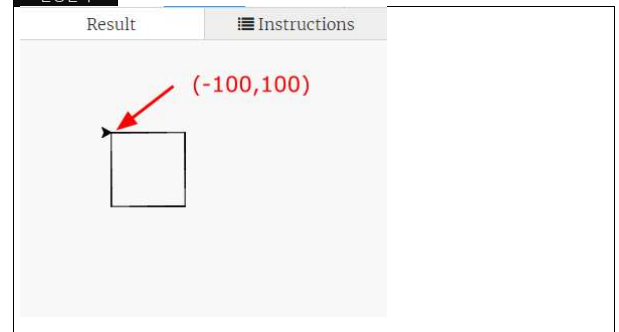
def MyGoto(x_xx,x_yy):
    aaa.penup()
    aaa.goto(x_xx,x_yy)
    aaa.pendown()
```

```
MyGoto(-100,100)
```

## 6. 문제

다음 그림과 같이 직사각형을 그리는 함수 MyRect 를 작성하시오.

### 실행결과



### 조건

- 함수명 : MyRect
- 파라미터 : x좌표, y좌표, 한변의 길이
- 호출 예:  
MyRect(-100,100,80)

```
import turtle
aaa = turtle.Turtle()
aaa.speed(10)

def MyGoto(x_xx,x_yy):
    aaa.penup()
```

```

aaa.goto(x_xx,x_yy)
aaa.pendown()

def MyRect(x_xx,x_yy,x_length):
    MyGoto (x_xx,x_yy)
    aaa.goto(x_xx+x_length,x_yy)
    aaa.goto(x_xx+x_length,x_yy-x_length)
    aaa.goto(x_xx,x_yy-x_length)
    aaa.goto(x_xx,x_yy)

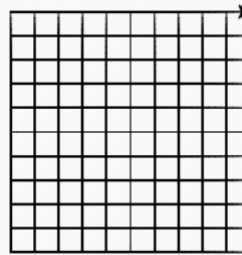
```

```
MyRect(-100,100,80)
```

## 7. 문제

다음 그림과 같이 바둑판 모양을 그리시오..

### 실행결과



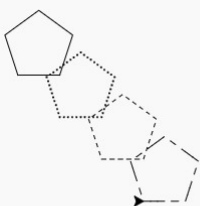
### 조건

- 반복문을 사용할 것

## ◆ HW ★

다음 그림과 같이 다각형의 선 모양을 4가지 종류로 선택 할 수 있는 함수 (DotPolygon) 을 만드시오.

### 실행결과



### 조건

- 1) 그림의 중복을 막기 위해 MyGoto 함수를 이용해서 좌표이동을 하시오.
- 2) 함수명 : DotPolygon
- 3) 파라메터 : n각형, 한변의길이, 선의 모양 (default 값)
- 4) 호출 예:  
DotPolygon(5,50,'-')

### Main.py

```

import turtle
import dot

aaa = turtle.Turtle()

```

```
aaa.speed(100)
```

```
def MyGoto(x_xx,x_yy):
```

```
    aaa.penup()
```

```
    aaa.goto(x_xx,x_yy)
```

```
    aaa.pendown()
```

```
def MyPolygon(cnt,length):
```

```
    for kk in range(cnt):
```

```
        aaa.forward(length)
```

```
        aaa.left(360/cnt)
```

```
def DotPolygon(cnt,length,x_op=''):

```

```
    for kk in range(cnt):
```

```
        if x_op == '':
```

```
            aaa.forward(length)
```

```
            aaa.left(360/cnt)
```

```
        elif x_op == '·':
```

```
            dot.Dotforward(aaa,length)
```

```
            aaa.left(360/cnt)
```

```
        elif x_op == '·-':
```

```
            dot.Dotforward2(aaa,length)
```

```
            aaa.left(360/cnt)
```

```
        elif x_op == '·-·':
```

```
dot.Dotforward3(aaa,length)
aaa.left(360/cnt)
```

```
MyGoto(-150,100)
DotPoylgon(5,50)
MyGoto(-100,50)
DotPoylgon(5,50,'.')
MyGoto(-50,0)
DotPoylgon(5,50,'-')
MyGoto(0,-50)
DotPoylgon(5,50,'.-')
```

dot.py

```
def Dotforward(aaa,x_length):
    pt_cnt = int(x_length/5) #int
    for jj in range(pt_cnt):
        aaa.dot(2)
        aaa.penup()
        aaa.forward(x_length/pt_cnt) #float
        aaa.pendown()

def Dotforward2(aaa,x_length):
    pt_cnt = int(x_length/10) #int
    for jj in range(pt_cnt):
```

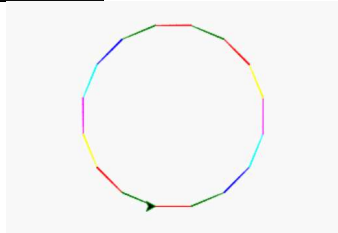
```
aaa.forward(x_length/pt_cnt/2)
aaa.penup()
aaa.forward(x_length/pt_cnt/2)
aaa.pendown()
```

```
def Dotforward3(aaa,x_length):
    pt_cnt = int(x_length/30) #int
    for jj in range(pt_cnt):
        aaa.pendown()
        aaa.forward(x_length/pt_cnt/6)
        aaa.forward(x_length/pt_cnt/6)
        aaa.forward(x_length/pt_cnt/6)
        aaa.penup()
        aaa.forward(x_length/pt_cnt/6)
        aaa.pendown()
        aaa.forward(x_length/pt_cnt/6)
        aaa.penup()
        aaa.forward(x_length/pt_cnt/6)
```

## ◆ HW ★

다음 그림과 같이 N각형의 마주보는 변의 색을 같은 색으로 표현하는 함수를 만들어보자

실행결과



조건

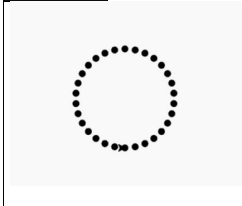
- 1) 색정보를 리스트를 담아서 사용할 것
- 2) 인접한 색이 연속해서 출력되지 않도록 할 것
- 3) 마주보는 면이 없는 경우는 흑백으로 출력 하도록 할 것
- 4) 함수명 : ColorPolygon
- 5) 파라미터 : n각형, 한변의길이
- 6) 호출 예:
 

```
ColorPolygon(16,30)
```

## 8. 문제

다음 그림과 같이 원을 점선으로 그리는 함수를 작성해보자.

실행결과



조건

- 1) 방법1) circle() 함수를 이용
- 2) 방법2) 원주율을 이용
- 3) 함수명 : DotCircle
- 4) 파라미터 : 점 갯수, 반지름
- 5) 호출 예:
 

```
DotCircle(30,100)
```

Circle 이용

```
import turtle

aaa=turtle.Turtle()

def DotCircle(x_dot_cnt,x_radius):
    for kk in range(x_dot_cnt):
        aaa.dot(10)
```

```

aaa.penup()
aaa.circle(100,360/x_dot_cnt)
aaa.pendown()

```

```

DotCircle(30,100)

```

## 원주율 이용

```

import turtle
import math
aaa=turtle.Turtle()
def DotCircle(x_dot_cnt,x_radius):
    length = 2*math.pi*x_radius
    for kk in range(x_dot_cnt):
        aaa.dot(10)
        aaa.penup()
        aaa.forward(length/x_dot_cnt)
        aaa.left(360/x_dot_cnt)
        aaa.pendown()
DotCircle(30,100)

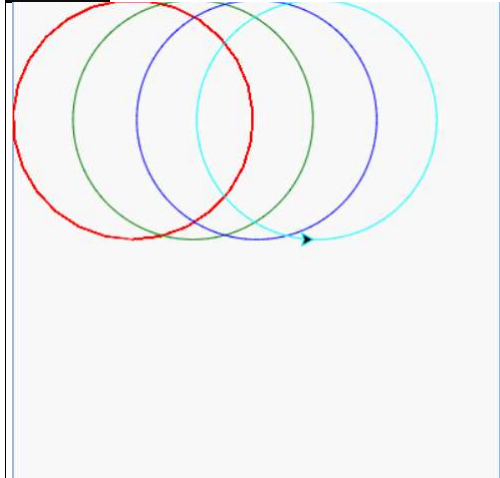
```

## ◆ HW ★★

아래 그림은 반지름 100이 주어졌을 때 원을 그리는 함수 4가지의 비교이다.

빨강색 원은 반복문을 사용하지 않고 circle() 함수의 입력값을 360 으로 하는 일반적인 형태이다. 나머지 3가지 원은 반복문을 이용하는 3가지 방법을 함수 MyCircle1, MyCircle2, MyCircle3 이름으로 각각 만들어 사용했다. 반복문을 이용한 3가지 방법의 함수를 모두 구현하시오.

## 실행결과



## 조건

- 1) 방법1) circle 함수 이용
- 2) 방법2) 원주율 이용
- 3) 방법2) 삼각함수 이용
- 4) 함수명 : MyCircle

- 5) 파라미터 : 반지름, 색
- 6) 호출 예:
 

```

MyCircle1 (100, "green")
MyCircle2 (100, "blue")
MyCircle3 (100, "cyan")

```

```

MyGoto(-100,0),
aaa.pencolor('red')
aaa.circle(100,360)
MyGoto(-50,0)
MyCircle1(100,'green')
MyGoto(0,0)
MyCircle2(100,'blue')
MyGoto(50,0)
MyCircle3(100,'cyan')

```