SARAVAJANIK COLLEGE OF ENGINEERING & TECHNOLOGY

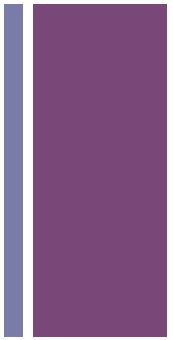
Turtle

Hello, Turtle Graphics!

Turtle Graphics

- Turtle graphics is a method of programming “vector” graphics using a relative cursor upon a Cartesian plane
- Python has a built in module that supports turtle graphics called the “turtle” module. Importing this module gives you access to all the turtle graphics functions you will need to draw vector graphics on the screen.

Turtle Graphics

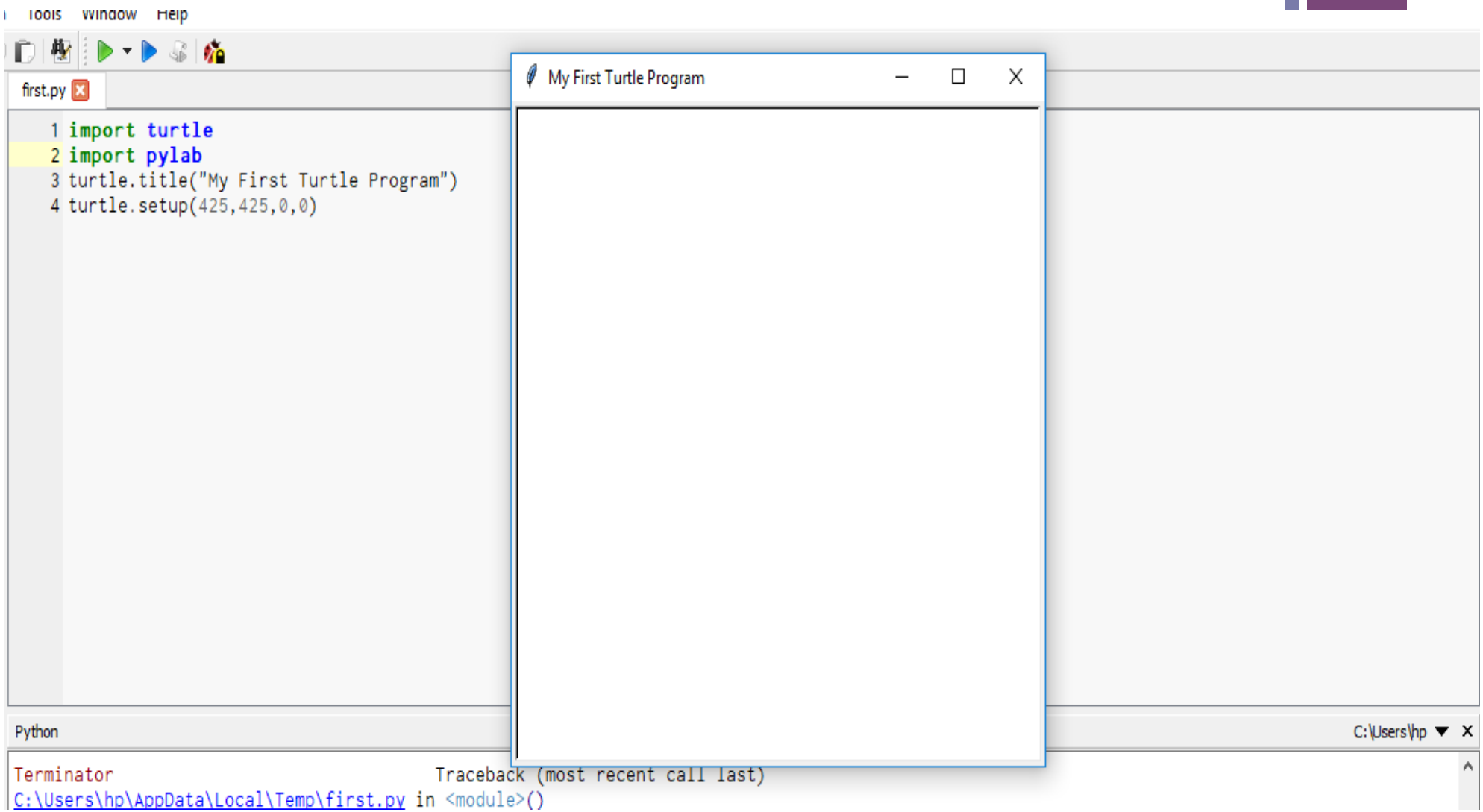


- In turtle graphics you control a cursor, also know as a "turtle". It has the following properties
  - A position in 2D space
  - An orientation, or heading
  - A pen that can lay down color on the canvas

Setting up your turtle environment

```
import turtle


# set a title for your canvas window

turtle.title(”My Awesome Turtle Animation")


# set up the screen size (in pixels - 425 x 425)

# set the starting point of the turtle (0, 0)

turtle.setup(425, 425, 0, 0)
```

Setting up your turtle environment

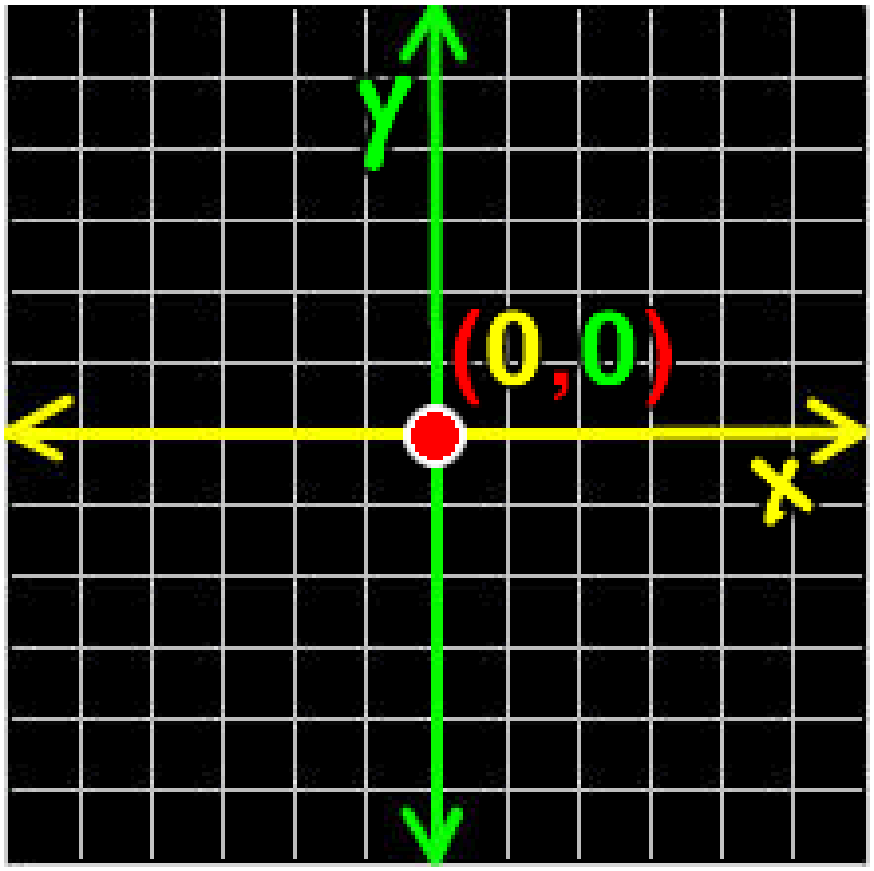


Getting rid of the turtle window

- This function call will cause your turtle window to deactivate when you click on it. Place it at the end of your program.

  turtle.exitonclick()

The Turtle Canvas



Basic Drawing

- You can move your turtle forward by using the following command:

  turtle.forward(pixels)

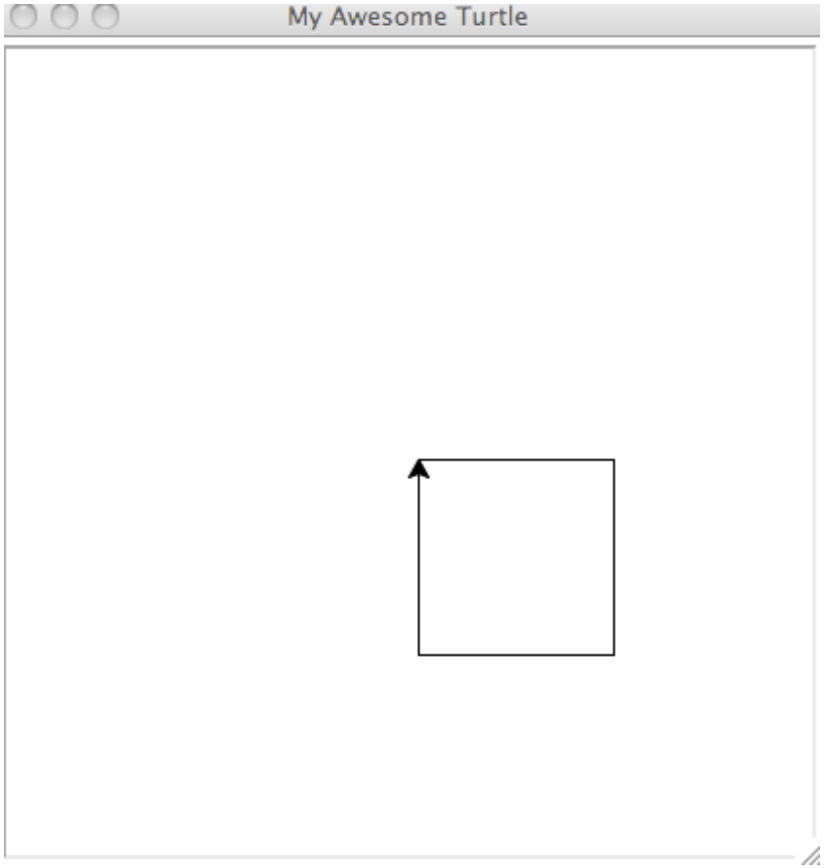
- And you can have your turtle turn by using the following commands:

  turtle.right(degres)
  turtle.left(degrees)

- Your turtle will continually "paint" while it's moving.

Programming Challenge: Draw a Box!

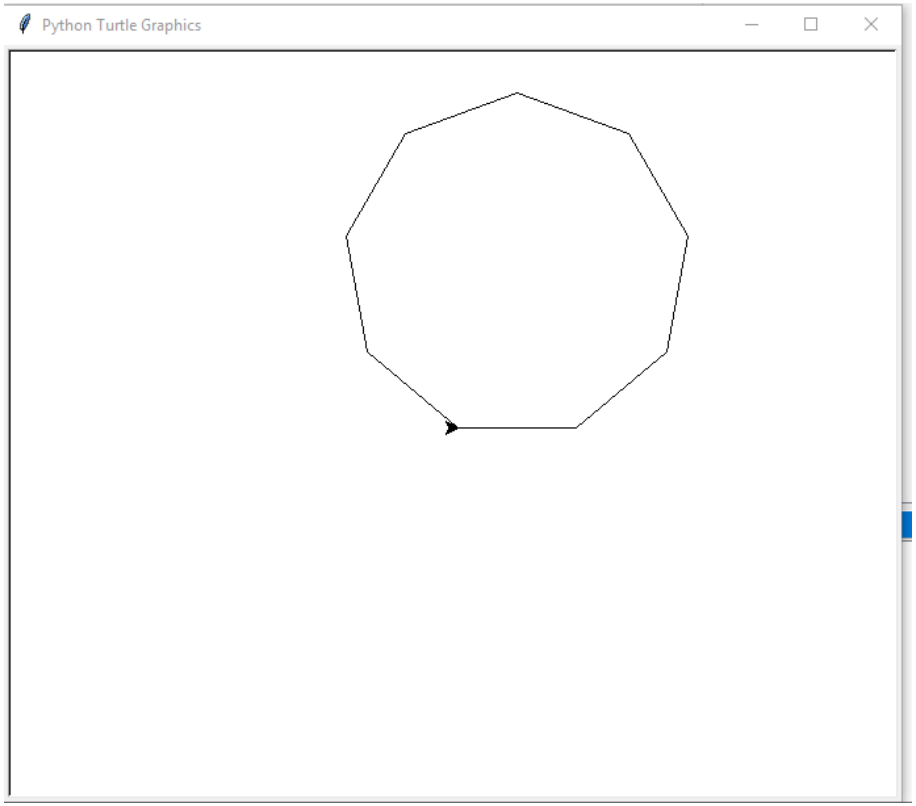
```
import turtle
turtle.setup(420,420,0,0)
turtle.forward(90)
turtle.left(90)
turtle.forward(90)
turtle.left(90)
turtle.forward(90)
turtle.left(90)
turtle.forward(90)
```



Programming Challenge: Polygon Function

- Write a function that allows you to draw a custom polygon. Your function should take two arguments – the number of sides you wish to draw and the length of a given side.

```
import turtle as t
def custompoly(side,length):
t.angle = 360.0 / side


for i in range(side):
t.forward(length)
t.left(t.angle)
r1 = int(input("Enter number of
side:"))
r2=int(input("Enter number of
length:"))
custompoly(r1,r2)
```



Moving your turtle

- You can cause your turtle to move to any coordinate on the screen by using the turtle.goto() function along with the x,y coordinate you wish to move to.
  Example:

  turtle.goto(50,50)

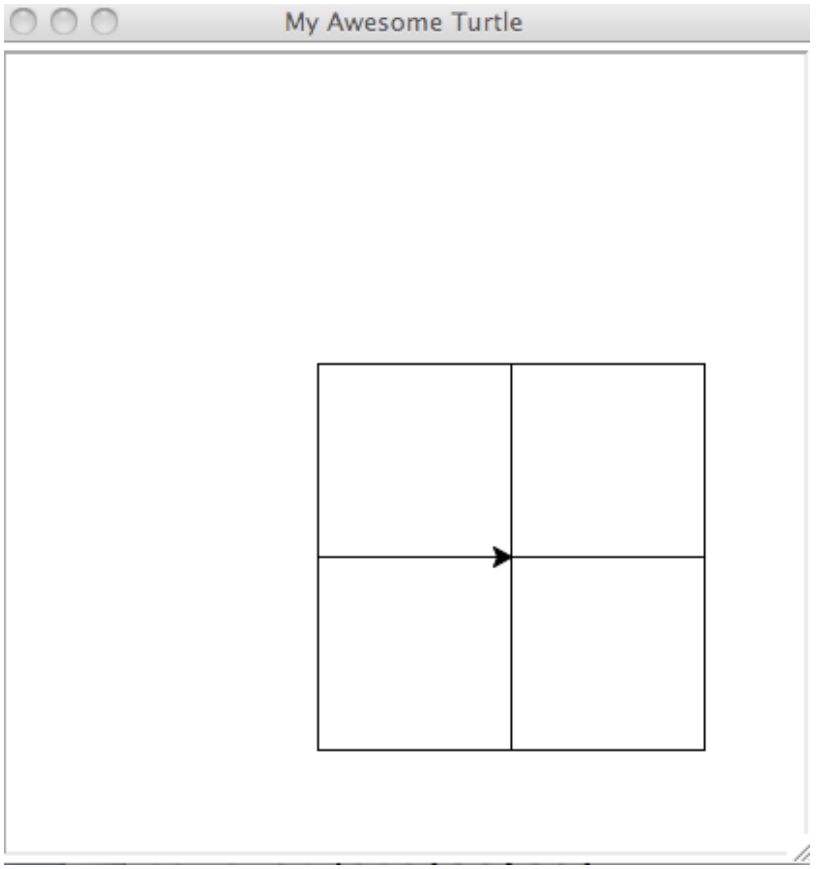
- Note that your pen will continue to draw as you move your turtle to a new position.

- You can tell the turtle to stop drawing by using the turtle.penup() function. You can tell it to start drawing again by using the turtle.pendown() function.

Programming Challenge: Four Squares

```
import turtle
turtle.goto(90,0)
turtle.penup()
turtle.pendown()
turtle.goto(90,90)
turtle.penup()
turtle.pendown()
turtle.goto(0,90)
turtle.penup()
turtle.pendown()
turtle.goto(0,0)
turtle.penup()
turtle.pendown()
turtle.goto(0,-90)
turtle.penup()
turtle.pendown()
turtle.goto(90,-90)
turtle.penup()
turtle.pendown()
turtle.goto(90,0)
turtle.penup()
turtle.pendown()
```

```
turtle.goto(0,0)
turtle.penup()
turtle.pendown()
turtle.goto(0,-90)
turtle.penup()
turtle.pendown()
turtle.goto(-90,-90)
turtle.penup()
turtle.pendown()
turtle.goto(-90,0)
turtle.penup()
turtle.pendown()
turtle.goto(-90,90)
turtle.penup()
turtle.pendown()
turtle.goto(0,90)
turtle.penup()
turtle.pendown()
turtle.goto(0,0)
turtle.penup()
turtle.pendown()
turtle.goto(-90,0)
turtle.penup()
```



Programming Challenge: Random Circles

```
from turtle import *
import random as r
for i in range(30):
penup()
goto(r.randint(-200,200), r.randint(-200,200))
pendown()
red_amount = r.randint( 0, 30) / 100.0
blue_amount = r.randint(50, 100) / 100.0
green_amount = r.randint( 0, 30) / 100.0
pencolor((red_amount,blue_amount,green_amount))
circle_size = r.randint(10, 40)
pensize(r.randint(1, 5))
for i in range(6):
circle(circle_size)
left(60)
```

Pen Color

- By default your turtle will draw solid black lines on the canvas
- You can change this behavior by calling the turtle.pencolor() function
- This function takes one argument – a color value.
- The color value can be a standard Red Green Blue (RGB) color code. Here's an example:

```
# sets the pen color to red
turtle.pencolor("#FF0000")

# sets the pen color to blue
turtle.pencolor(#0000FF")
```

RGB Color

- RGB color codes define how much red, green and blue should be mixed into a given color
- They are divided into three channels – one for red, one for green and one for blue, in that order
- They usually start with the "#" sign
- The first two characters in a color string signify how much red goes into the color
- The second two signify how much green
- And the third set signifies how much blue

RGB Color

- RGB uses a number system called "hexadecimal"
- Hexadecimal (or "hex") is a base 16 number system. This means it uses 16 "digits" to represent numbers
- Counting in hex:
  - 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F,10
  - 11,12,13,14,15,16,17,18,19,1A,1B,1C,1D,1E,1F,20

RGB Color

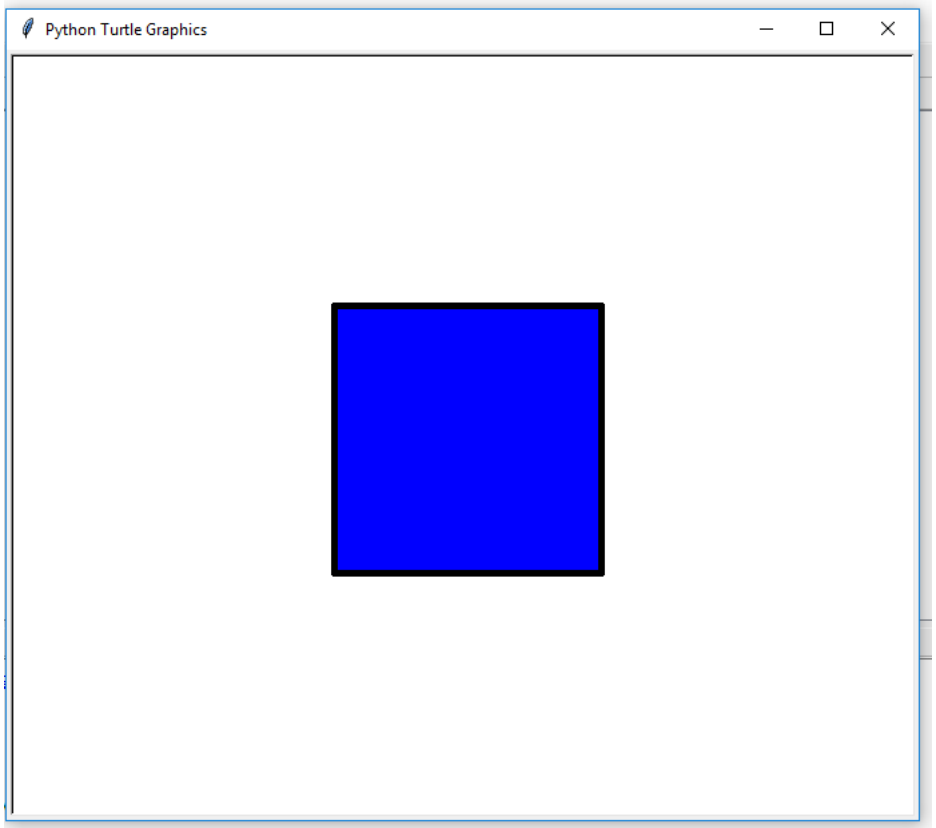
- `#000000` # no red, green or blue – all black
- `#FFFFFF` # full red, green and blue – all white
- `#FF0000` # all red
- `#00FF00` # all green
- `#0000FF` # all blue

Creating filled shapes

- If you are drawing closed polygons you can tell Python to fill them with an RGB color by calling the turtle.fillcolor() function. This function takes one argument – a color value.
- Next, you need to call the turtle.begin_fill() function to tell Python to start filling in your shape. You would call this function just before you start drawing your shape.
- At the end of your shape you can use the turtle.end_fill() to tell Python to stop filling in your shape.

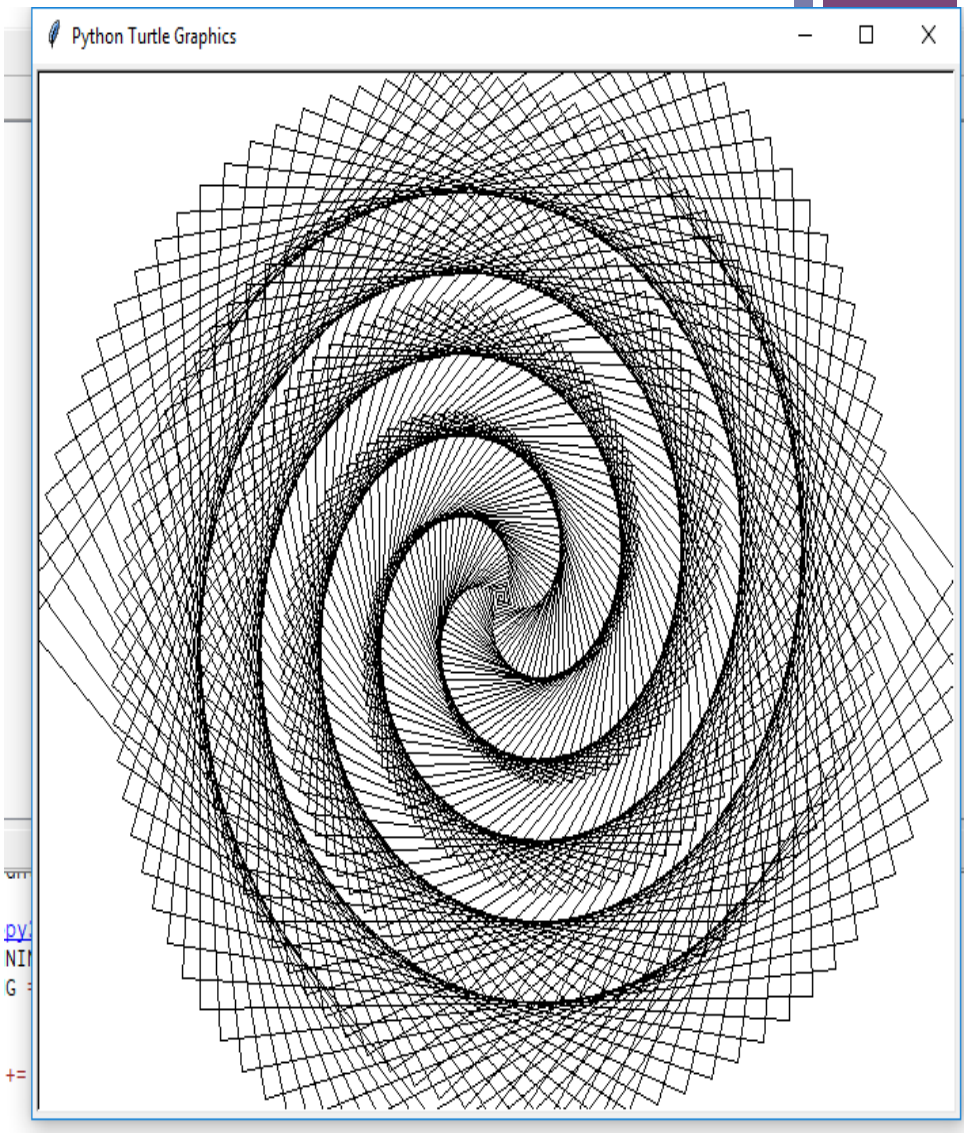
Programming Challenge: Filled Squares

```
import turtle as t


t.color('black','blue')
t.begin_fill()
t.shapesize(10,10,5)
t.shape("square")
t.shape()
t.end_fill()
```



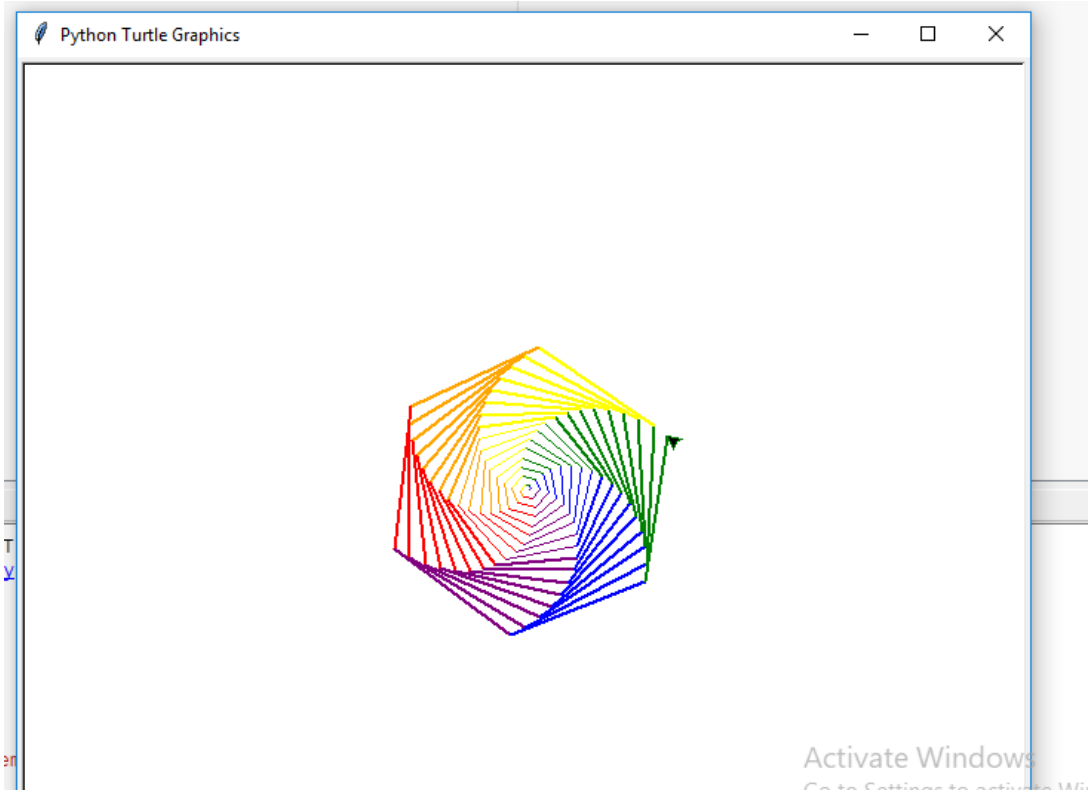
Programming Challenge: Spiral

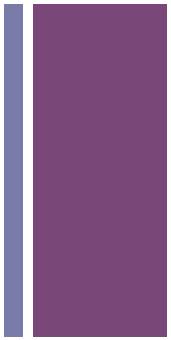
```
import turtle
for i in range(500):
turtle.forward(i)
turtle.left(91)
```



Programming Challenge: Color Spiral

```
import turtle
colors = ['red', 'purple', 'blue', 'green',
'yellow', 'orange']

for x in range(100):
turtle.pencolor(colors[x % 6])
turtle.width(x / 100 + 1)
turtle.forward(x)
turtle.left(59)
```





THANK YOU