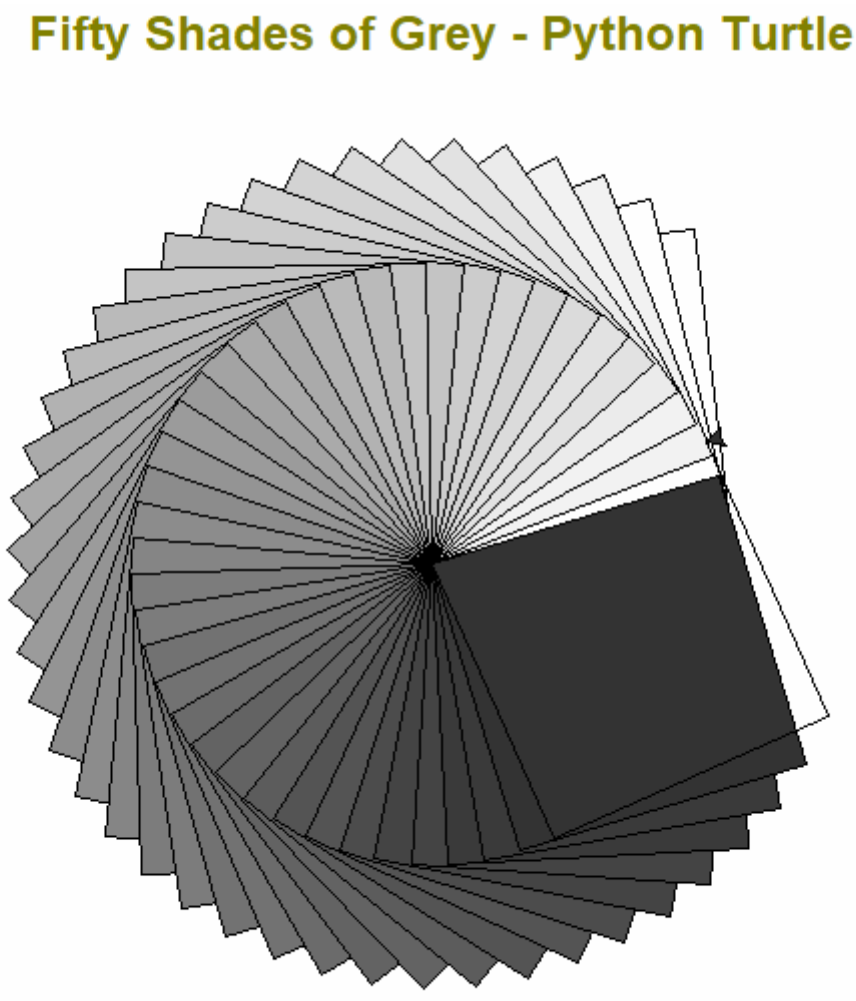
Web Development

Python Turtle Tutorial and Animations

Python Turtle is something that evolved from Logo programming language, invented in 1966 by Wally Feurzig. With the aid of **Object Oriented Programming** approach, we can create an impressive set of animations easily.

The following animation was created by Python Turtle; the code is at the bottom of this tutorial.



The Requirements

Before using Python Turtle for animations, please take the following steps to install the environment:

- Download the latest version of Python from here.
- Load Python IDLE - Integrated Development and Learning Environment - from Windows.
- Open a new file and save it with **.py** extension.
- Write down **import turtle** at the top of the file in order to import the module - classes and methods.
- Write down the code and make impressive animations.

The best way to learn the Python turtle is running set of codes, from the simplest to the more advanced gradually, rather than making an effort to understand the simulator fully at first. This is the approach adopted in this tutorial.

The documentation of Python turtle is here.

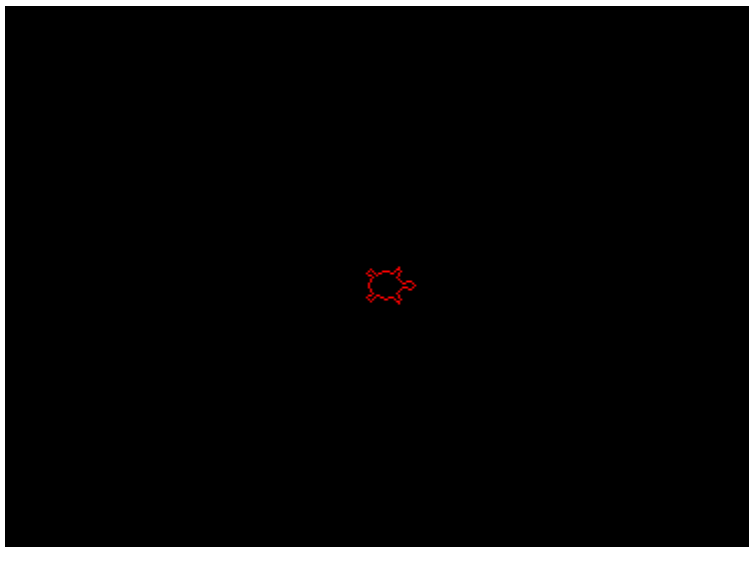
> Here are some of the turtle methods; they direct the turtle what to do:

Instructions	Meaning
turtle.forward(number)	Move forward
turtle.back(number)	Move backward

turtle.back(number)	Move backward
turtle.right(angle)	Turn clockwise
turtle.left(angle)	Turn anti-clockwise
turtle.pencolor(colour string)	Drawing colour
turtle.pensize(number)	Choosing size of pen nib
turtle.circle(radius)	Drawing a circle
turtle.speed(number)	Choosing speed - 1 to 10
turtle.write(message,font)	Writing on the screen
turtle.ht()	Hiding the turtle
turtle.setpos(x,y)	Changing the position of turtle

Drawing a right-angle

The following animation shows the turtle at work in producing a right-angle on the screen:



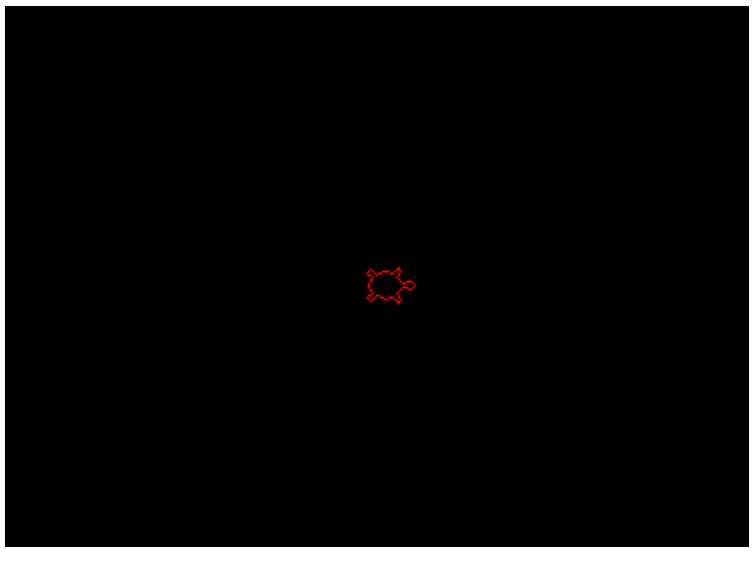
This is the code for the above animation:

```
import turtle   # importing the module
trtl = turtle.Turtle()   #making a turtle object of Turtle class for drawing
screen=turtle.Screen()   #making a canvas for drawing
screen.setup(400,300)   #choosing the screen size
screen.bgcolor('black')   #making canvas black
trtl.pencolor('red')   #making colour of the pen red
trtl.pensize(5)   #choosing the size of pen nib
trtl.speed(1)   #choosing the speed of drawing
trtl.shape('turtle')  #choosing the shape of pen nib
trtl.forward(150)   #drawing a line of 200 pixels
trtl.right(90)   #asking turtle to turn 90 degrees
trtl.forward(150)   #drawing a line of 200 pixels
trtl.penup()   # preparing for moving pen without drawing
trtl.setpos(-140,-120)   # making the new position of the turtle
trtl.pendown()  # bringing the pen down for drawing again

trtl.pencolor('green')   # choosin the pen colour as green
trtl.write('Vivax Solutions', font=("Arial", 20, "bold"))   # chosing the font
trtl.penup()
trtl.ht()   # hiding the turtle from the screen
```

sign indicates the comments in Python scripts.

Drawing a Square

The following animation shows how the turtle draws a square on the screen:



This is the code for the above animation:

```
import turtle # importing the module
trtl = turtle.Turtle()   #making a turtle object of Turtle class for drawing
screen=turtle.Screen()   #making a canvas for drawing
screen.setup(400,300)   #choosing the screen size
screen.bgcolor('black')   #making canvas black
trtl.pencolor('red')   #making colour of the pen red
trtl.pensize(5)   #choosing the size of pen nib
trtl.speed(1)   #choosing the speed of drawing
trtl.shape('turtle')  #choosing the shape of pen nib
trtl.forward(100)   #top line
trtl.right(90)
trtl.forward(100)   # right vertical line
trtl.right(90)
trtl.forward(100)  # bottom line
trtl.right(90)
trtl.forward(100)  # left vertical line
# information printing
trtl.penup()
trtl.setpos(-120,100)
trtl.pendown()
trtl.pencolor('green')
trtl.write('Square - Vivax Solutions', font=("Arial", 16, "bold"))
trtl.penup()
trtl.ht()
```

Although, the above code produces a square, it is not good programming practice due to repetition of the code, which could have been tackled by a simple loop. Therefore, the code can be revised to produce the same shape with efficiency as follows:

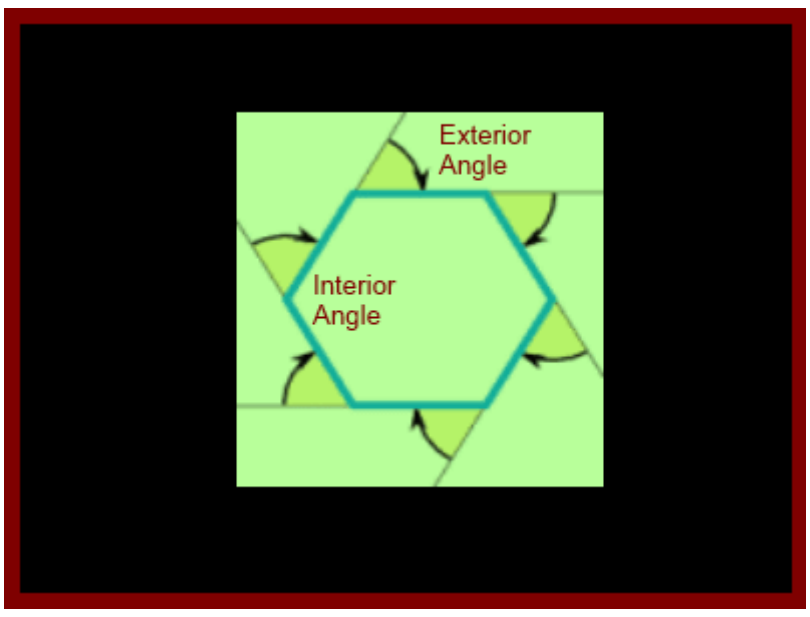
```
import turtle   # importing the module
trtl = turtle.Turtle()   #making a turtle object of Turtle class for drawing
screen=turtle.Screen()   #making a canvas for drawing
screen.setup(400,300)   #choosing the screen size
screen.bgcolor('black')   #making canvas black
trtl.pencolor('red')   #making colour of the pen red
trtl.pensize(5)   #choosing the size of pen nib
trtl.speed(1)   #choosing the speed of drawing

trtl.shape('turtle')  #choosing the shape of pen nib
for i in range(4): # for loop to minimize the same lines of codes being written
    trtl.forward(100)  # for lines
    trtl.right(90)   # for turning
# information printing
```

```
trtl.penup()
trtl.setpos(-120,100)
trtl.pendown()
trtl.pencolor('green')
trtl.write('Square - Vivax Solutions', font=("Arial", 16, "bold"))
trtl.penup()
trtl.ht()
```

The **for loop** makes the drawing of the square much easier. It role is really important when we create other polygons.

Drawing Polygons

In order to change the above code to draw polygons, we need to take into account some concepts in geometry. As you can see, with each turn, the turtle moves through a certain angle, specified by **right(angle).** This is the **exterior angle** of the polygon, which is the same as **360/n**, where **n** is the number of sides.



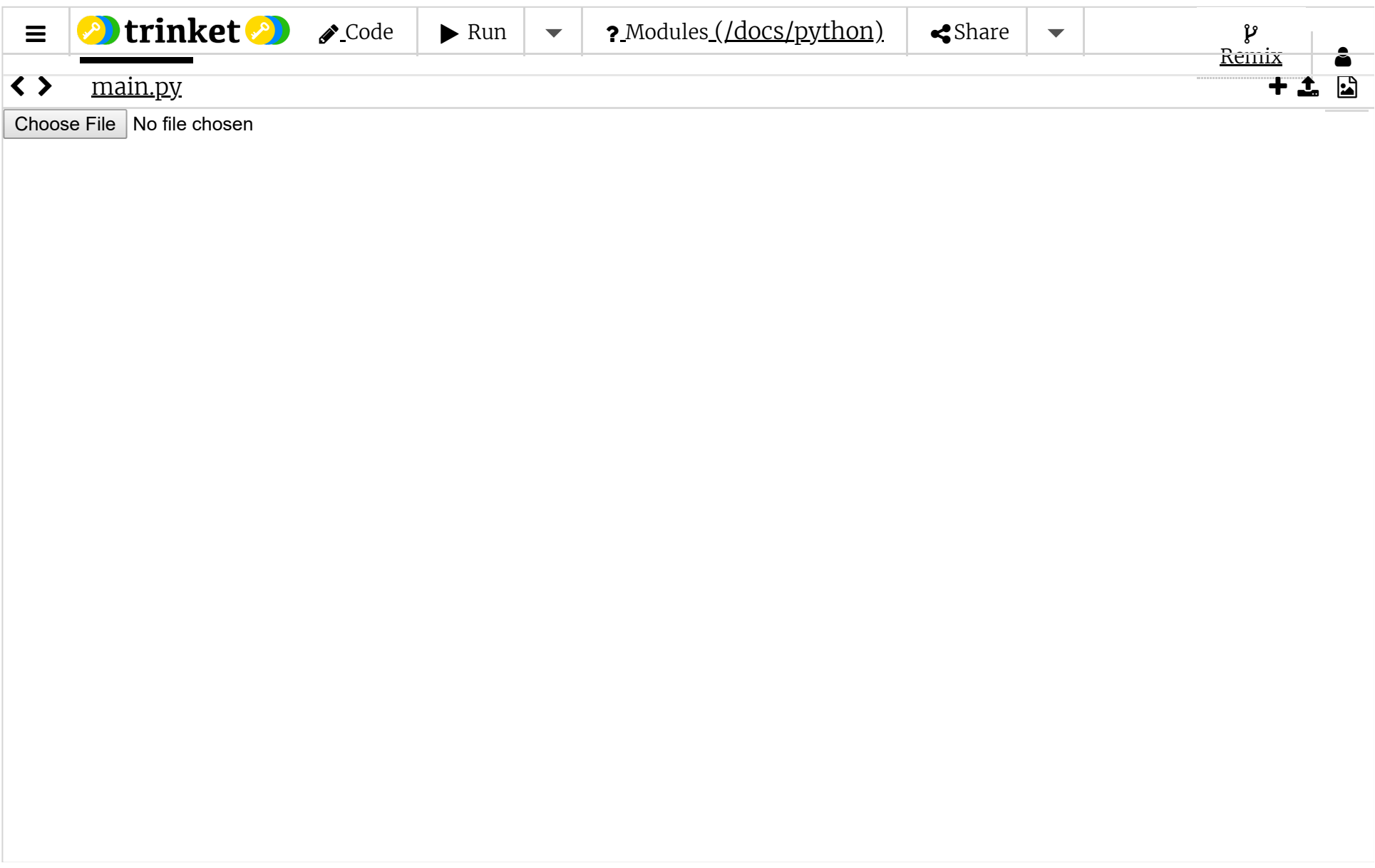
With **for** and **while** loops, the following code produces a set of polygons on screen - from a triangle to a decagon.

The code for the above animation is as follows:

```
import turtle    # importing the module
import time    #importing the time module trtl = turtle.Turtle()    #making a turtle object of Turtle class for drawing
screen=turtle.Screen()    #making a canvas for drawing
screen.setup(620,470)    #choosing the screen size
screen.bgpic('bg.gif')    #making canvas black
trtl.pencolor('red')    #making colour of the pen red
trtl.pensize(5)    #choosing the size of pen nib
trtl.speed(1)    #choosing the speed of drawing
trtl.shape('turtle')   #choosing the shape of pen nib
time.sleep(12)
n=3   starting for a triangle
shapes=['Triangle','Square','Pentagon','Hexagon','Heptagon','Octagon','Nonagon','Decagon']
while n<11: #   limiting to a decagon
     for i in range(n):    # for loop to minimize the same lines of codes being written
        trtl.pencolor('red')
        trtl.forward(100)    #top line
        trtl.right(360/n)   #determining the exterior angle of the polygon
trtl.penup()
trtl.setpos(-80,180)   #moving the turtle to make the animation more centric
trtl.pendown()
trtl.pencolor('blue')
trtl.write(' This is '+shapes[n-3], font=("Arial", 16, "bold"))   #printing the name of the polygon
n=n+1
time.sleep(1)
  #making turtle sleep for one second trtl.clear()
trtl.penup()
trtl.setpos(-n*8,n*14)
trtl.pendown()
```

If you want to practise it interactively, here is the code:

Drawing Letter E



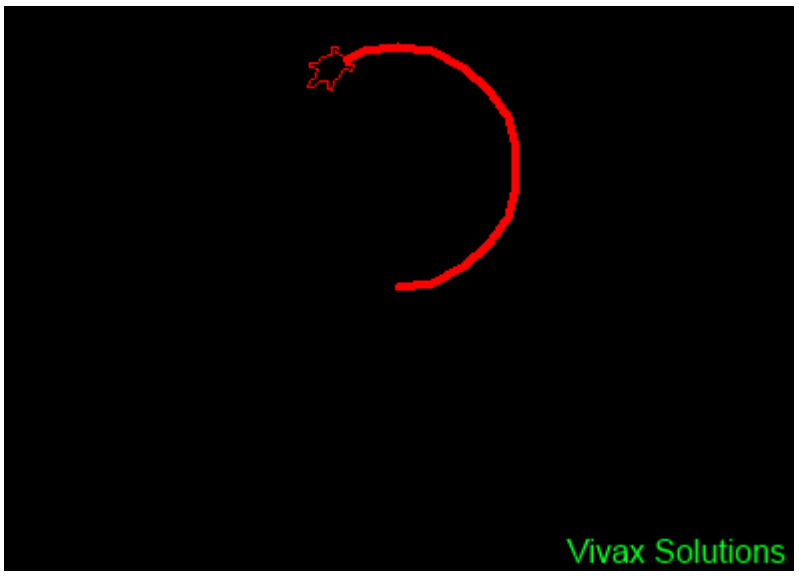
Please note how the image is centred on the screen, with turtle.setpos() method.

Drawing Circles

Here is the code for adding two numbers and drawing a circle:

```
import turtle    # importing the module
trtl = turtle.Turtle()    #making a turtle object of Turtle class for drawing
screen=turtle.Screen()    #making a canvas for drawing
screen.setup(420,320)    #choosing the screen size
screen.bgpic('bg.gif')    #making canvas black
trtl.pencolor('red')    #making colour of the pen red
trtl.pensize(4)    #choosing the size of pen nib
trtl.speed(1)    #choosing the speed of drawing


trtl.shape('turtle')  #choosing the shape of pen nib
trtl.circle(60)  drawing circle with radius 60 pixels
```

The following animation shows how the two numbers are taken in as two inputs and later answer is given out:



As you can see, the starting point of the circle is the centre of the screen by default, which is not the centre of the circle. In order to get round this problem, we have to set the position by code as follows:

```
import turtle    # importing the module
trtl = turtle.Turtle()    #making a turtle object of Turtle class for drawing
screen=turtle.Screen()    #making a canvas for drawing
screen.setup(420,320)    #choosing the screen size
screen.bgpic('bg.gif')    #making canvas black
trtl.pencolor('red')    #making colour of the pen red
trtl.pensize(4)    #choosing the size of pen nib
trtl.shape('turtle')   #choosing the shape of pen nib
trtl.penup()   #moving the pen up
trtl.setpos(0,-60)   #setting new position
trtl.pendown()   #moving the pen down
trtl.circle(60)   #drawing circle with radius 60 pixels
```

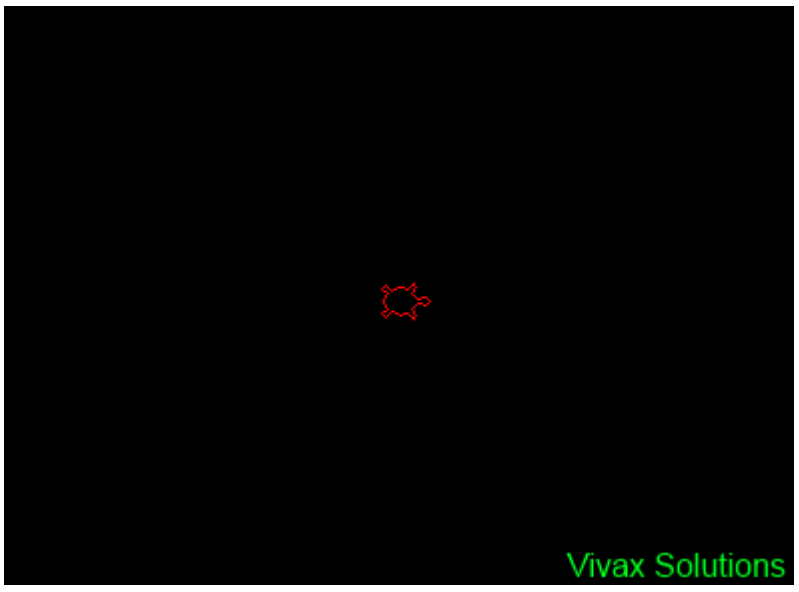
Drawing Concentric Circles

Here is the code for drawing concentric circles with the aid of loops:

```
import turtle    # importing the module
trtl = turtle.Turtle()    #making a turtle object of Turtle class for drawing
screen=turtle.Screen()    #making a canvas for drawing
screen.setup(420,320)    #choosing the screen size
screen.bgpic('bg.gif')    #making canvas black
trtl.pencolor('red')    #making colour of the pen red
trtl.pensize(4)    #choosing the size of pen nib
trtl.shape('turtle')   #choosing the shape of pen nib
n=0

while n<7:   #loop for 7 circles
n=n+1
trtl.penup()
trtl.setpos(0,-n*20)
trtl.pendown()
```

```
trtl.circle(20*n)
```

The following animation shows how the animation works:



Changing Colours

The colour of the pen can be changed in many different ways; here are two ways:

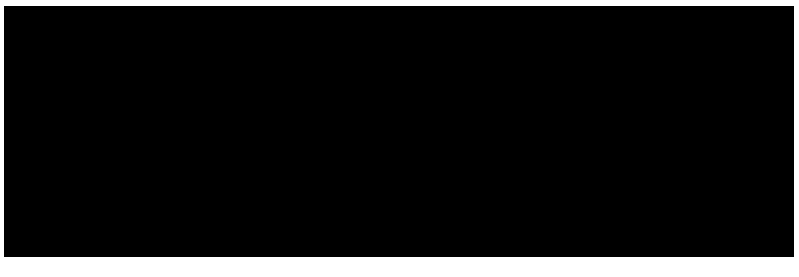
1. turtle.pencolor('red')
2. turtle.pencolor(red, green, blue)

If you use the second method, red, green and blue can be any integer between **1 - 255.** However, before that, **turtle.colormode(1) or turtle.colormode(255)** must be declared in the code.

Here is the code:

```
import turtle    # importing the module
import random    # importing random module for generating random numbers
trtl = turtle.Turtle()    #making a turtle object of Turtle class for drawing
screen=turtle.Screen()    #making a canvas for drawing
screen.setup(420,320)    #choosing the screen size
screen.bgpic('bg.gif')    #making canvas black
trtl.pensize(4)    #choosing the size of pen nib
trtl.speed(1)    #choosing the speed of drawing
trtl.shape('turtle')   #choosing the shape of pen nib
n=0
while n<7:
r=random.randint(1,120)   #random numbers for red
g=random.randint(81,200)   #random numbers for green
b=random.randint(61,255)   #random numbers for blue
turtle.colormode(255)   #declaring the colour mode
trtl.pencolor(r,g,b)   #pen colour
n=n+1
trtl.penup()
trtl.setpos(0,-n*20)
trtl.pendown()
trtl.circle(20*n)
```

The following animation shows the output - with random colours, of course.



Advanced Animations - turtle in its habitat!

The following code creates a turtle that moves around on a beach - leaving behind a certain regular pattern!

```
import turtle
import random
trtl = turtle.Turtle()
screen=turtle.Screen()
screen.setup(420,320)
screen.bgpic('bg.gif')
trtl.pensize(4)
trtl.speed(1)
trtl.shape('turtle')
turtle.colormode(255)
trtl.pencolor(242,242,242)
trtl.penup()
trtl.setpos(-160,-100)
trtl.pendown()
for i in range(4):
    trtl.forward(40)
    trtl.left(90)
    trtl.forward(30)
    trtl.right(90)
    trtl.forward(40)
```

The following animation shows the iteration that leads to a countdown, based on the user input:

Advanced Animations - diverging turtles

In this animation, turtles leave their foot print on the beach: this is achieved by **turtle.stamp()** method along with **turtle.penup()**. This is the code:

```
import turtle
import random
import time
screen=turtle.Screen()
trtl=turtle.Turtle()
```

```
screen.setup(420,320)
screen.bgpic('bg.gif')
trtl.shape('turtle')
trtl.color('darkgoldenrod','black')
s=10
trtl.penup()
trtl.setpos(30,30)
for i in range(28):
    s=s+2
    trtl.stamp()
    trtl.forward(s)
    trtl.right(25)
    time.sleep(0.25)     #activated with a break of a 1/4th of a second
```

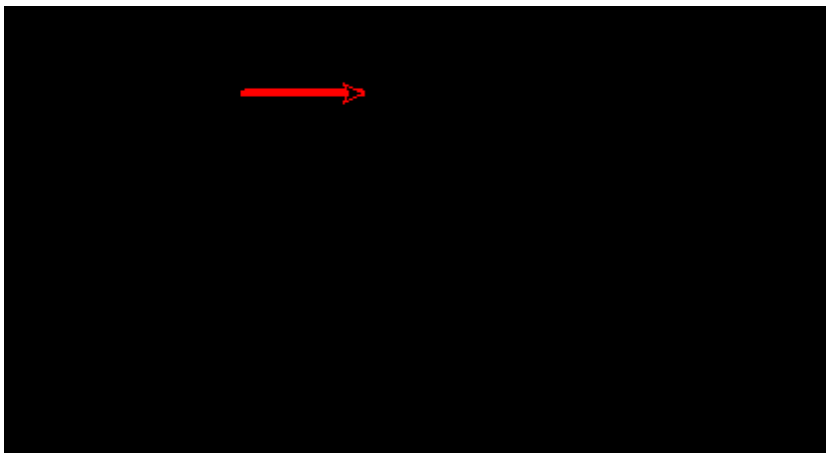
The animation is as follows:

Advanced Animations - colourful star

With the aid of simple geometry, a star can be drawn by Python Turtle. This is the code:

```
import turtle
import time
screen=turtle.Screen()
trtl=turtle.Turtle()
screen.setup(420,320)
screen.bgcolor('black')
clr=['red','green','blue','yellow','purple']
trtl.pensize(4)
trtl.penup()
trtl.setpos(-90,30)
trtl.pendown()
for i in range(5):
   trtl.pencolor(clr[i])
   trtl.forward(200)
   trtl.right(144)
trtl.penup()
trtl.setpos(80,-140)
trtl.pendown()
trtl.pencolor('olive')
trtl.write('Vivax Solutions',font=("Arial", 12, "normal"))
trtl.ht()
```

This is the animation that produces the star:

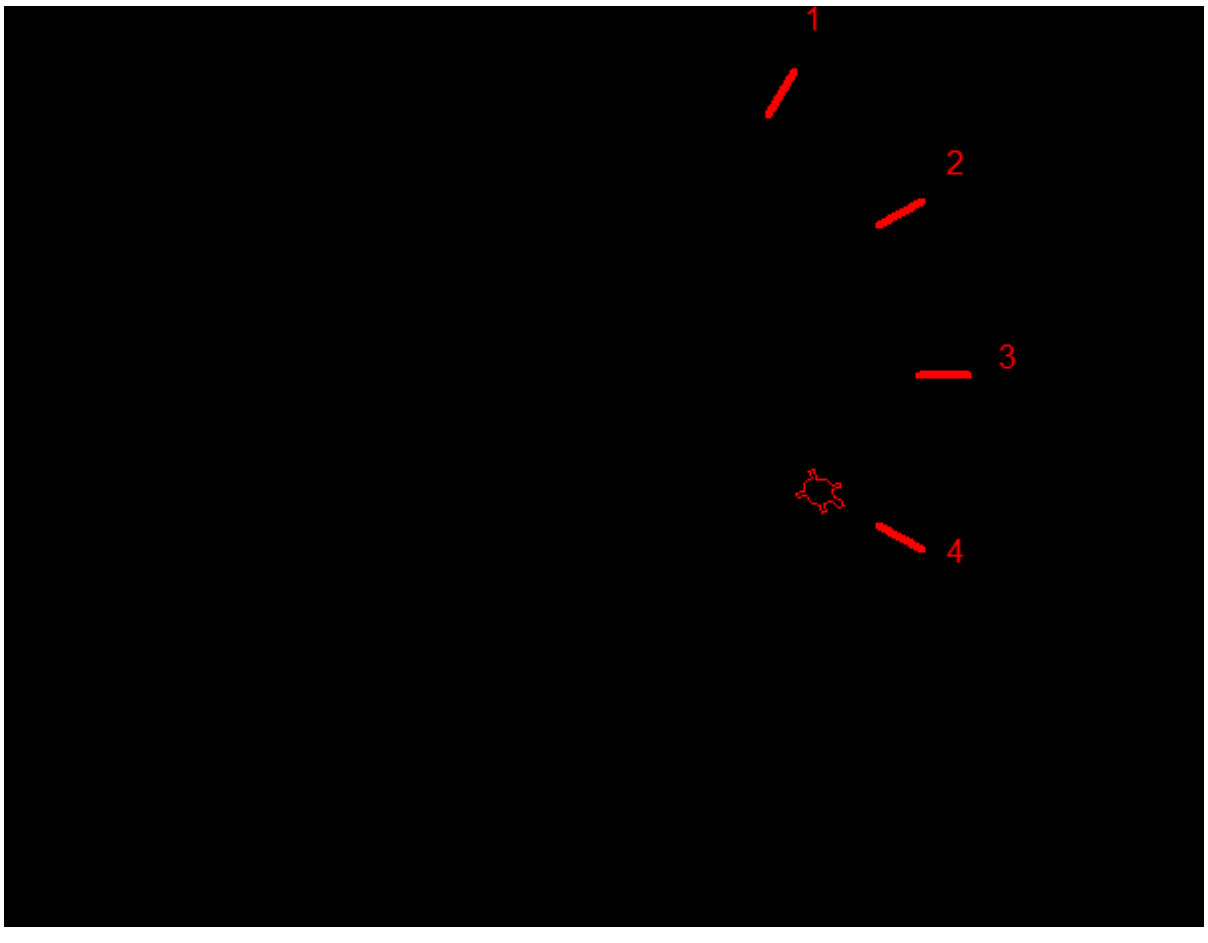


Advanced Animations - a clock dial

In the following animation, Python Turtle is used to construct a clock dial - with numbers, 1 to 12, surrounded by a circle. This is the code:

```
import turtle
screen=turtle.Screen()
trtl=turtle.Turtle()
screen.setup(620,620)
screen.bgcolor('black')
clr=['red','green','blue','yellow','purple']
trtl.pensize(4)
trtl.shape('turtle')
trtl.penup()
trtl.pencolor('red')
m=0
for i in range(12):
    m=m+1
    trtl.penup()
    trtl.setheading(-30*i+60)
    trtl.forward(150)
    trtl.pendown()
    trtl.forward(25)
    trtl.penup()
    trtl.forward(20)
    trtl.write(str(m),align="center",font=("Arial", 12, "normal"))
    if m==12:
      m=0
    trtl.home()
trtl.home()
trtl.setpos(0,-250)
trtl.pendown()
trtl.pensize(10)
trtl.pencolor('blue')
trtl.circle(250)
trtl.penup()
trtl.setpos(150,-270)
trtl.pendown()
trtl.pencolor('olive')
trtl.write('Vivax Solutions',font=("Arial", 12, "normal"))
trtl.ht()
```

This is the animation at work:

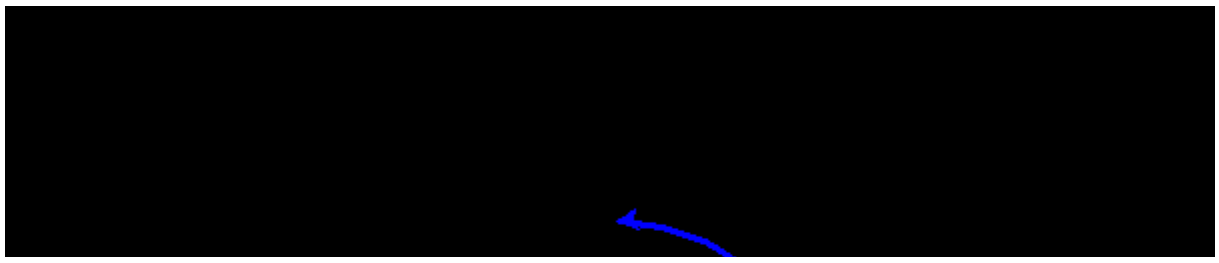


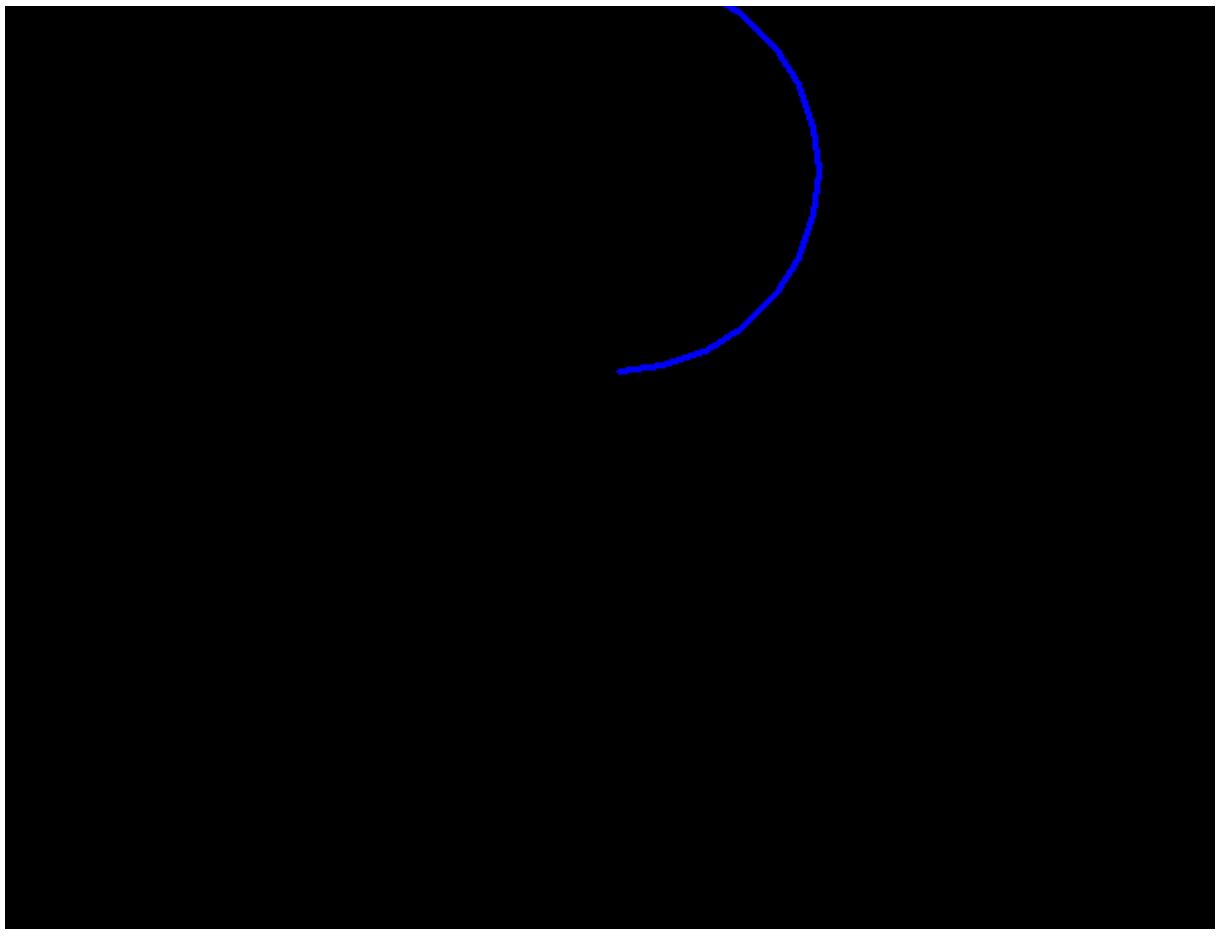
Advanced Animations - multi-coloured flower

The following code produces a flower in different colours. The colours are provided with the aid of a list.The code is as follows:

```
trtl=turtle.Turtle()
screen=turtle.Screen()
screen.setup(620,620)
screen.bgcolor('black')
trtl.pensize(3)
trtl.speed(10)
n=-1
for angle in range(0,360,15):
    n=n+1
     if n==5:
     n=-1
     trtl.color(colors[n])
     trtl.seth(angle)
     trtl.circle(100)
trtl.penup()
trtl.setpos(150,-270)
trtl.pendown()
trtl.pencolor('olive')
trtl.write('Vivax Solutions',font=("Arial", 12, "normal"))
trtl.ht()
```

This is the animation:

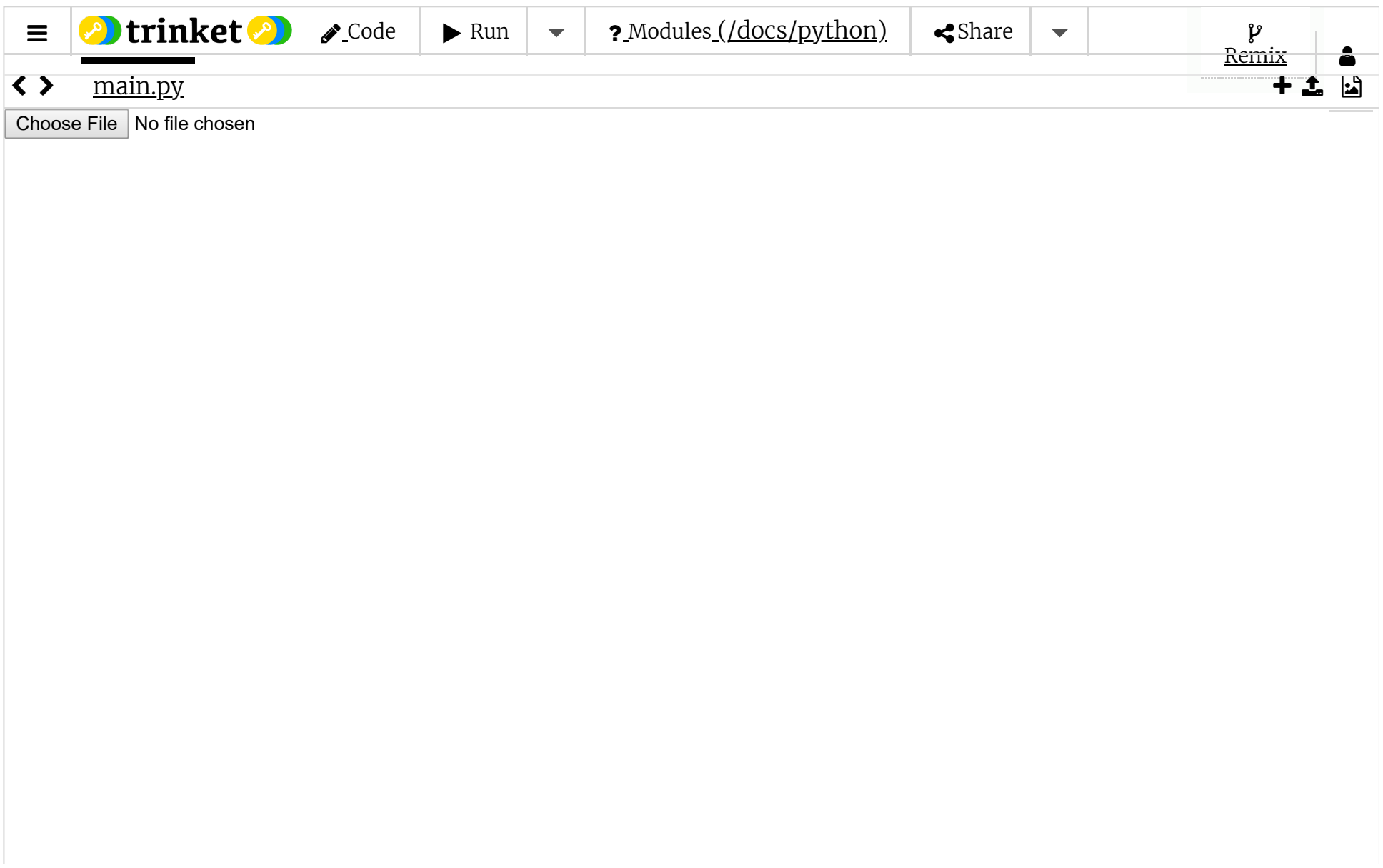




As you can see, the effect is produced by **turtle.seth()** function - in turning the direction of turtle.

Drawing Multiple Squares

In this animation, predetermined number of squares are drawn with the aid of a function. The function has two parameters - length of the square and the colour. Then, using a for loop, we can draw the number of squares we need. You can change the length and colour in order to practise interactively.



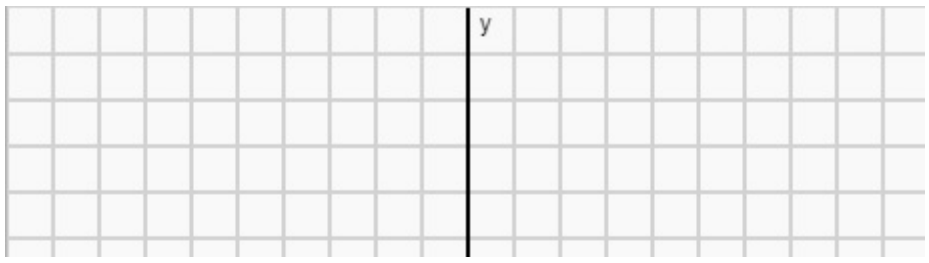
Creating a Graph Paper

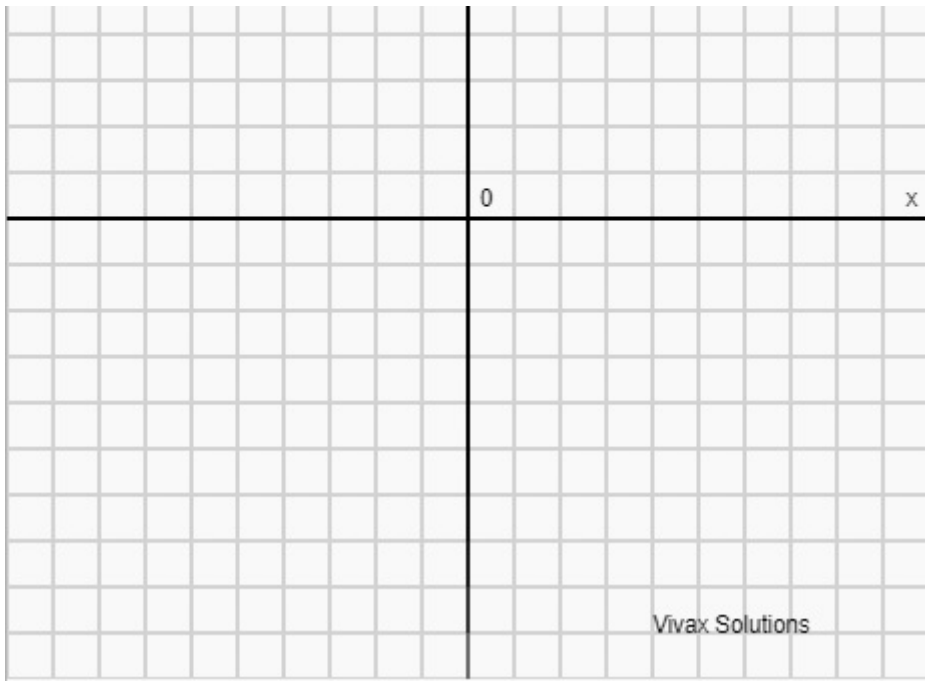
With the code given below, you can create a graph paper easily with Python Turtle:

```
import turtle
```

```
trtl=turtle.Turtle()
trtl.speed(10)
for i in range(0,400,20):
trtl.pencolor('lightgrey')
    trtl.penup()
    trtl.setpos(-200+i,-200)
    if i==0:
      trtl.left(90)
    trtl.pendown()
    trtl.forward(400)
    trtl.backward(400)
for i in range(0,400,20):
    trtl.pencolor('lightgrey')
    trtl.penup()
    trtl.setpos(-200,-200+i)
    if i==0:
      trtl.right(90)
    trtl.pendown()
    trtl.forward(400)
    trtl.backward(400)
trtl.penup()
trtl.penup()
trtl.home()
trtl.pendown()
trtl.pencolor('black')
trtl.backward(200)
trtl.forward(400)
trtl.backward(200)
trtl.left(90)
trtl.forward(200)
trtl.backward(400)
trtl.penup()
trtl.setpos(5,5)
trtl.pendown()
trtl.write(0)
trtl.penup()
trtl.setpos(190,5)
trtl.pendown()
trtl.write("x")
trtl.penup()
trtl.setpos(5,190)
trtl.pendown()
trtl.write("y")
trtl.penup()
trtl.setpos(80,-180)
trtl.pendown()
trtl.write("Vivax Solutions")
trtl.ht()
```

With the above code, the following can be produced:





You can practise it interactively here:

trinket Code Run Modules (/docs/python) Share Remix

main.py

Choose File No file chosen

The Code for Fifty Shades of Grey Animation

In order to produce the animation at the top of this tutorial, please follow this code:

```
import turtle
turtle.penup()
turtle.setpos(-100,250)
turtle.pendown()
turtle.pencolor('olive')
turtle.write('Fifty Shades of Grey',font=("Arial", 18, "bold"))
turtle.penup()
turtle.setpos(0,0)
turtle.pendown()
turtle.color("black", "white")
turtle.colormode(1.0)
SQUARES = 50
SIDE = 150
shade = 1.0
for count in range(SQUARES):
```

```
turtle.fillcolor(shade, shade, shade)
turtle.begin_fill()
turtle.left(360 // SQUARES)
for side in range(4):
  turtle.forward(SIDE)
  turtle.left(90)
  turtle.end_fill()
  shade -= turtle.colormode() / float(SQUARES)
turtle.penup()
turtle.setpos(150,-270)
turtle.pendown()
turtle.pencolor('olive')
turtle.write('Vivax Solutions',font=("Arial", 12, "normal"))
turtle.done()
```

Now that you have read this tutorial, you will find the following tutorials very helpful too:

- Python: basic to intermediate - interactive
- Visual Basic - interactive
- Little Man Computer - LMC Tutorial
- Object Oriented Programming - OOP
- HTML Tutorial
- JavaScript Tutorial

must read.

Mathematics

- Bearings
- Venn Diagrams
- Worksheet Generator
- Transformation of Graphs
- Maths Challenge
- Maths Lab
- Statistics Lab
- Basic Differentiation
- Basic Integration
- Use of Casio Calculator for Statistics
- GCSE Mathematics (9-1)

Physics

- Electricity
- Refraction
- Total Internal Reflection
- Electrostatics
- Physics Lab
- Motor Effect
- Eclipses
- Logic Gates
- Terminal Velocity
- Satellites
- Errors and Uncertainties

Web Development

- ASP.Net Charts
- ASP.Net Security
- CSS 3
- HTML5
- HTML5 Canvas Animations
- JavaScript
- Percentage Circles with JavaScript
- Solar System
- Pendulum
- Dragging Elements
- Python for Beginners

Home | About | Maths | Physics | Programming | Contact