CAMBRIDGE
UNIVERSITY PRESS

Coding Club

Python: Programming Art

Supplement 1

Chris Roffey

CAMBRIDGE
UNIVERSITY PRESS

University Printing House, Cambridge CB2 8BS, United Kingdom

Cambridge University Press is part of the University of Cambridge.

It furthers the University's mission by disseminating knowledge in the pursuit of education, learning and research at the highest international levels of excellence.

www.cambridge.org
Information on this title: www.cambridge.org/9781107631090

Printed in Poland by Opolgraf

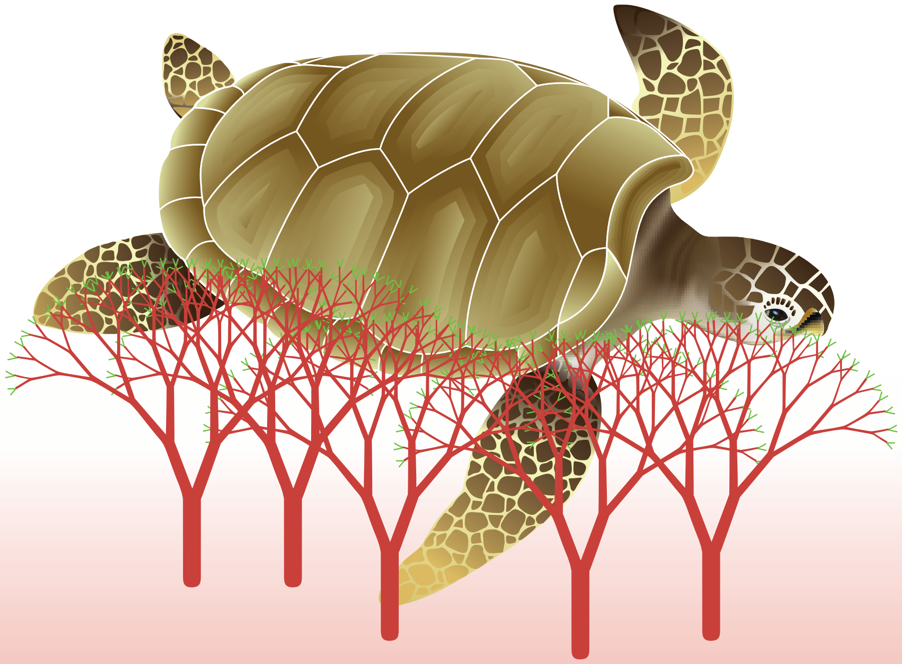
A catalogue record for this publication is available from the British Library

ISBN 978-1-107-63109-0 Paperback

Contents

Section One

Python's turtle module



Chapter 1
Introducing turtle

In *Coding Club: Python Basics* you learned the fundamentals of programming using Python 3. In this book, you will use that hard-won knowledge to have some fun making some little applications while re-enforcing your knowledge and learning a few more tricks.

Python 3 comes with some great, ready-built **modules** some of which we have already used such as `tkinter` and `random`. Another module we can use is `turtle`. This is an implementation of the turtle graphics part of a complete programming language called Logo which was created for educational use; schools often used it to drive a toy turtle around classrooms. The **commands** available in Python's turtle module are very easy to learn. The fantastic thing about this Python module is that there is nothing new to install and we can combine the turtle commands with the Python language we have already learned.

In this chapter, you will learn how to:

- import the turtle module
- make your turtle move around in all directions
- change what the turtle looks like.

The original Logo programming language was developed by Daniel G. Bobrow, Wally Feurzeig, Seymour Papert and Cynthia Solomon in 1967.

Hello World

We are introducing a new module, so why not start with a Hello World program to test and see how turtle graphics works? Well, one reason would be that we are going to use our turtle to draw the letters rather than use text! However, here we go anyway.

Open **IDLE** from your Python 3 install and then open a new window by selecting *File* then *New Window*. Type out the code from Code Box 1.1 and save the file as `hello_world.py` into your *Python Code* folder which you created when reading *Python Basics*. Run the program and see what happens. It really is, almost, a 'hello world' program!

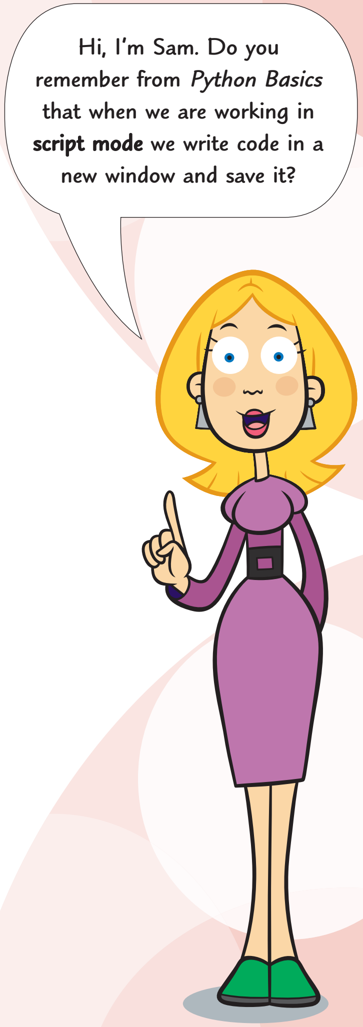
Delving Deeper

Turtle is not a new language; it is a Python module. What this module does, though, is give access to many of the commands of the turtle graphics part of the Logo language. This does not mean we cannot write a "Hello World" program. It is quite common to do so when learning new aspects of a language. For example, when learning about the **tkinter** module many programmers would write a short program that opens a window and displays the text: "Hello World".

Code Box 1.1

```
# hello_world.py introduces the turtle module
from turtle import *

# change line width
pensize(5)
```

(continues on the next page)


Hi, I'm Sam. Do you remember from *Python Basics* that when we are working in **script mode** we write code in a new window and save it?

```
# change to an actual turtle
shape("turtle")

# draw the letter H
left(90)
forward(100)
back(50)
right(90)
forward(40)
left(90)
forward(50)
back(100)

# move to start of next letter
penup()
right(90)
forward(40)
left(90)
pendown()

# draw the letter i
forward(50)
penup()
forward(25)

# tell Python to stop waiting for turtle instructions
done()
```

Analysis of Code Box 1.1

Comments

Comments are for humans only and begin with the hash symbol #

Modules

Modules are collections of useful code collected in one place. Modules have to be imported before they can be used. In this app we import the `turtle` module. We import it in the same way that we imported `tkinter` in *Python Basics* so that we do not have to precede each command with `turtle`.

Functions and arguments

All of the `turtle` commands in this little program end with brackets. This is because they are **functions**. The code for the functions has been written for us in the `turtle` module. We just need to know how to use them. The following experiments will help you learn how to do this. They are very easy. `penup()` and `pendown()` do not require any arguments.

Arguments are pieces of information required by a function so that it can perform its task. E.g. `forward(100)` – the argument `100` is required by the function `forward()` so it knows how far to move.

done()

The turtle module is sometimes not very stable and can crash on Mac, Windows and Linux PCs under different circumstances. To avoid this, it is best to avoid using turtle in **interactive mode** and always end your code with `done()`. This function tells Python that it has finished using turtle and so it stops waiting for turtle commands. Some school environments might still have a few problems depending on what permissions are granted to students. If you find a turtle window will not respond, quit IDLE and restart.

Experiment

Open your `hello_world.py` app and try these experiments:

1. Change the value in the `pensize()` function and run the app again. You can use any value between 0 and 10.
2. Try adding a hash symbol in front of the `shape()` function like this:

   ```
   # change to an actual turtle
   # shape("turtle")
   ```

   This is called **commenting out** code.
3. Delete the hash symbol and instead try changing turtle's appearance with any of the following **strings**: `"arrow"`, `"circle"`, `"square"`, `"triangle"`, `"classic"`
4. Try changing the **integers** in any of the `forward()` or `back()` functions to see what happens.
5. Try changing the arguments in the `left()` and `right()` functions. The numbers you are supplying are the angles to turn in degrees.

Typing less

To make the code easier to read, this book uses the following four turtle commands:

`forward()`, `back()`, `left()` and `right()`.

There are however, shorter alternatives available that you can use if you want to instead:

`fd()`, `bk()`, `lt()` and `rt()`

Thus the code in Code Box 1.1 could be written like the code in Code Box 1.2 instead:

Code Box 1.2

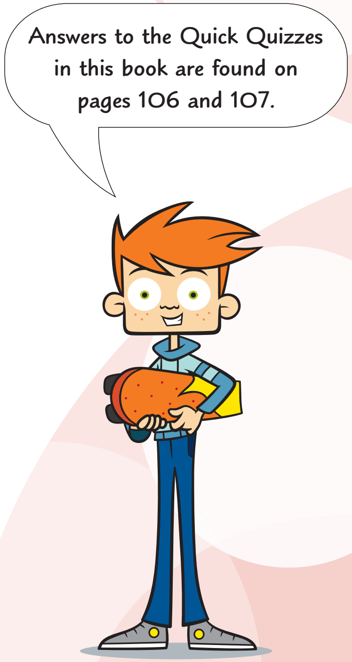
```
# hello_world.py introduces the turtle module
from turtle import *

# change line width
pensize(5)

# change to an actual turtle
shape("turtle")

# draw the letter H
lt(90)
fd(100)
bk(50)
rt(90)
```

(continues on the next page)

```
fd(40)
lt(90)
fd(50)
bk(100)

# Move to start of next letter
penup()
rt(90)
fd(40)
lt(90)
pendown()

# Draw the letter i
fd(50)
penup()
fd(25)

# tell Python to stop waiting for turtle instructions
done()
```

Quick Quiz 1

1 Can you still write a simple "Hello World" program in interactive mode?
2 Try and write a simple hello world program in **script mode** that prints "Hello World" in a small window using the tkinter module.


Answers to the Quick Quizzes in this book are found on pages 106 and 107.

Chapter summary

In this chapter, you have revised:

- variables
- functions
- modules.

You have also learned how to:

- import the turtle module
- make the turtle move around a screen
- lift and lower the pen.

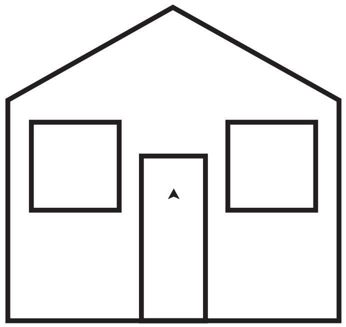
You have learned enough about the turtle module to do all sorts of drawings now. Here are a few quick ideas. In the next chapter we will learn a few more turtle features so that you can really start to be creative.

Idea 1

Try and draw a square.

Idea 2

Try and draw a house with a door and 2 square windows.



Idea 3

If you are using this book in school, you could get everyone in your class to create two letter functions each, for example `upper_h()` and `lower_h()`. These could then be copied into a single file that could become your own turtle font module. A simple program would then look like this:

```
# hi.py
# An example of how to use your turtle_font module
from turtle import *
import turtle_font
```

(continues on the next page)

```
# change to an actual turtle
shape("turtle")

# change line width
pensize(5)

# Move to start
penup()
setup(width=600, height=300)
setposition(-250,0)
pendown()

# Write Hi
turtle_font.upper_h()
turtle_font.lower_i()

# Hide the turtle after completing the message
hideturtle()

# Tell tkinter to stop waiting for turtle instructions
done()
```

Hint: Make sure every letter function lifts the pen, moves right 80 pixels and then puts the pen down.

Answers and all the source code for this book can be downloaded from the companion website www.codingclub.co.uk