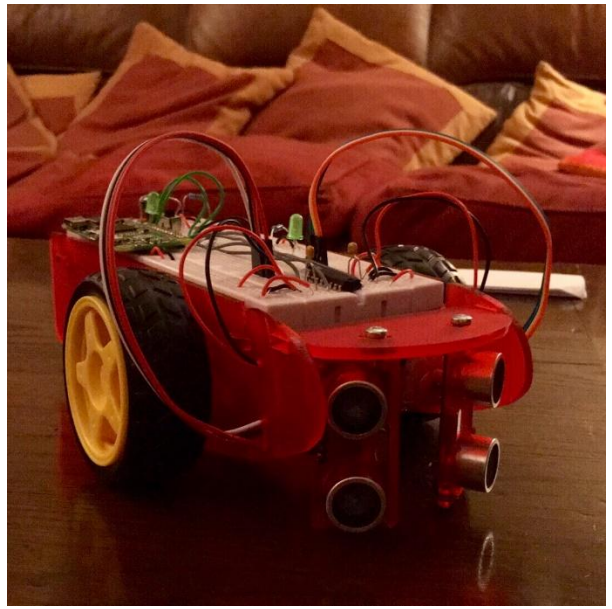
Physical Computing

Course Structure

- The first three sessions after this introduction are on Physical Computing
- Introduction to Python and using it in the real world on the Raspberry Pi and with sensors and motors
- Assessment due at the end of the theme

Course Structure

- Session 1 – Python tutorial and using the Raspberry Pi
- Session 2 – Working in teams using sensors with the Pi
- Session 3 – The challenge



Assessment

- Homework after each session to keep you on track with the work in class (pass/fail)
- Success/Failure document
  - 400 words (+/-20%)
  - Emphasis on understanding of the problems
  - Introduction
  - What was your method to achieving the task?
  - What worked (successes)?
  - What didn’t work (failures), and why?

Python Installation

- Use Python3 if you can (Minimum version 2.7)

- Online editor:
  - https://repl.it/languages/python3

- Linux:
  - http://docs.python-guide.org/en/latest/starting/install3/linux/
- Mac:
  - Should be pre-installed. Run *which python* to check which version.
- Windows:
  - http://docs.python-guide.org/en/latest/starting/install3/win/
  - Download the latest version from https://www.python.org/downloads/

Hello World

- Print statements allow us to print information to the screen
- Python 2:
  - *from __future__ import print_function*
  - *print (“Hello World”)*
- Python 3:
  - *print (“Hello World”)*
- Or try
  - *print (“Hello”, “World”)*
  - *print (‘I can’t print’)*
  - *print (‘I can\’t print’)*

Math

- Lets try some simple maths
- Addition
  - *1+1*
- Subtract
  - *5-3*
- Multiply
  - *2*3*

Math

- Lets try some simple maths
- Divide
  - *5/4*
  - Note that on Python 2 it will round down. To get the correct answer you can use *from __future__ import division*
- Remainder/Modulus
  - *5%4*
- Exponent
  - *5 ** 3*

Python Variables

- Sometimes we need to store data rather than use literal constants
- Variables can hold values of different types called *data types*
- Python is dynamically typed so types are associated with run-time values

```
1 i = 5
2 print (i)
3 i = 10
4 10/i
5 print (i)
6 s = “Hello”
7 print (s)
8 print (s + str(i))     #string concatenation
9
10
```

If Statements

- Used to check a condition
- If the condition is true run a block of statements, else run another block

```
1 a = 1
2 if a > 5:
3     print (“a is more than 5”)
4 else:
5     print (“a is less than 5”)
6
7
```

For Loops

- The for..in statement is a looping statement which iterates over a sequence of objects a set number of times

```
1 for i in range(1, 5):
2     print(i)
3
```

- Range(1,5) gives the sequence [1, 2, 3, 4]. By default, range takes a step count of 1. If we supply a third number to range, then that becomes the step count. For example, range(1,5,2) gives [1,3].

While loops

- The while statement allows you to repeatedly execute a block of statements as long as a condition is true.

```
1 while (i < number):
2     print ("i is " + str(i))
3     i = i + 1
4
5
```

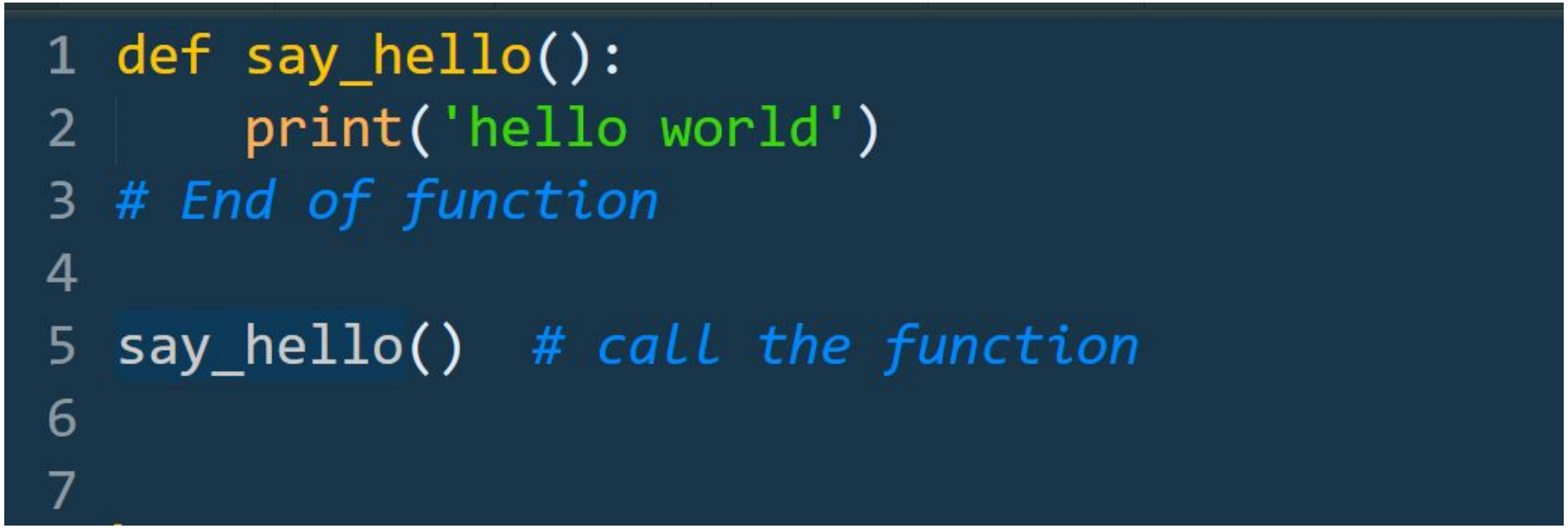
Lists

- A list is a data structure that holds an ordered collection of items i.e. you can store a sequence of items in a list.

```
1 shoppinglist = ['Apple','Banana','Pear']
2 print (shoppinglist)
3 shoppinglist.append('Bread')
4 print (shoppinglist)
5 print (shoppinglist[0])
6
7 for item in shoppinglist:
8     print (item, end = ' ')
```

Functions

- Functions are reusable pieces of programs. They allow you to give a name to a block of statements and run the block by simply calling the name.



```
1 def say_hello():
2     print('hello world')
3 # End of function
4
5 say_hello()  # call the function
6
7
```

Exercises

Homework

- https://python.swaroopch.com/
- http://sthurlow.com/python/

- Go through dictionaries and functions.
- Complete exercises