Workshop 14a: Python

Written by CREATIV3.1415i

This workshop focuses on the basics of Python and using it with the Sense HAT.

Difficulty: Introductory workshop

Contents

0 Introduction

Python is a computer programming language designed for general-purpose use and can be found in use almost everywhere. Python was designed to be easy to read and write, making it the perfect language to start off with!

Python has been around since 1991 and grew to be popular in 2003, where it was ranked in the top ten most popular programming languages (and it's still there!). It was the programming language of the year in 2007 and 2010 and even comes pre-installed on your Pi!

This is an introductory workshop. Once you're familiar with using your Pi on the desktop, you should be ready to attempt this workshop.

How to use these booklets

The aim of these booklets is to help you attempt these workshops at home, and to explain concepts in more detail than at the workshop. You don't need to refer to use these booklets during the workshop, but you can if you'd like to.

Code listings & asides

When you need to make changes to your code, they'll be presented in boxes like the following:

```
1 #This is a comment
2 while True:
3         print("Hello, World!")
```

You might not need to copy everything, so check the line numbers to make sure you're not copying something twice. Occasionally, something will be explained in greater detail in asides, like the one below. You can read these as you wish.

Aside

Red lines beginning with number signs (#) are called **comments**. They are ignored by the program, so we can use them to label pieces of code.

What you'll need

For this workshop, you'll just need a working Raspberry Pi running a Linux operating system like Raspbian with Python installed. Luckily for us, Raspbian (and many other Linux distributions) include Python right out of the box.

Everything else

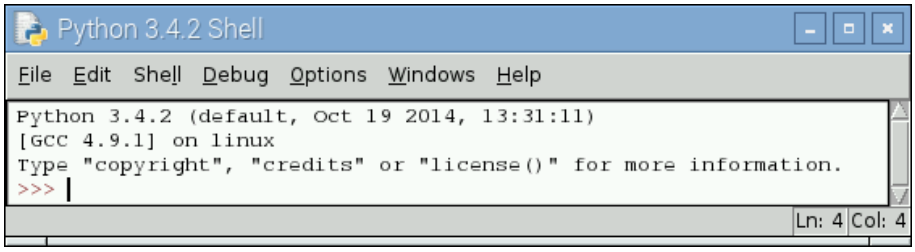
These booklets were created using LaTeX, an advanced typesetting system used for several sorts of books, academic reports and letters.

 Latex source documents are available at http://github.com/McrRaspJam/booklet-workshops

1 IDLE

IDLE is Python's editor, used to edit and run Python programs on your Pi. You can find it under `Programming > IDLE` from the Raspberry Pi logo in the top left of your screen. There may be more than 1 version of IDLE, as Raspbian comes with both Python 2 and Python 3 installed. Python 3 is the preferred choice (although this booklet will work with Python 2 as well), as it has some improvements over Python 2.

IDLE opens into it's shell by default, allowing you to enter code and have it run immediately without having to create and save any files. Note that this shell doesn't save your code for you, so make sure you save any code you want to keep!



Try typing the following code into the shell and hitting enter.

```
print("Hello, World!")
```

You should get the following output after hitting enter.

```
Hello, World!
```

Now, compare that to the equivalent Java program:

```
public class HelloWorld
{
    public static void main(String[] args)
    {
        System.out.println("Hello, World!");
    }
}
```

As opposed to Java, Python is a simple language in which a lot of functionality is implicit and not explicit. There is no need for `System.out.println` or even the `main` function, as Python provides them for you.

Execution

Code in Python is executed from top to bottom in every file, meaning no `main` method is required. Simply placing the code in a file is enough for it to run!

2 The Basics

2.1 Variables

Variables in Python are just names for a value, like a label on a box. Variables can contain any type, and their type and contents can even change throughout the program's execution!

You can declare a variable by simply giving it a name and a value as shown below.

```
name = "Jack"
print(name)
```

In this code we create a variable called `name` containing a person's name and print the variable's contents.

Extra: Why not try changing this to say your name?

2.2 Loops

Loops in programming allow us to run some code multiple times, meaning we don't have to repeat ourselves if we want something doing more than once. There are two types of loops in Python called `for` and `while` loops.

The `for` loop allows us to run some code a specific number of times, an example being printing the numbers 1 through 10.

```
for i in range(1, 11):
    print(i)
```

The above `for` loop runs through a list of numbers from 1 to 10 in order, which is what the `range` function has provided us with.

Extra: Why not try modifying this code to print your name 10 times?

2.3 Conditional Statements

Conditional statements are used to selectively run code based on a condition. This can be useful as we can only run specific code when we want to. In this example we'll change our code so that it only prints out even numbers.

```
for i in range(1, 11):
    if i % 2 == 0:
        print(i)
```

This code goes through each number between 1 and 10 and checks if it is a multiple of 2. If it is then we print it, otherwise we ignore that number and go on to the next one.

Extra: Why not try modifying this code to print multiples of 3?

Indentation

Indentation is how far along the line your code is positioned and it matters a lot in Python! The indentation is needed so that Python knows what code is inside a block (e.g. the `for` loop) and what code isn't. IDLE automatically indents the code for you when starting a block and unindents when it finds an empty line.

3 Exercises

3.1 Input and Output

Make sure you're using IDLE for Python 3 here.

Input and output are fundamental features of almost any programming language and Python is no exception. Python provides us with a simple set of functions to handle input and output, named `input`, `raw_input` (Python 2 only) and `print`.

```
name = input("What's your name? ")
print("You said your name is:")
print(name)
```

This example asks the user for their name, stores their input in a variable and then prints it out.

Extra: Why not try asking for the user's favourite colour too?

`input` and Python 2/3

In Python 2, the `input` function actually *runs* the text that the user inputs, as opposed to just returning it into the variable. `raw_input` actually returns the value the user inputs instead of running it. In Python 3 the `input` function replaces Python 2's `raw_input` and `raw_input` does not exist. You can try this out by running the above example and entering `3 + 4` as your name!

3.2 Randomness

Randomness can be found everywhere in programming, whether it be to work out where to put an enemy in Mario or even find π!

```
import random
print(random.randint(0, 100))
```

Extra: Why not try modifying this code to print a random number between 0 and 1000?

`random`

Python's random library provides a wide range of functions that return random values. Some examples are `random.random()` which returns a random number between 0 and 1, `random.getrandbits` which returns a number of random bits and `random.choice`, which returns a random item from a list.

3.3 Turtle

Python provides a library called `turtle`, which allows us to move a little robot around a virtual world! It's designed to be easy to use and uses plain English for all of it's commands, making it great to play around with.

To get going with Python's `turtle`, we first need to import it so that we can use it's functions.

```
import turtle
```

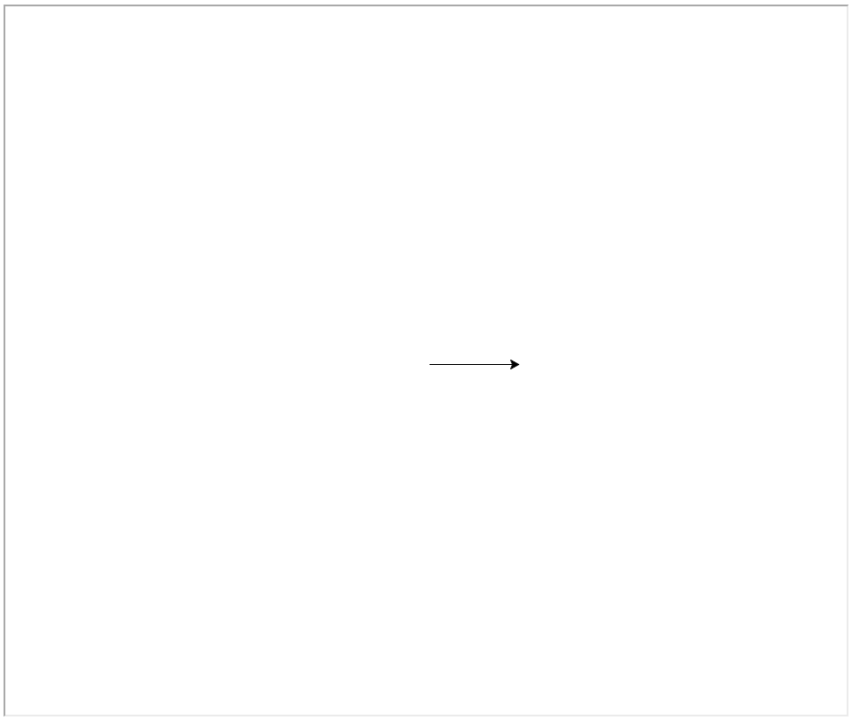
Running this code won't do anything, as we need to move our turtle before a window opens.

Opening the Window

You can get a window to open without a turtle by calling `turtle.setup()`. This function also allows you to set the size of the window and the starting position of the turtle.

Our turtle can only go forwards and turn, so if we want to move in a different direction then we need to turn and face that direction. Our turtle always starts off facing to the right, so in the example below we'll move the turtle to the right a bit.

```
import turtle
turtle.forward(100)
```



IDLE turtles

`turtle` works great in IDLE, as IDLE remains open even after you finish entering your code. Running Python code without IDLE is a different story, as Python exits once it reaches the end of your program! This means that the window will close after your turtle has finished moving, which is not what you want. An easy way to fix this is to add `turtle.mainloop()` at the very end of your code, which ensures

that your program keeps running until you click the close button.

We can also turn our turtle left and right by a certain number of degrees, and joined with our ability to move the turtle in the direction it's facing means we can move in direction! In the example below we draw a rectangle and end up back where we started.

```
import turtle
turtle.forward(100)
turtle.left(90)
turtle.forward(100)
turtle.left(90)
turtle.forward(100)
turtle.left(90)
turtle.forward(100)
turtle.left(90)
```

We can use this alongside what we learned earlier about variables and loops to draw a hexagon!

```
import turtle
for i in range(0, 6):
    turtle.forward(100)
    turtle.left(60)
```

4 The Sense HAT

The Sense HAT is a small board that plugs into the GPIO port on the top of the Pi. It has a small multi-colour LED display, a joystick and some sensors and works great with Python!

4.1 The Sense HAT and Python

The Sense HAT has amazing support for Python, allowing you to display images, access all of the sensors and even work out what direction the Pi is facing! This is all done using the Sense HAT's official library, which comes pre-installed with the Pi.

Connecting to the Sense HAT using Python is surprisingly easy and can be done in just 2 lines.

```
from sense_hat import SenseHat
sense = SenseHat()
```

The first line imports the `SenseHAT` class from the `sense_hat` library (provided by the creators of the HAT). The second line connects to the HAT and stores the connection in a variable which we'll use later on.

4.2 Displaying Some Text

The Sense HAT provides us with an easy-to-use function for displaying text called `show_message`. This function scrolls text across the screen of the HAT whilst allowing you to change the speed and the colour.

```
from sense_hat import SenseHat
sense = SenseHat()

sense.show_message("Hello, my name is Jack!")
```

Running this code will cause the text `"Hello, my name is Jack"` to scroll across the screen of the Sense HAT in white.

You can change the colour by telling the function what the `text_colour` should be.

```
from sense_hat import SenseHat
sense = SenseHat()

sense.show_message("Hello, this is red!", text_colour=[255, 0, 0])
```

Extra: Why not try modifying this code to print your name in green?

Colours of the Rainbow

The Sense HAT accepts colours as a list of three numbers between 0 and 255, allowing you to vary the amount of red, green and blue being shown. Bright red is `[255, 0, 0]`, bright green is `[0, 255, 0]` and bright blue is `[0, 0, 255]`. You can mix these to create over 16 million colours!

4.3 Drawing a Face

The Sense HAT also allows you to set individual pixels on it's screen to certain colours. We're going to use this to draw a smiley face!

```
from sense_hat import SenseHat
sense = SenseHat()

# Here are the eyes.
sense.set_pixel(1, 1, [255, 255, 255])
sense.set_pixel(6, 1, [255, 255, 255])

# Here is the smile.
sense.set_pixel(0, 5, [255, 255, 255])
sense.set_pixel(1, 6, [255, 255, 255])
sense.set_pixel(2, 6, [255, 255, 255])
sense.set_pixel(3, 6, [255, 255, 255])
sense.set_pixel(4, 6, [255, 255, 255])
sense.set_pixel(5, 6, [255, 255, 255])
sense.set_pixel(6, 6, [255, 255, 255])
sense.set_pixel(7, 5, [255, 255, 255])
```

The first two numbers on each `set_pixel` line are the `x` and `y` positions for the pixel and the list of numbers at the end is the colour for the pixel (as we saw earlier when displaying text).

If you make a mistake and the face looks wrong you can use `sense.clear()` to clear the screen and start again.

Extra: Why not try changing it to a sad face?

Find that pixel!

The Sense HAT's coordinate system starts at `0, 0` (top left) and goes up to `7, 7` (bottom right), meaning you should only use `x` and `y` values between these!