Raspberry Pi for Beginners

LESSON 3

MAKERHOUSE

EMPOWERING MAKERS

TABLE OF CONTENTS

PYTHON PROGRAMMING WITH TURTLE

1 INTRODUCTION

You have been introducing with the basics Scratch programming in the previous lesson. With the approach of visual block programming, you achieved to construct a couple of simple programs – moving the scratch cat according to key-pressed, control input and output of Raspberry Pi GPIO -- by drag, drop and joining them to build a complete sequence of instruction. Give a big clap on your success.

Now, we will proceed to the next lesson which is learning the Python programming with Turtle.

1.1 VISUAL INTRODUCTION TO PYTHON

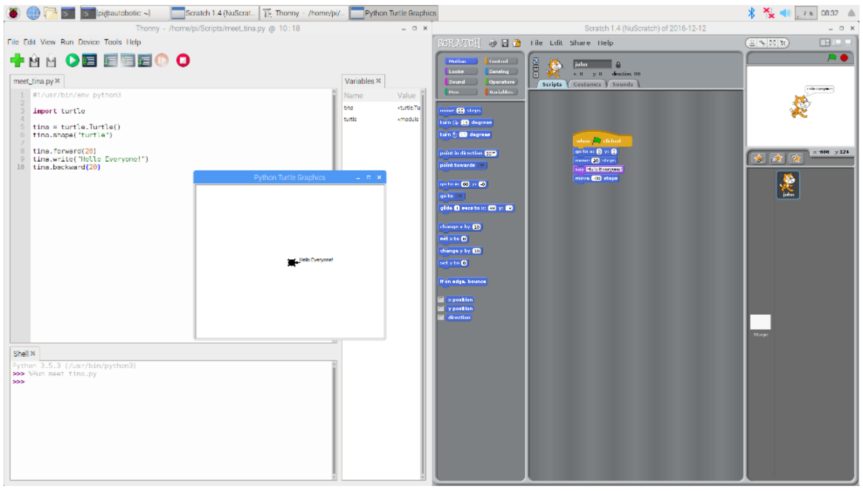
Programming is consisting of some of the computer sciences, some arts and some maths. Thus, throughout this lesson, we will implement Python programming with Turtle module to create some illustrations with it – techniques of programming.

What is Python?

Python is a programming language which is easier to read and write. It is crammed with lots of useful stuff, including this turtle module. Come on; we begin to use the turtle function today!

What is Turtle?

In simple word, it is a "pen" in Python programming. A built-in module in Python where you can draw any shape, the image on the screen, and it is fun. By learning to programme controlling the turtle, it will help ease the transition from visual block programming; Scratch to text-based programming; Python.


Thonny - /home/pi/Scripts/meet_tina.py @ 10:18
File Edit View Run Device Tools Help
meet_tina.py
Variables
Python Turtle Graphics
Hello Everyone!
Shell

```
#!/usr/bin/env python3

import turtle

tina = turtle.Turtle()
tina.shape('turtle')

tina.forward(20)
tina.write('Hello Everyone!')
tina.backward(20)
```

```
Python 3.5.3 (/usr/bin/python3)
>>> %Run meet_tina.py
>>>
```

Figure 1: Scratch vs Python ** Hello Everyone!

1.2 GETTING STARTED

Writing a Python code does not require any special and dedicated software or IDE (Integrated Development Environment) yet using regular text editor is enough. However, there are many available Python IDE for you to use like Thonny which already installed in Raspbian OS.

What IDE do/for?

An IDE (integrated development environment) is a software suite that consolidates the essential tools required to write and test software. For Python beginners like us, Thonny Python IDE may help a lot.

1.2.1 THONNY PYTHON IDE

As usual, there are two ways to open the software – GUI and Terminal. Same goes here for Thonny. Did you remember?

1. Open Thonny using GUI:

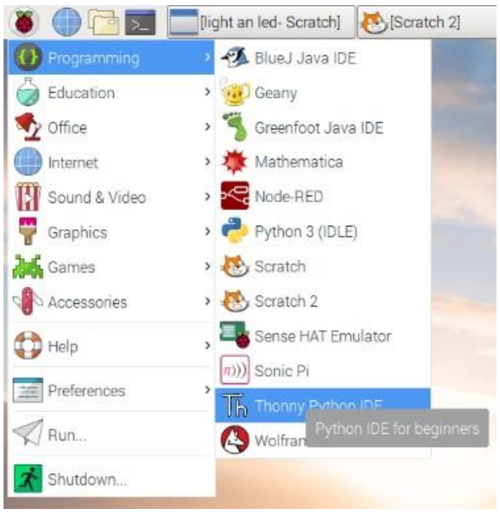
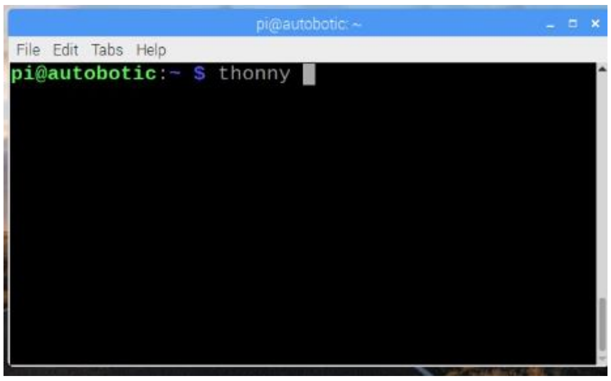


Figure 2: **Raspberry Pi Icon** > **Programming > Scratch**.

2. Open Thonny using Terminal:

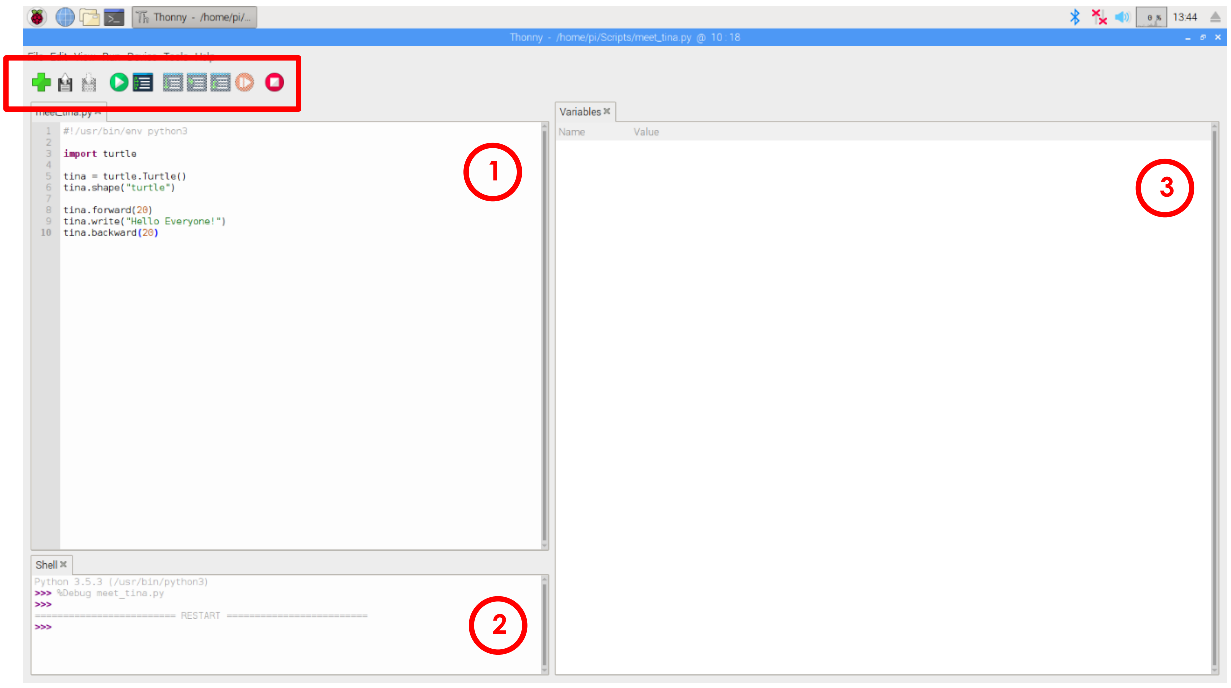


```
pi@autobotic:~ $ thonny
```

Figure 3: On terminal, type **thonny**. Enter.

1.2.1.1 IDE INTERFACE

Here are the basics Thonny Python IDE interface – beautiful, sleek, and simple – everything you need is just started to type a code and run it. That's all. Moreover, it is also provided with the code line by line debugging mode, which is for us the beginner to know and understands on each line of code typed was doing.



```
#!/usr/bin/env python3

import turtle

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

tina.forward(20)
tina.write("Hello Everyone!")
tina.backward(20)
```

```
Python 3.5.3 (/usr/bin/python3)
>>> %Debug meet_tina.py
>>>
======================== RESTART ========================
>>>
```

Figure 4: Thonny IDE interface

There are three main area in Thonny:

No	Area	Description
1	Script	Type your code here
2	Variables	All the Variables you used, will update here
3	Shell	Simple shell for you to test a pythonic code

Table 1: Interface area description

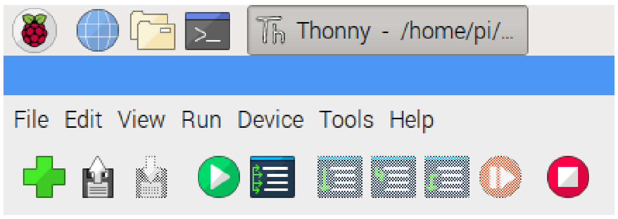


Figure 5: Button configuration – new file, save, load, run code, debugging (step over, step into, step out), continue, and stop.

1.2.2 TURTLE MODULE

Python build in module; Turtle is good starting point for beginners to learn Python – from Scratch to Python -- as in Figure 1. The turtle module instruction allows us to be experimenting a little bit of mathematics and computer sciences to produce a beautiful and abstracts art.

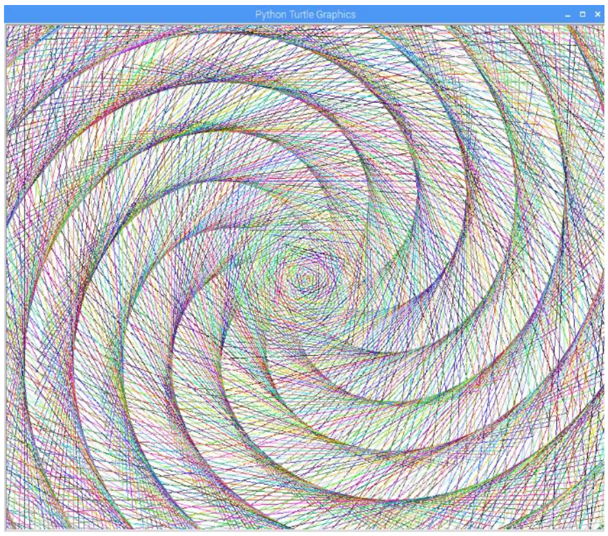


Figure 5: A beautiful arts made with Turtle; Python code – it not only arts, it compromises of computer science and maths too.

1.2.2.1 TURTLE MODULE INSTRUCTION (METHOD)

Python programming require a text instruction command. It is unlike Scratch where each instruction categorized by color; coded instruction. Below is

No	Instruction (Method)	Parameter	Description
		Create an instance	
1	Turtle()	None	Creates and return new Turtle
2	Screen()	None	Create and return new Screen
		Shape	
3	shape()	Name	Set turtle shape to shape with given name; return current shapename
		Drawing	
4	penup()	None	Pull the pen up – no drawing on moving
5	pendown()	None	Pull the pen down – drawing when moving
6	circle()	Radius	Draw a circle with a given radius
		Movement	
7	forward()	Distance	Moves the turtle forward by the specified distance

8	backward()	Distance	Moves the turtle backward by the specified distance
9	right()	Angle	Turn turtle right by angle units
10	left()	Angle	Turn turtle left by angle units
11	goto()	Grid – (x, y)	Move turtle to an absolute position
		Color	
12	color()	Color	Return or set the pencolor and fillcolor
13	colormode()	1.0 or 255	Return or set colormode of the Screen
14	bgcolor()	Color	Return or set background of the Screen
15	begin_fill()	None	Called just before drawing a shape to be filled
16	end_fill()	None	Fill the shape drawn after the call begin_fill()
		Write	
17	write()	String	Write text at the current turtle position

Table 1: Commonly use Turtle instruction.

There is a lot of instruction available under Turtle module. I can't remember it all and listed it here. However, there is a way to find and search for the help – using the Terminal.

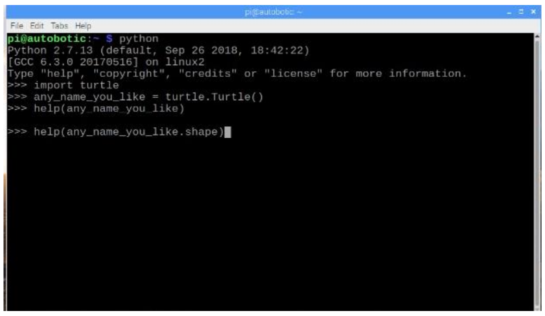
Pheeeewwwwww!

1.2.2.2 MORE INSTRUCTION (METHOD)

Open the Terminal. If you already open the Terminal (used to open Thonny), just hit **ctrl+shift+t** to open a new tab. Else, hit **ctrl+alt+t** for a new Terminal.

How to find help on Turtle module:

1. Type **python** and **Enter**.
2. Next, type **import turtle**. Enter.
3. Create an instance for new turtle; **any_name_you_like = turtle.Turtle()**. Enter. Python Turtle Graphic window will appear once hit Enter. Ignore it.

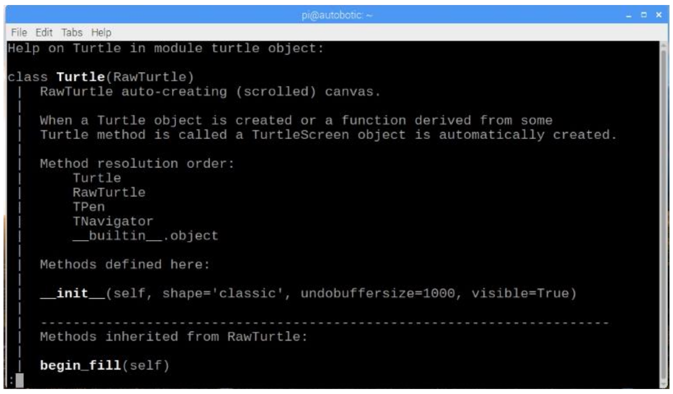


```
pi@autobotic:~ $ python
Python 2.7.13 (default, Sep 26 2018, 18:42:22)
[GCC 6.3.0 20170516] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import turtle
>>> any_name_you_like = turtle.Turtle()
>>> help(any_name_you_like)

>>> help(any_name_you_like.shape)
```

Figure 6: Python environment in Terminal

4. Then, type **help(any_name_you_like)**. Enter. **Once Enter, the Terminal will listed all the available instruction (method) available – a long list – scroll with mouse cursor or up and down key. To exit from the help, hit "q" – will resume in Python environment.**



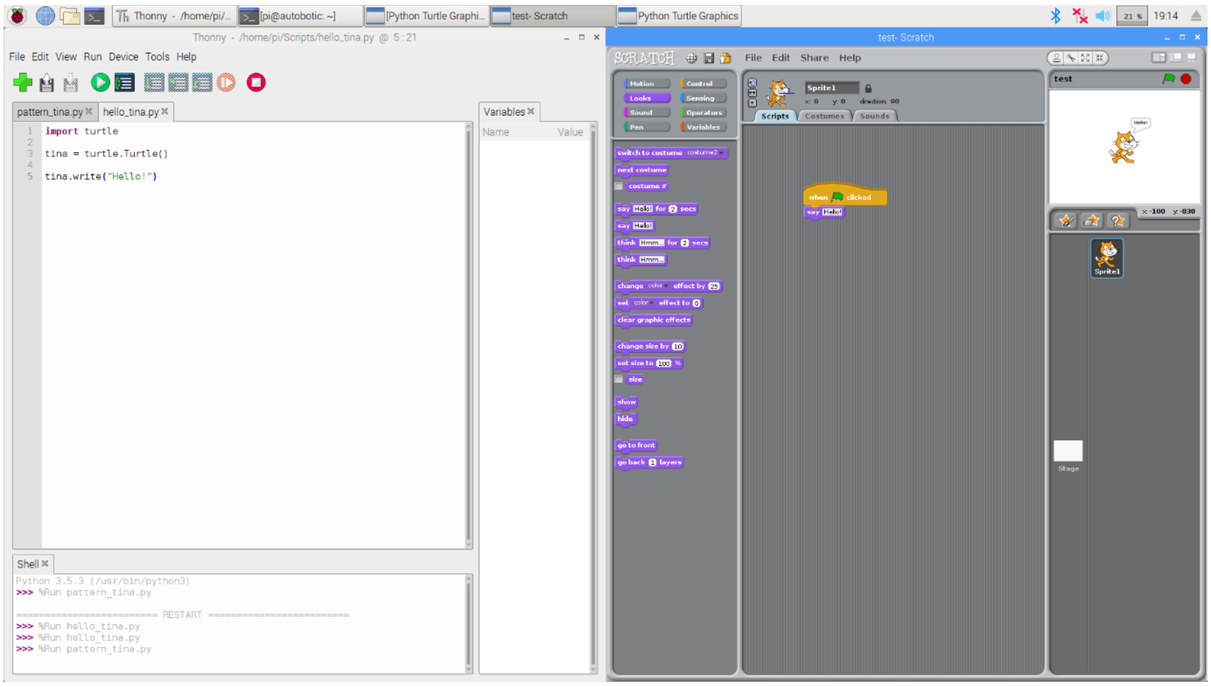
```
Help on Turtle in module turtle object:

class Turtle(RawTurtle)
 |  RawTurtle auto-creating (scrolled) canvas.
 |
 |  When a Turtle object is created or a function derived from some
 |  Turtle method is called a TurtleScreen object is automatically created.
 |
 |  Method resolution order:
 |      Turtle
 |      RawTurtle
 |      TPen
 |      TNavigator
 |      __builtin__.object
 |
 |  Methods defined here:
 |
 |  __init__(self, shape='classic', undobuffersize=1000, visible=True)
 |
 |  ----------------------------------------------------------------------
 |  Methods inherited from RawTurtle:
 |
 |  begin_fill(self)
:
```

Figure 7: Finding a help on instruction for **any_name_you_like**. In this case; **help(any_name_you_like.shape).**

1.3 HELLO TURTLE

Let us create the basic program – Hello World! – for both in Python (use Turtle module) and Scratch. Looks at the similarity and differentially – in term of each instructions means.



```
import turtle

tina = turtle.Turtle()

tina.write("Hello!")
```

```
Python 3.5.3 (/usr/bin/python3)
>>> %Run pattern_tina.py

======================== RESTART ========================
>>> %Run hello_tina.py
>>> %Run hello_tina.py
>>> %Run pattern_tina.py
```

Figure 8: Hello Turtle and Hello Scratch code

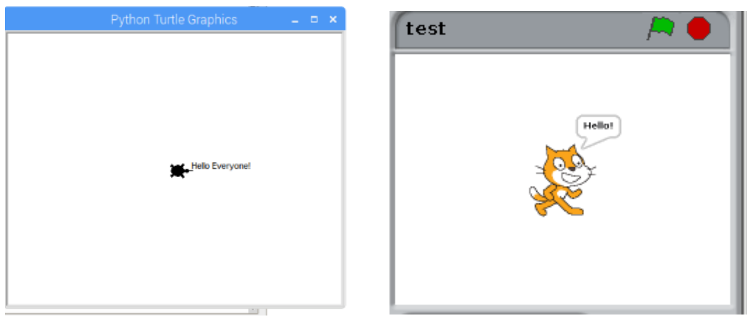


Figure 9: Hello Turtle and Hello Scratch output

Code may always look unfamiliar. To do so, keep going as you will observe, there is similarity. To simplify it, we tabulated it as

No	Scratch	Python (Turtle module)	Description
1	Choose a sprite (image)	import turtle	Selecting a module/sprite to use
2	Rename the sprite – given a name	tina = turtle.Turtle()	Create an instance -- naming
3	Say Hello!	tina.write("Hello!")	Print out "Hello"

Table 2: Scratch code and Python (turtle module) code.

2 CREATIVE TURTLE

We have seen an example of how easy to write a Python code to control our turtle. It is straight forward – readable – programming language. So now let our creativity converted into code.

2.1 TURTLE BASICS

Before we jump into complicated instruction, let us go through with the basics first. Lesson will be taught as a sequence – complete program.

2.1.1 CALL TURTLE

Before we were able to control the turtle, we shall call it first – name it as you wish. Here we name it as "tina"

```
import turtle

tina = turtle.Turtle()
```

Listing 1: Call a turtle name as "tina" – you can choose others name too.

2.1.2 DRAWING A LINE

```
import turtle

tina = turtle.Turtle()

tina.forward(100)
```

Listing 1: Turtle "tina" draw a line.

This code for control/asks the turtle to move forward 100 steps. Try to change with others distance and backward movement.

How if the distance is minus? What will you observe?

2.1.3 TURNING

```
import turtle

tina = turtle.Turtle()

tina.forward(100)

tina.right(90)

tina.forward(100)
```

Listing 2: Turtle "tina" make a right turn.

This code for control/asks the turtle to move forward 100 steps, then turn right 90 degree. Try to change with different distance and angle; left.

With this code, can you draw a pattern – rectangle, triangle, square, and circle?

```
import turtle

tina = turtle.Turtle()

tina.forward(200)

tina.right(90)

tina.forward(100)

tina.right(90)

tina.forward(200)

tina.right(90)

tina.forward(100)
```

Listing 3: Turtle "tina" make a rectangle pattern.

```
import turtle

tina = turtle.Turtle()

tina.forward(100)

tina.right(120)

tina.forward(100)

tina.right(120)

tina.forward(100)
```

Listing 4: Turtle "tina" make a triangle pattern.

```
import turtle

tina = turtle.Turtle()

tina.forward(100)

tina.right(90)

tina.forward(100)

tina.right(90)

tina.forward(100)

tina.right(90)

tina.forward(100)
```

Listing 5: Turtle "tina" make a square pattern.

```
import turtle

tina = turtle.Turtle()

tina.circle(100)
```

Listing 6: Turtle "tina" make a circle pattern.

2.1.4 CHANGING COLORS

As observed. Default color for "tina" drawing is black, while the background is white. Boring. Let us change with our own loved colors – formatted in RGB. Here how

```
import turtle

tina = turtle.Turtle()

bg = turtle.Screen()

bg.colormode(255)

tina.color((0, 0, 255))

bg.bgcolor((255, 0, 0))
```

Listing 7: Changing Turtle "tina" color – Blue – and background -- Red.

Can you change to others color as well – Green, Blue, etc? For color picker: https://www.w3schools.com/colors/colors_rgb.asp

2.1.5 REPETITION

For looping or repetition process, there are two type of instruction – **for** and **while** – are mostly like will be use. To differentiate both two; **while → repeat forever, for → repeat until certain condition is meet.**

```
import turtle

tina = turtle.Turtle()

for i in range(4):

        tina.forward(100)

        tina.right(90)
```

Listing 8: Turtle “tina” make a square pattern (using **for**).

Did you any different with the listing with listing 5? Possible we apply on listing 4 too?

```
import turtle

tina = turtle.Turtle()

for i in range(4):

        tina.forward(100)

        tina.right(120)
```

Listing 9: Turtle “tina” make a triangle pattern (using **for**).

2.1.6 BETTER SPIRALS

```
import turtle

tina = turtle.Turtle()

for i in range(2000):

        tina.forward(i)

        tina.right(120)
```

Listing 9: Turtle “tina” make a repetitive square pattern (using **for**).

2.1.7 LOOPY COLORS

Previous we managed to make a beautiful repetitive square pattern. Let's make it more cool – more colourful randomly. Can we?

```
import turtle

import random

tina = turtle.Turtle()

bg = turtle.Screen()

bg.colormode(255)

for i in range(2000):

        tina.color((random.randint(0, 255), random.randint(0, 255), random.randint(0, 255)))

        tina.forward(i)

        tina.right(90)
```

Listing 10: Turtle "tina" make a repetitive square pattern (using **for**) – full of randomize colour.

Try yourself for other shape; pattern – triangle, circle, etc.

```
import turtle

import random

tina = turtle.Turtle()

bg = turtle.Screen()

bg.colormode(255)

for i in range(2000):

        tina.color((random.randint(0, 255), random.randint(0, 255), random.randint(0, 255)))

        tina.forward(i)

        tina.right(120)
```

Listing 11: Turtle "tina" make a repetitive triangle pattern (using **for**) – full of randomize colour.

```
import turtle

import random

tina = turtle.Turtle()

bg = turtle.Screen()

bg.colormode(255)

for i in range(2000):

        tina.color((random.randint(0, 255), random.randint(0, 255), random.randint(0, 255)))

        tina.circle(i)
```

Listing 12: Turtle "tina" make a repetitive circle pattern (using **for**) – full of randomize colour.

2.2 HAVING FUN WITH PYTHON TURTLE MODULE

Finished with the basics. Before ending these lesson – lets put everything together and make your first beautiful abstract art made with Python (turtle module).

```
import turtle

import random

tina = turtle.Turtle()

bg = turtle.Screen()

bg.colormode(255)

for i in range(10):

        tina.pendown()

        for I in range(200):

                tina.color((random.randint(0, 255), random.randint(0, 255), random.randint(0, 255)))

                tina.forward(i)

                tina.right(i)

        tina.penup()

        tina.goto((random.randint(0, 255), random.randint(0, 255), random.randint(0, 255)))
```

Listing 13: Turtle "tina" make an abstract art – full of randomize colour.

You can make your on – mixed shape; square, circle, triangle or custom shape.