



Learning Python

»»» Advanced Level

# Python Run

► Student's Book



Level

# 3



Rana Dajani



## PythonRun - Advanced Level



Published by **LKD Educational Resources 2022**

Amman - Jordan

Tel: +962 6 5374141

Fax: +962 6 5516404

P.O.Box: 851346

Email: [info@lkd.com.jo](mailto:info@lkd.com.jo)

Web: [www.lkd.com.jo](http://www.lkd.com.jo)

 **Author**

Rana Dajani

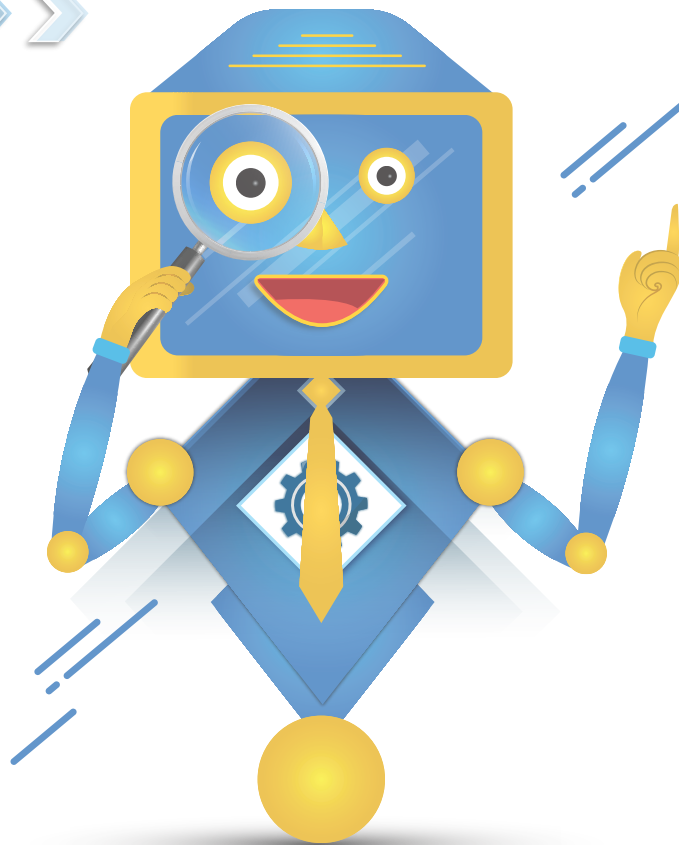
**ISBN: 978-9923-781-06-7**



# Learning Python Advanced Level Python Run

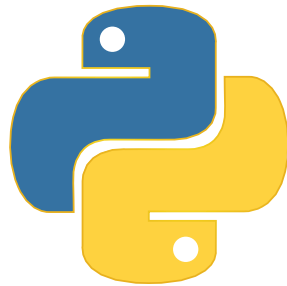


Level  
3



A guide to learning Python programming language

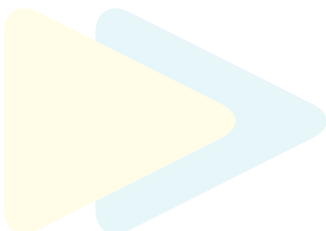
# Introduction

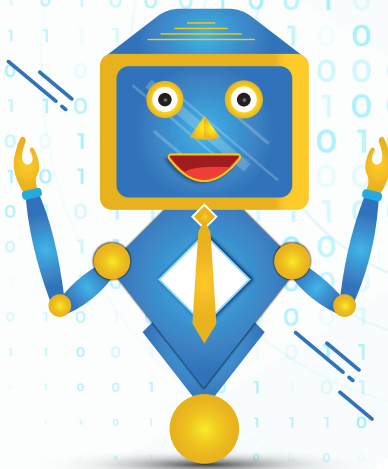


This is the advanced level book in the *PythonRun* series, that takes your coding journey to the next level. This book provides an opportunity to consolidate all the previously learnt skills, from the beginner and intermediate level books, in slightly more challenging material.

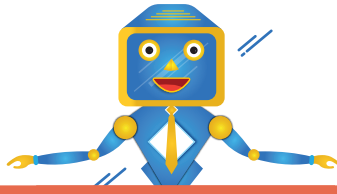
You will learn how to build graphical user interfaces (GUIs) and other fun visual projects. Furthermore, you will discover essential new programming ideas which will make you a much more accomplished coder.

These new skills may ultimately lead you to write the next big app that sweeps the world!





**START CODING  
NOW! .....**



# Table of Contents

## Unit 1 >> Turtles!

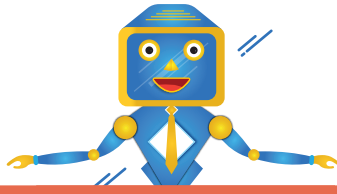
<b>1.1</b> Turtle Module .....	12
<b>1.2</b> Creating a Canvas .....	13
<b>1.3</b> Moving the turtle .....	14
<b>1.4</b> Turtle Position .....	15
<b>1.5</b> Turtle Shape and Stamps .....	16
<b>1.6</b> Clean Canvas .....	17
Quick Tests .....	20

## Unit 2 >> Better Turtle Graphics

<b>2.1</b> Drawing Stars.....	24
<b>2.2</b> Circle Method .....	25
<b>2.3</b> Pen Properties.....	26
<b>2.4</b> Colour Palette.....	27
<b>2.5</b> Fill Shapes.....	30
<b>2.6</b> Canvas Colour and Title.....	31
Quick Tests .....	34

## Unit 3 >> Object Oriented Programming

<b>3.1</b> Object-Oriented Programming .....	40
<b>3.2</b> Classes .....	41
<b>3.3</b> Pass Statement .....	42
<b>3.4</b> All the Things.....	44
<b>3.5</b> Inheritance.....	45
<b>3.6</b> Adding Objects to Classes.....	47

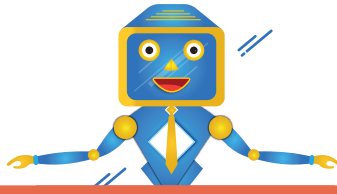


## Table of Contents

<b>3.7</b> Defining Functions of Classes.....	<b>48</b>
<b>3.8</b> Self parameter.....	<b>49</b>
<b>3.9</b> Object Methods.....	<b>50</b>
<b>3.10</b> Functions Calling Other Functions.....	<b>51</b>
<b>3.11</b> Class Variables .....	<b>52</b>
<b>3.12</b> Instance Variables .....	<b>54</b>
<b>3.13</b> Initializing an Object .....	<b>56</b>
<b>3.14</b> Override.....	<b>59</b>
<b>3.15</b> Super() Call .....	<b>60</b>
Quick Tests .....	<b>62</b>

## Unit 4 >> Reading and Writing Files

<b>4.1</b> File Input/Output .....	<b>66</b>
<b>4.2</b> Create a File.....	<b>67</b>
<b>4.3</b> Open() Function .....	<b>68</b>
<b>4.4</b> Close() Function .....	<b>69</b>
<b>4.5</b> Writing on a File .....	<b>70</b>
<b>4.6</b> Automatically Close.....	<b>71</b>
<b>4.7</b> Reading From a File.....	<b>73</b>
<b>4.8</b> Delete a File.....	<b>76</b>
<b>4.9</b> Overwriting on a File .....	<b>78</b>
<b>4.10</b> Letters and Digits in a String.....	<b>79</b>
<b>4.11</b> Upper and Lower Case Strings .....	<b>81</b>
<b>4.12</b> Count Method .....	<b>83</b>
<b>4.13</b> String Split .....	<b>84</b>
Quick Tests .....	<b>86</b>

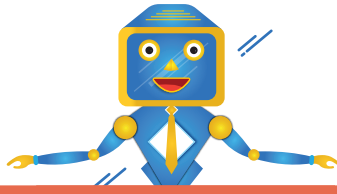


# Table of Contents

## Unit 5 >> Graphics Tkinter

<b>5.1</b>	SIMPLE Graphical User Interface .....	90
<b>5.2</b>	TKINTER MODULE.....	91
<b>5.3</b>	Tk() Class .....	92
<b>5.4</b>	Tk Window Title and Size .....	93
<b>5.5</b>	Tk Window Restrictions.....	94
<b>5.6</b>	Tk Mainloop .....	95
<b>5.7</b>	Widgets .....	97
<b>5.8</b>	Organising Layout and Widgets .....	108
<b>5.9</b>	Geometry Management .....	109
<b>5.10</b>	Delete Widget in Parent .....	117
<b>5.11</b>	Frames Widget .....	118
<b>5.12</b>	Widgets Configure Options .....	122
<b>5.13</b>	Widgets Parent .....	123
	Quick Tests .....	124



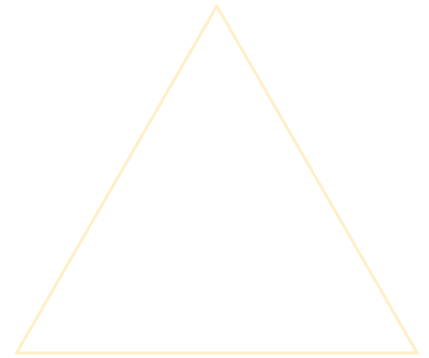


## Table of Contents

### Unit 6 >> Tkinter Canvas

<b>6.1</b>	Canvas Widget .....	<b>130</b>
<b>6.2</b>	Create Canvas Elements .....	<b>134</b>
<b>6.3</b>	Delete Canvas Element .....	<b>141</b>
<b>6.4</b>	Move Elements .....	<b>142</b>
<b>6.5</b>	After Method.....	<b>143</b>
<b>6.6</b>	Event Handling .....	<b>145</b>
<b>6.7</b>	Bind Method .....	<b>146</b>
	Quick Tests .....	<b>148</b>

### >> Project - Based Assessments



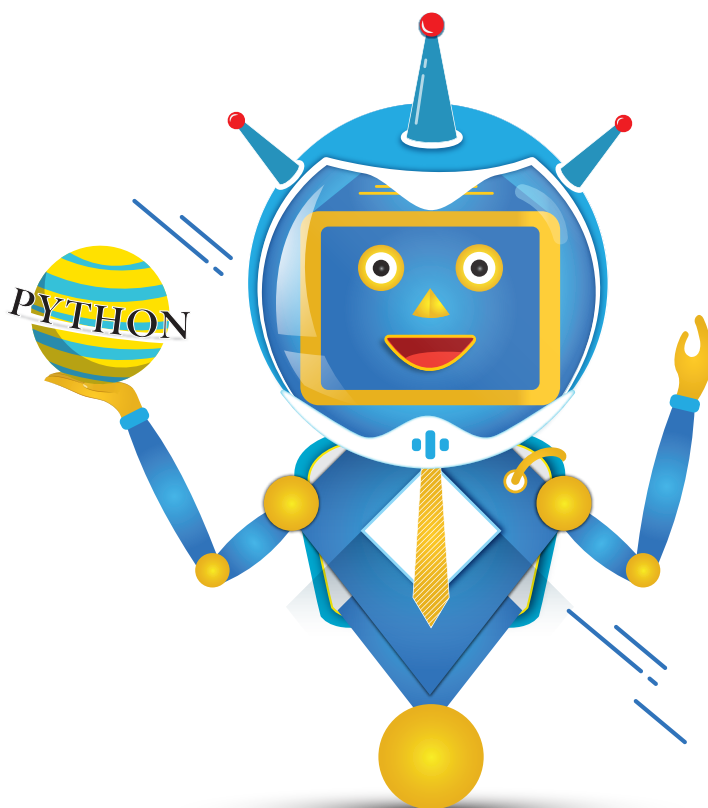
# Unit 1 >>

## >> Turtles!

- >> Turtle Module
- >> Creating a Canvas
- >> Moving the turtle
- >> Turtle Position
- >> Turtle Shape and Stamps
- >> Clean Canvas
- >> Quick Tests



# Turtles!



66

1.1

## Turtle Module



The turtle module is a library that enables users to basically just draw with simple lines, dots, curves, and create pictures and shapes by providing them with a virtual canvas. It is a way of programming vector graphics

```
>>> import turtle as t
```

- ▶ The on screen pen that you use for drawing is called the **turtle**.
- ▶ The turtle has certain changeable characteristics, like size, colour and speed.

### Note:

Make sure you do not have a file of your own lying around named turtle.py as this will interfere with loading Python's own turtle.py library.

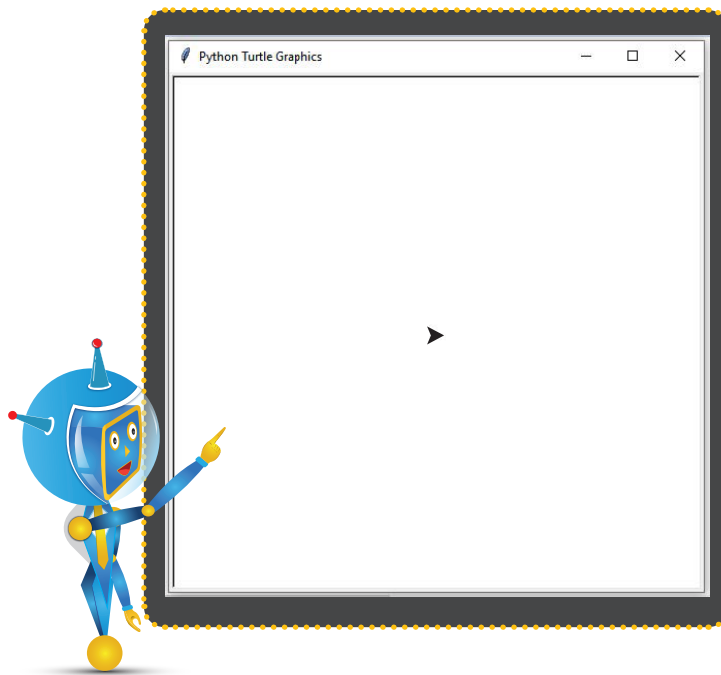
## 1.2

### Creating a Canvas

We need to create a canvas—a blank space to draw on, like an artist's canvas. **Pen()** function from the turtle module, automatically creates a canvas. You should see a blank box (the canvas), with an arrow in the center.

```
>>> import turtle as t
```

```
>>> t.pen ()
```



**To draw with the turtle pen:**

- **up():** means that no line will be drawn when it moves.
- **down():** means that a line will be drawn when it moves.

66

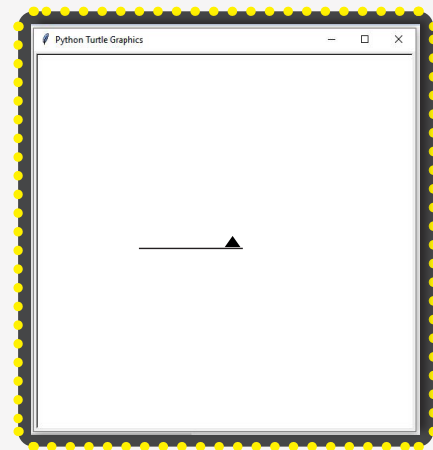
## 1.3

## Moving the Turtle

There are four directions that a turtle can move in:

- **forward (distance) or .fd()**
- **backward (distance) or .bk()**
- **left (degree) or .lt()**
- **right (degree) or .rt()**

```
>>> import turtle as t
>>> t.pen()
>>> t.down ()
>>> t.forward(50)
>>> t.right(30)
```



The turtle arrow moves forward 50 pixels on the screen then turns 30 degrees clockwise.

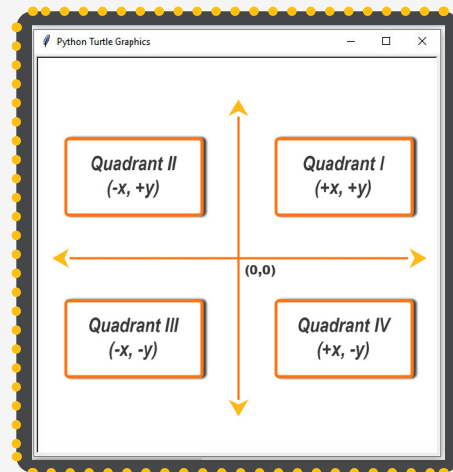
## 1.4

## Turtle Position

The screen is divided into four quadrants. The point where the turtle is initially positioned at the beginning of your program is **t.home (0,0)** . Use **t.pos()** to know the position of the turtle on the screen at any time.

To move the turtle to any other area on the screen, you use **t.goto()** and enter the coordinates like this:

```
>>> import turtle as t
>>> t.up()
>>> t.home()
>>> t.goto(0,20)
>>> t.forward(50)
>>> print(t.pos())
(50,20)
```



► **t.setheading(degree)**: turns the turtle to face a particular direction.

66

## 1.5

## Turtle Shape and Stamps

You can change the way the turtle looks.

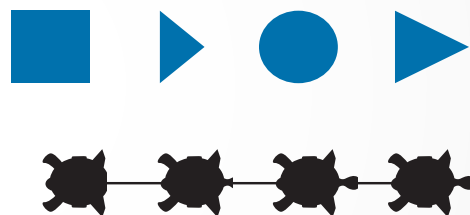
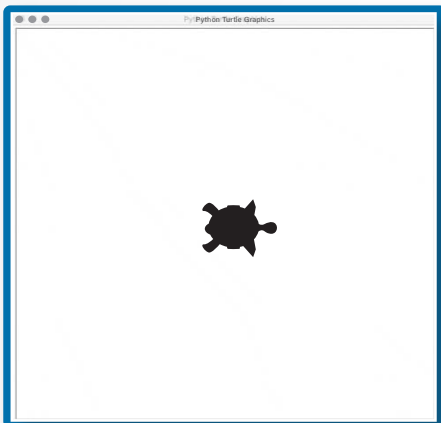
You have a couple of options that you can try :

- ▶ Square
- ▶ Arrow
- ▶ Circle
- ▶ Turtle
- ▶ Triangle

```
>>> t.shape("turtle")
```

You have the option of leaving a stamp of your turtle on the screen, which is nothing but an imprint of the turtle.

```
>>> for i in range (4):
    t.shape('turtle')
    t.stamp()
    t.fd(150)
```





“

## 1.6

### Clean Canvas

```
>>> import turtle as t
```

To erase the canvas, enter `reset`. This clears the canvas and puts the turtle back at its starting position.

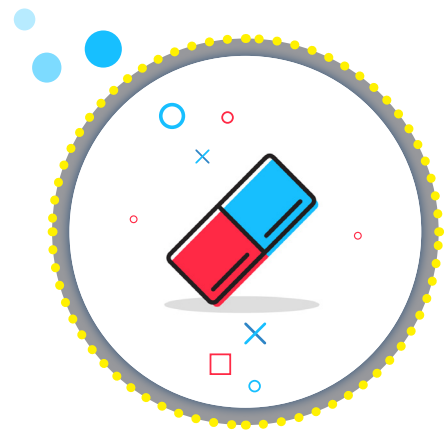
```
>>> t.reset()
```

You can also use `clear`, which just clears the screen and leaves the turtle where it is.

```
>>> t.clear()
```

You can undo the last turtle action.

```
>>> t.undo()
```



If you have other turtles on your screen other than the original turtle, then their drawings will not be cleared out unless you specifically call them out in your code.

## Practice: Turtle Module

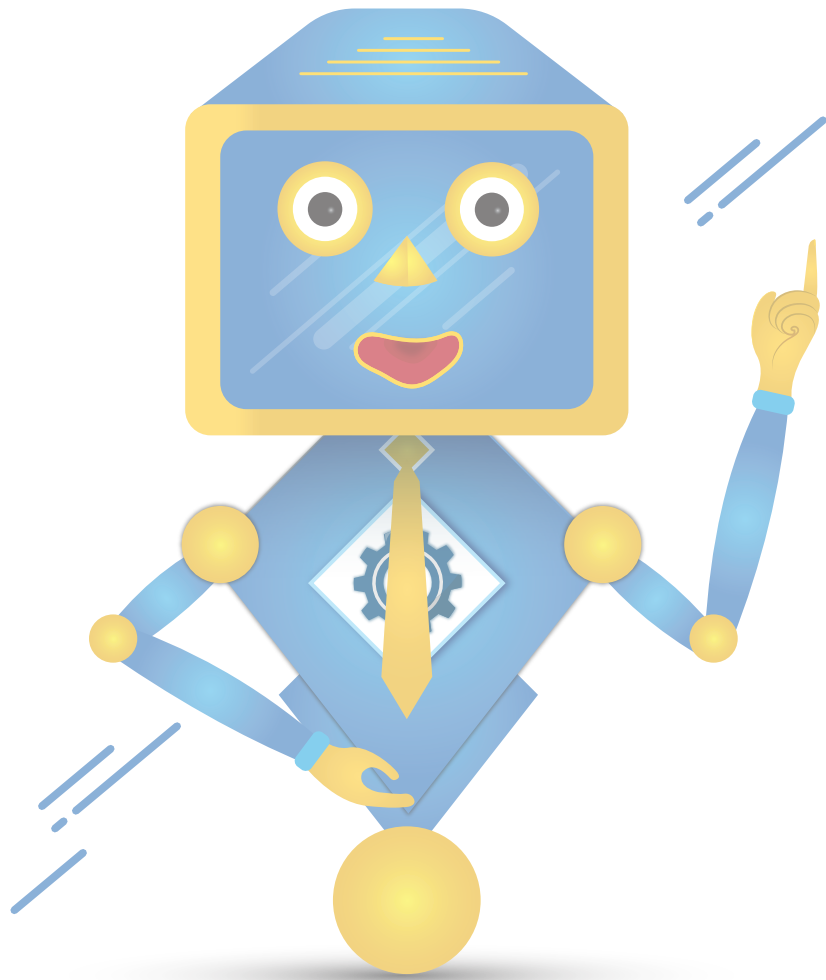
### ► Draw a square with the turtle

```
>>> import turtle as t
>>> for x in range(4):
    t.forward(100)
    t.left(90)
```

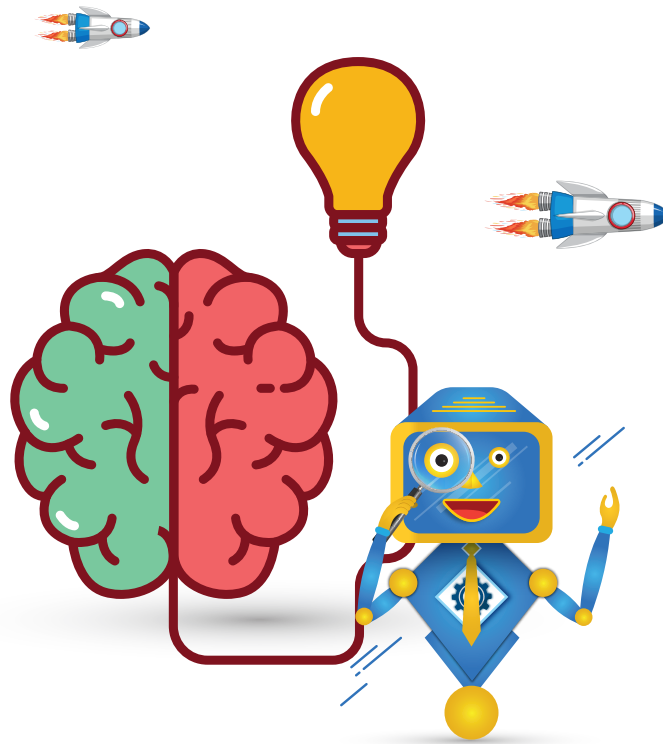
Or

```
>>> t.clear()
>>> t.forward(50)
>>> t.left(120)
>>> t.forward(50)
>>> t.left(120)
>>> t.forward(50)
```



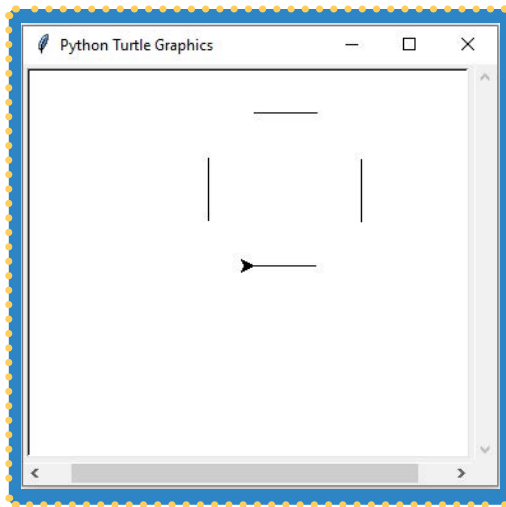


# Quick Tests



## Quick Test: Box Without Corners

Write a program to draw the four lines shown here (the size is not important, just the shape):



## Quick Test: Stamping Trail

Write a program to recreate this pattern:

