Introduction to Python2.7

David Croft

September 7, 2017

Abstract

This week we are going to be looking at the Python programming language and learning how to write simple programs in it that will draw shapes on the screen.

1 Introduction

The first thing that you are going to need to do is open up Python. It is important that we open up Python2 and **NOT** Python3, while the languages are both similar there are some important differences that can cause surprising problems. The document Python-InstallingRunning.pdf (available on Moodle) covers installing Python on your own machine and running it in general.

Go to the file menu and click "new file".

The first thing that we are going to do, is get our turtle to draw a shape. Try copying the writing in the box on the right → into Python. When you have finished click on "Run module" option in the "Run" menu or press F5.

```
import turtle

turtle.forward(100)
turtle.left(120)
turtle.forward(100)
turtle.left(120)
turtle.forward(100)
```

Pre-lab work:

1. What does the turtle draw for us?

2 Moving/Appearance

We've got our turtle to draw a shape, we did this by telling them to move around the screen and they drew a black line as they went. The turtle moved because we gave it commands. If we give our turtle the right commands, in the right order, we can draw anything.

We can move our turtle around the screen by telling them to go forwards, left, right or backwards. If we are telling them to go forwards or backwards, we have to say how far to move. If we are telling them to go left or right, we have to say what angle to turn. We tell our turtle to move by writing commands like the ones on the right → into Python and then telling Python to go run them.

```
turtle.forward(100)
turtle.left(90)
turtle.right(180)
turtle.backward(2)
```

Pre-lab work:

2. Can you get your turtle to turn left and right?
3. Can you get your turtle to draw a square?
   - How many times will we have to turn?
   - How much will we have to turn?

At the moment our turtle doesn't look much like a turtle does it? But we can change this to what we want using the `.shape()` command. The shapes we can have are `"arrow"`, `"turtle"`, `"circle"`, `"square"`, `"triangle"` and `"classic"`. Make sure you put the shape you want in `"` marks because it's text. In Python pieces of text are called strings.

```
turtle.shape("turtle")
turtle.shape("square")
```

We can change the size of our turtle if we use the `.turtlesize()` command. You just have to say how many times bigger you want your turtle to be. **WATCH OUT!** If you make them too big they won't fit on the computer screen!

```
turtle.turtlesize(2)
turtle.turtlesize(10)
turtle.turtlesize(20)
```

Pre-lab work:

4. Can you make your turtle look like a big turtle instead of a boring little arrow?

Your turtle's colour can be also be changed, just say the colour you want but make sure to put it inside `"` marks because it's a string. If we change the colour of the turtle then we also change the colour of the lines that it draws.

```
turtle.color("green")
turtle.color("red")
turtle.color("blue")
```

Our turtle is drawing a line everywhere they go. We know how to change the colour that they are drawing but we can also change the size of the pen they are using. If we give our turtle a big pen then they will draw a big line. We can change the size of our turtle's pen with the `.pensize()` command.

```
turtle.pensize(1)
turtle.pensize(10)
turtle.pensize(20)
```

Pre-lab work:

5. Can you draw a multi-coloured square?
    - With fat lines? With skinny lines?

If you are trying to draw big shapes you might want to make your turtle move faster (not too fast it's still a turtle), or if you just want to see what your turtle is doing you may want to make it move slower.

You can change how fast they are moving with the `.speed()` command. Just tell your turtle if you want it to move `"fastest"`, `"fast"`, `"normal"`, `"slow"` or `"slowest"`.

```
turtle.speed("fastest")
turtle.speed("normal")
turtle.speed("slowest")
```

Pre-lab work:

6. Can you change the speed that your turtle moves?

3 Repeaters/Loops

Lets look at our draw-a-square code. We have to go forwards and turn, forwards and turn, forwards and turn, forwards and turn. We are doing the same thing over and over again which is very boring for us programmers if we have to do all that typing. Wouldn't it be much better if we could get Python to write the same thing over and over for us?

Python has something called a loop, everything that we put inside a loop will happen lots of times, as many times as we tell Python. In the box on the right → you can see a program for drawing a square. It's much smaller than the program you made earlier isn't it? Everything that is underneath the bit where we say `for loop` can happen again and again, but this time it is going to happen 4 times because that's the number we specified.

```
import turtle

for loop in range(4):
    turtle.forward(100)
    turtle.right(90)
```

Lab work:

7. Can you get your turtle to draw a hexagon?
   - How many times will we have to turn?
   - How much will we have to turn?

4 Variables

When we are trying to draw any kind of shape we had to figure out how much to turn. This was pretty easier when we were trying to draw squares because we know that the corners of squares are all 90° but things are more difficult if we are trying to draw other shapes like hexagons.

Fortunately it is easy to figure out how big the angles should be, the corners in all shapes will add up to 360°. So we can find out the size of the corner if we divide 360 by the number of sides that we want.

But it gets even better, instead of us going and getting a calculator to find out the size of the corners. We can get Python to do it for us. All we have to do is type our maths question into Python and it will solve it for us! Watch out, if you want a floating point answer then at least one of the inputs must be a floating point number too.

```
turtle.right( 360/4 )
turtle.right( 360/6 )
turtle.right( 360.0/11 )
```

Lab work:

8. Can you draw a 42 sided shape?
   - It's a tetracontakaidigon!

We've managed to get Python to draw shapes, we've got it loop and do the same thing over and over again, we've even got it to do the maths for us but there's still one annoying thing.

Every time we change the number of sides for our shape, we have to change the number in the `for` loop AND the number where we tell the turtle to turn. It would be a better idea if when we changed it in one place, it changed in both places wouldn't it? Well Python can do that too! We just need to use something called a variable.

A variable lets us save a value and then use it lots and lots of times later on. To make one we just have to say what we want to call the variable (it's name) and then tell it what value we want it to save by using a = sign.
We can just say the name of our variable any time we want to use it. This means that if we change what we have saved in a variable, it also changes for all the places where we use the variable.

```
sides = 3

for loop in range(sides):
    turtle.right( 360/sides )
```

Lab work:

9. Can you change your program to use a variable for the number of sides?

4.1 Functions

Functions allow us to structure our code into blocks, each block does a specific task and can be used multiple times. This makes it much easier to manage our code and means less code overall.

Functions are 'called' by writing the name of the function followed by a pair of brackets (i.e. `print`). Some function will take parameters, these are variables that represent pieces of information that the function needs in order to run. In the case of the print function it takes 1 or more parameters, the values that you want to be printed to the screen.
As well as using functions that already exist, we are able to create our own. This is done using the `def` keyword.
The code that is inside a function will not do anything until the function is called. So make sure that you have a function call otherwise nothing is going to happen.

```
def drawside():
  turtle.forward( 50 )
  turtle.left( 45 )
  turtle.forward( 50 )
  turtle.right( 90 )
  turtle.forward( 50 )
  turtle.left( 45 )
  turtle.forward( 50 )
  turtle.right( 90 )

for i in range(4):
  drawside()
```

5 Going further

This module is only going to cover a small part of what Python is able to do. If you want to know more then feel free to investigate Python and the turtle module in your own time.

In particular resources such as CodeAcademy[a] are very good. Be aware that CodeAcademy uses Python2 not Python3, it's still useful but be aware that there are a few differences.
A good Python3 resource is the book "Automate the Boring Stuff with Python", you can buy a physical copy but it's also available for free online[b].

Investigating some of the more advanced features in Python can only help you in this module so don't be afraid to give it a go and ask questions about it.

Extended work:

10. Try and run the program on the right →.

    - Isn't it amazing what Python can do?

[a] https://www.codecademy.com/
[b] https://automatetheboringstuff.com/

```
import turtle

def drawside( size ):
    if size < 10:
        turtle.forward(size)
    else:
        drawside( size/3 )
        turtle.left( 60 )
        drawside( size/3 )
        turtle.right( 120 )
        drawside( size/3 )
        turtle.left( 60 )
        drawside( size/3 )

sides = 8
size = 200

turtle.speed("fastest")
turtle.penup()
turtle.goto( size/2, size )
turtle.pendown()

for i in range(sides):
    turtle.right( 360.0/sides )
    drawside( size )
```

6 Real world programs

While drawing stuff with the turtle was fun it's not really useful for a professional program. Given that we are working with ubiquitous computing devices they are unlikely to have a display attached and so Graphical User Interface (GUI) interfaces aren't going to be much use.

We are going to focus on writing Command Line Interface (CLI) programs, these are programs that work by reading and writing text messages to a terminal.

6.1 Output

If we want Python to output a simple message then we can use the `print` command.

```
print "Hello World!"
```

You can give any type of variable[1] to the `print()` command and it will output it to the

[1] Covered in section 4 on page 4.

terminal. There are thousands of different types of variables in Python but the main ones are:

- `str` - Called a string, these store text. If you want to create a string variable then you need to make sure that the text is `"inside double"` or `'single quote marks'`.
- `int` - Are whole numbers such as `3` or `42`.
- `float` - Are decimal or floating point numbers such as `12.34` or `0.0000005`.
  - If you divide two ints and the result would be a floating point number then Python will still return an `int`. If you want floating answer at least one of the inputs must be a `float`.
- `bool` - A type of variable that can only have a value of `True` or `False`.
- `None` - A type of variable that can only have a value of `None`.
- `list` - A variable that can stores several other variables inside itself. Equivilent to a vector or array in other languages.

Extended work:

11. Write a program that prints the complete 99 bottles song.
    - `https://en.wikipedia.org/wiki/99_Bottles_of_Beer`