INTRODUCTION TO PYTHON

Edmond Lascaris and Anne Jessup - Creative Commons – 4 May 2021

OVERVIEW

The Python programming language is one of the most universal computer languages used today. The makers of the Raspberry Pi though so highly of the language that the name "Pi" was named after Python. It is an easy language to learn and will help us to write programs to collect and process data from environmental sensors. We will be using the latest version of Python, which is Python3 (Python 3.7.3).

LEARNING OBJECTIVES

- Learn how to install the Python3 Integrated Development and Learning Environment (IDLE3).
- Run python3 code in the Python Shell Window
- Run python programs in the Python Editor Window

INSTALLING PYTHON3 IDLE

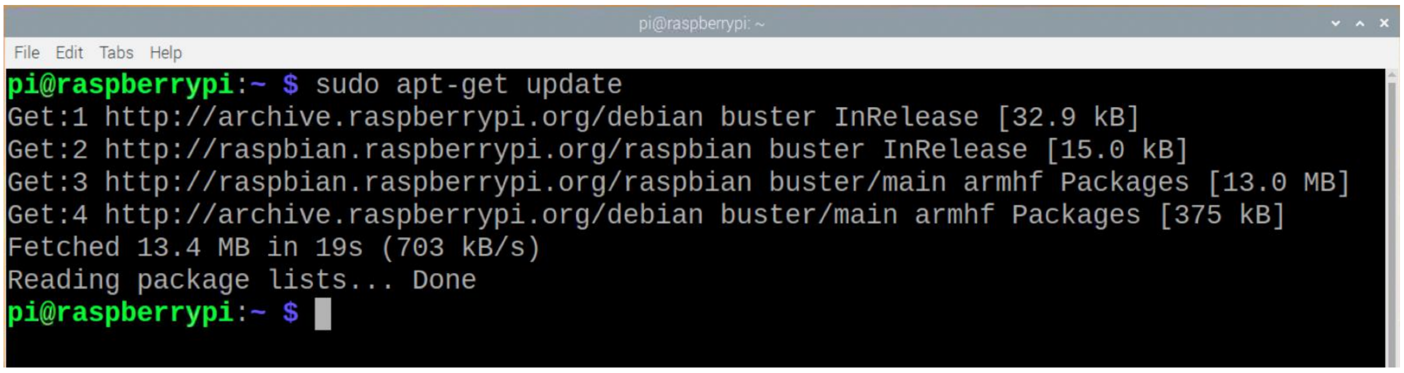
To make programming in Python3 easier we will be using an Integrated Development and Learning Environment (IDLE). There are many different development environments available for Python. Other common environments include: Thonny, PyCharm and iPython.

INSTALLING PYTHON3 IDLE USING THE TERMINAL

IN this example we will install the Python3 IDLE using the Terminal on the Raspberry Pi. When we need to install programs on the Raspberry Pi we always use the Terminal. To install software, we use an application manager called apt.

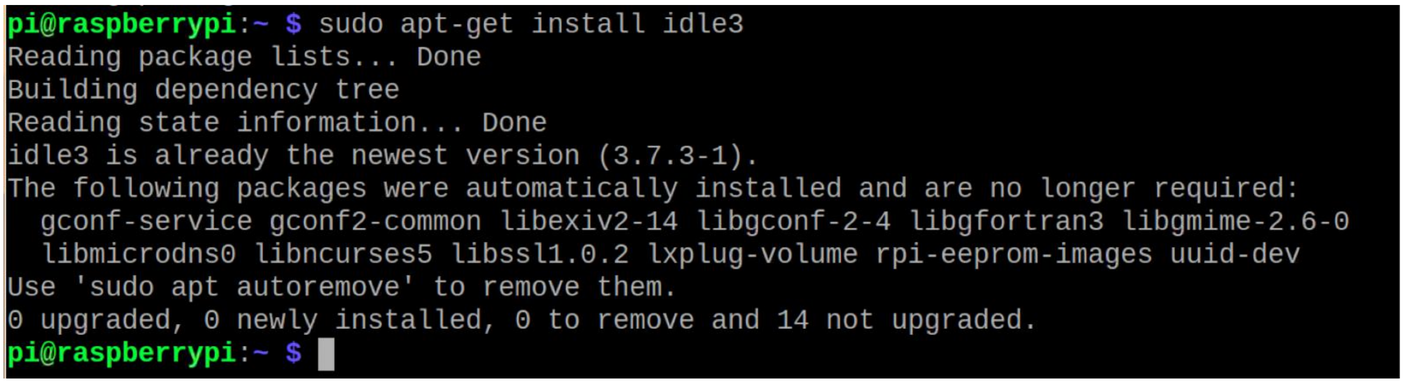
1. **Installing Python3 IDLE**

- Open the Terminal on the Raspberry Pi
- Update our existing software libraries by entering the command **sudo apt-get update**
- We always need to update our software libraries before we install new software. This because software sometimes needs other software to help make it work. The additional software is called dependencies. If we install an older version of the software, some of the dependencies may have changed and this could make our program run erratically.



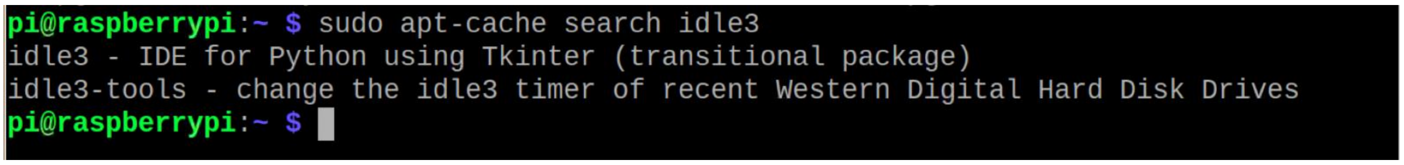
```
pi@raspberrypi:~ $ sudo apt-get update
Get:1 http://archive.raspberrypi.org/debian buster InRelease [32.9 kB]
Get:2 http://raspbian.raspberrypi.org/raspbian buster InRelease [15.0 kB]
Get:3 http://raspbian.raspberrypi.org/raspbian buster/main armhf Packages [13.0 MB]
Get:4 http://archive.raspberrypi.org/debian buster/main armhf Packages [375 kB]
Fetched 13.4 MB in 19s (703 kB/s)
Reading package lists... Done
pi@raspberrypi:~ $ 
```

- Now we can install the idle3 software by entering the command **sudo apt-get install idle3**
- If the software is already installed (as in the example below) the Terminal will tell you.



```
pi@raspberrypi:~ $ sudo apt-get install idle3
Reading package lists... Done
Building dependency tree
Reading state information... Done
idle3 is already the newest version (3.7.3-1).
The following packages were automatically installed and are no longer required:
  gconf-service gconf2-common libexiv2-14 libgconf-2-4 libgfortran3 libgmime-2.6-0
  libmicrodns0 libncurses5 libssl1.0.2 lxplug-volume rpi-eeprom-images uuid-dev
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 14 not upgraded.
pi@raspberrypi:~ $ 
```

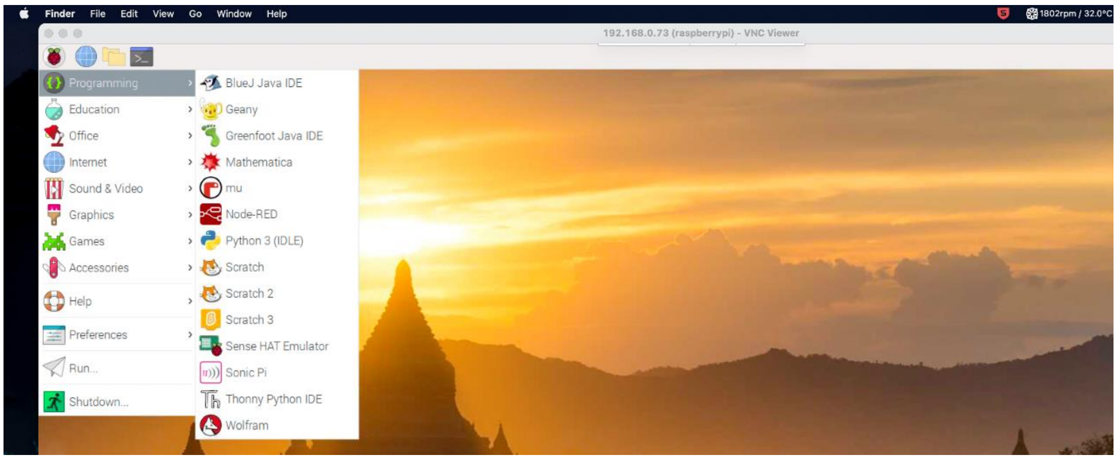
- If you don't know what software to install or if the software exists, you can search for software in the library by entering the command **sudo apt-cache search idle3**
- If you are interested in installing a game of chess you could enter the command **sudo apt-cache search chess** or use the wildcard symbol * to expand your search, **sudo apt-cache search chess***



```
pi@raspberrypi:~ $ sudo apt-cache search idle3
idle3 - IDE for Python using Tkinter (transitional package)
idle3-tools - change the idle3 timer of recent Western Digital Hard Disk Drives
pi@raspberrypi:~ $ 
```

- If the installation of Python3 IDLE has worked correctly you should see the Python3 software link appear in the main Raspberry Pi menu under **Programming**.

SMART CITIES



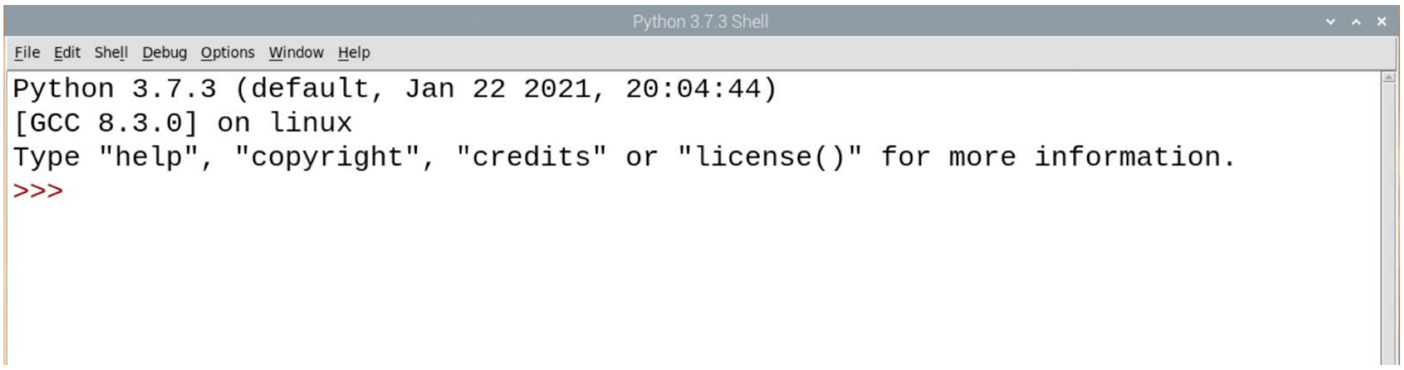
USING THE PYTHON SHELL TO ENTER CODE

The Python3 Integrated Development and Learning Environment (IDLE) is a common environment for most programming languages. Sometimes it is shortened to IDE (Integrated Development Environment). These interfaces make editing and running programs easier. Python3 IDLE is very simple and easy to use and it was the first environment installed on the Raspberry Pi for learning Python.

In this example, we will start using the Shell Window. The Shell Window allows single Python instructions (or small instruction sets) to be entered and tested. We will run through a few examples.

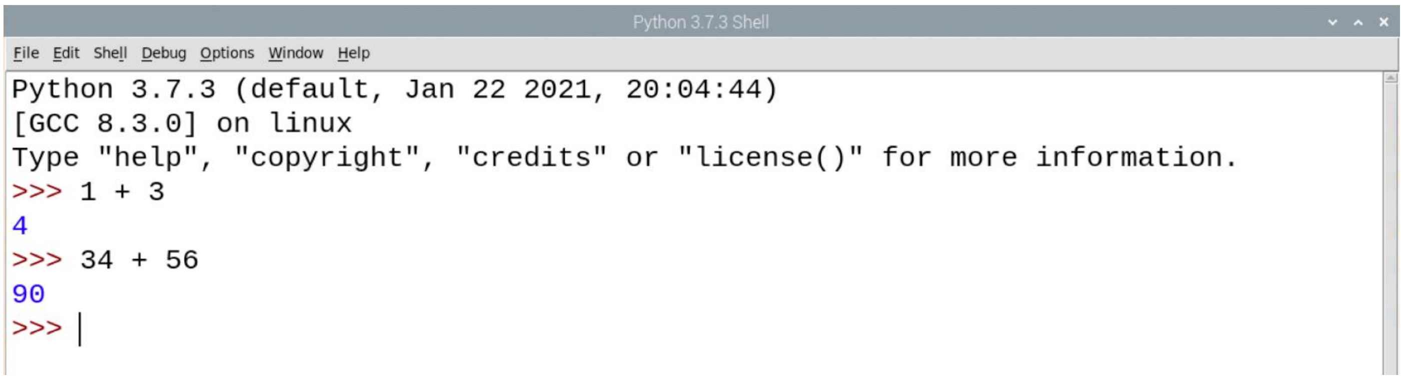
2. **Explore the Python Shell Window and the Python Language**

- Go to the Python3 IDLE icon in the Raspberry Pi > Programming menu and open IDLE3.
- The program will open to the Shell Window.



```
Python 3.7.3 (default, Jan 22 2021, 20:04:44)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
```

- We can now start to enter python language instructions directly in the Shell Window.
- Let's start with some simple arithmetic instructions. Type in **1 + 3** and press **Enter**. Try some other operations.



```
Python 3.7.3 (default, Jan 22 2021, 20:04:44)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>> 1 + 3
4
>>> 34 + 56
90
>>> |
```

- More complex maths will require some special instructions and the import of additional libraries. We will cover this in future lessons.

3. Printing Words, Text and Strings

- In addition to numbers, we can also print out words and text.
- To make python print out text you enter the command **print("Hello")**
- In the older days when there were no computer monitors, all output from computers went to printers. This legacy continues today, and when we want to display the output from a computer, we use the **print** statement.
- To tell a computer that we are working with text we need to include the text in quotation marks **"Hello".**
- Another way we refer to words or text in computer languages, is to call them **Strings**. Computers manipulate the characters in a word or sentence like beads threaded on a string. Hence the term String.

```
>>> print("Hello - my first print statement")
Hello - my first print statement
>>>
```

- ○ Comments or addition information that does not require students to do anything. E.g. "This may take several moments to load

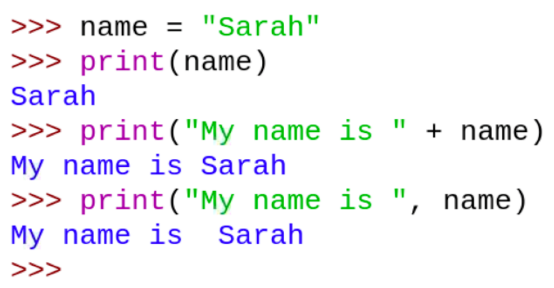
4. Working with Variables

- Computer programming languages can also manipulate variables.
- Variables are a bit like the variables we use in algebra.
- Variables can represent numbers, text and other entities.
- Let's start with some simple numerical variables.
- Enter **a = 7** and **b = 3**
- This will **assign** the number 7 to the variable name **a**.
- It will also **assign** the number 3 to the variable name **b**.
- Finally, just like in algebra, we can add two variables together. In this case **a + b** equals 10.

```
>>> a = 7
>>> b = 3
>>> a + b
10
>>>
```

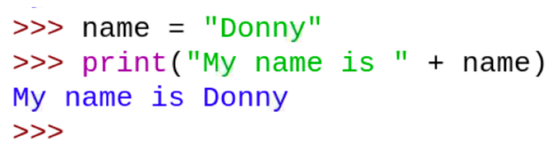
- We can also use variables to hold text or Strings.

- Enter **name = "Sarah"**
- To print out the value held in the variable name we use the print statement.
- Enter **print(name)** – this will print out the value currently held in the variable called name.
- We can also add some additional text within the print statement to make the output more readable.
- Two variations are shown.
    - **print("My name is " + name).** This statement adds two Strings together using the plus "**+**" symbol
    - **print("My name is ", name)** . This statement uses a special feature of the print statement. Strings and variables are separated by a **comma**.



```
>>> name = "Sarah"
>>> print(name)
Sarah
>>> print("My name is " + name)
My name is Sarah
>>> print("My name is ", name)
My name is  Sarah
>>>
```

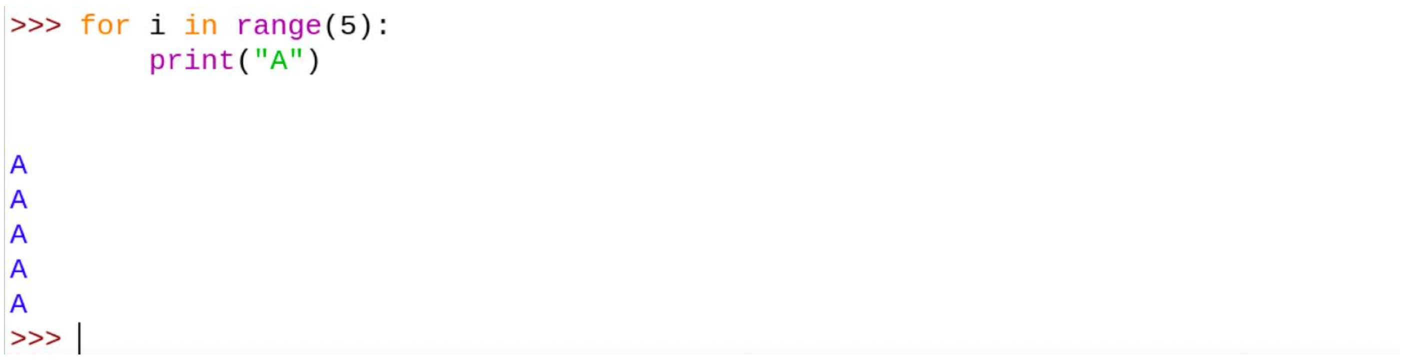
- One of the reasons we use variable to hold values is because these values may change.
- If we enter **name = "Donny"** and then enter **print(name)** we will see that "Sarah" has been replaced with "Donny"



```
>>> name = "Donny"
>>> print("My name is " + name)
My name is Donny
>>>
```

5. For loops in Python

- Sometimes you need to repeat a certain operation a specific number of times within a program.
- To do this we use loops.
- One such loop operation is called a **For Loop**.
- In this example we will create a **For Loop** that will repeat itself **five times**.
- Enter the following code and observe the result.

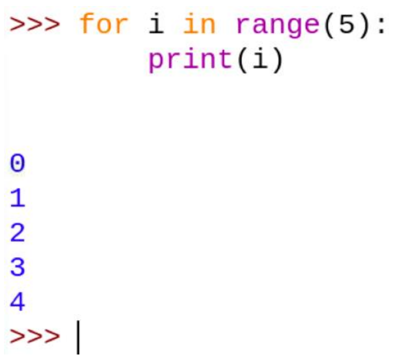


```
>>> for i in range(5):
        print("A")


A
A
A
A
A
>>> 
```

- This **For Loop** will repeat 5 times because the **range** is set to **5**.
- We can tell what code is within the loop structure because the code is indented 4 spaces.
- The only code within the loop is the **print("A")** statement.

6. For loop with variables

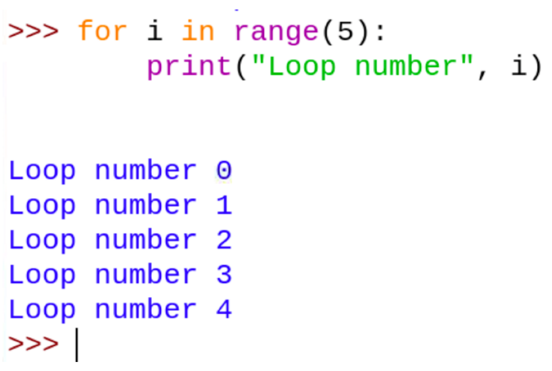
- When we declare a **for loop** we always include a variable.
- In the **for loop** example above the variable name was **i**
- The **variable i** gives us a running count of the number of times we have looped.
- Enter the following example:



```
>>> for i in range(5):
        print(i)


0
1
2
3
4
>>> 
```

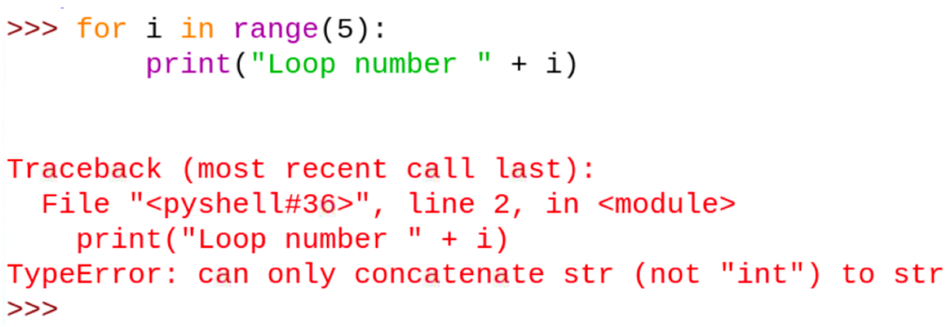
- In this example we are printing the value within the variable **i** with each loop iteration.
- You can see that the count starts at **zero** and continues up to the number **four**.
- This is because computers always start **counting from zero**, not 1.

- We can make the loops more human readable with the following print statement.



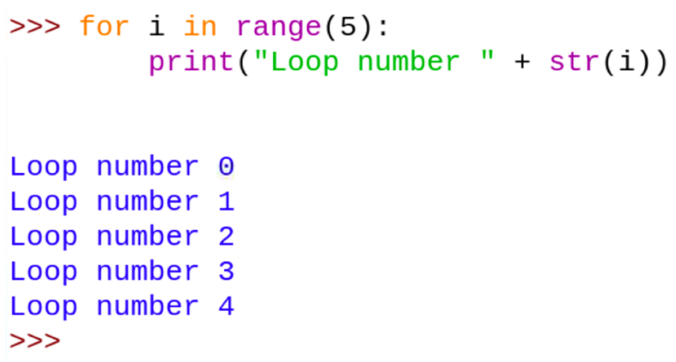
```
>>> for i in range(5):
        print("Loop number", i)


Loop number 0
Loop number 1
Loop number 2
Loop number 3
Loop number 4
>>> |
```

- In this example we are using a special print statement
    - text or **String** in quotation marks
    - **comma**
    - numerical values contained in variable **i**
- If we try to print both a String and a numerical value together we get an error.



```
>>> for i in range(5):
        print("Loop number " + i)


Traceback (most recent call last):
  File "<pyshell#36>", line 2, in <module>
    print("Loop number " + i)
TypeError: can only concatenate str (not "int") to str
>>>
```

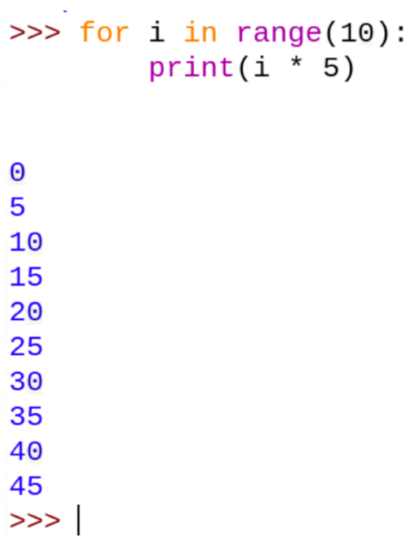
- To get around this problem we need to use an inbuilt function within python called **str()**
- The **str()** function will convert a number (or integer) into a String. Then we can add "**+**" two Strings within the one print statement.
- Enter the following example code. By putting the variable **i** inside the **str(i)** function, we convert it to a String.



```
>>> for i in range(5):
        print("Loop number " + str(i))


Loop number 0
Loop number 1
Loop number 2
Loop number 3
Loop number 4
>>>
```

7. Times Tables with For Loops

- We can do simple times tables examples using for loops.
- Let's try an example using the five times tables.
- Enter the following code.
- Note that with computer languages the mathematical symbol for times (**x**) is actually an Asterix (*****).



```
>>> for i in range(10):
        print(i * 5)


0
5
10
15
20
25
30
35
40
45
>>> |
```

- Notice that in this example we do have the makings of the 5 times tables, except that we normally start counting from 1, not zero.
- To fix this we can nominate that the starting value of the count should be 1.
- Unfortunately, we also need to state the ending value. The ending value needs to be final number + 1. So, if we need to end at the number 10, we need to enter 11.

- The final range will be **range(1, 11)** which will deliver integer values from **1 to 10** as outputs.

- See the following example for clarity.

```
>>> for i in range(1, 11):
        print(i * 5)


5
10
15
20
25
30
35
40
45
50
>>>
```

- To make the output more human readable we are going to have to convert some of the numerical outputs to Strings so that we can add them together (also called **concatenate**) to make a larger String.

- Enter the following code to display the Five Times tables in a readable form.

```
>>> for i in range(1, 11):
        print(str(i) + " times 5 = " + str(i * 5))


1 times 5 = 5
2 times 5 = 10
3 times 5 = 15
4 times 5 = 20
5 times 5 = 25
6 times 5 = 30
7 times 5 = 35
8 times 5 = 40
9 times 5 = 45
10 times 5 = 50
>>> |
```

- In this example we had to do the following operations in the print statement:

    - **str(i)** – convert the integer in the variable **i** to a String using the **str()** function

    - **str(i * 5)** – convert the result of **i * 5** to a String using the **str()** function

- Add all the Strings together (concatenate) using the "**+**" symbol

- Try to do another example from the Times Tables.
    - Extend the Times Tables from 1 to 12
    - Add a variable name for an item you are multiplying, eg. thing = "eggs"
    - Answers below!

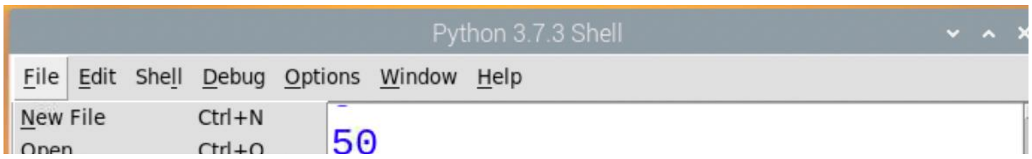
```
>>> thing = "eggs"
>>> for i in range(1,13):
        print(str(i) + " " + thing + " times 5 equals " + str(i * 5) + " eggs")


1 eggs times 5 equals 5 eggs
2 eggs times 5 equals 10 eggs
3 eggs times 5 equals 15 eggs
4 eggs times 5 equals 20 eggs
5 eggs times 5 equals 25 eggs
6 eggs times 5 equals 30 eggs
7 eggs times 5 equals 35 eggs
8 eggs times 5 equals 40 eggs
9 eggs times 5 equals 45 eggs
10 eggs times 5 equals 50 eggs
11 eggs times 5 equals 55 eggs
12 eggs times 5 equals 60 eggs
>>>
```

USING THE EDITOR WINDOW IN PYTHON3 IDLE

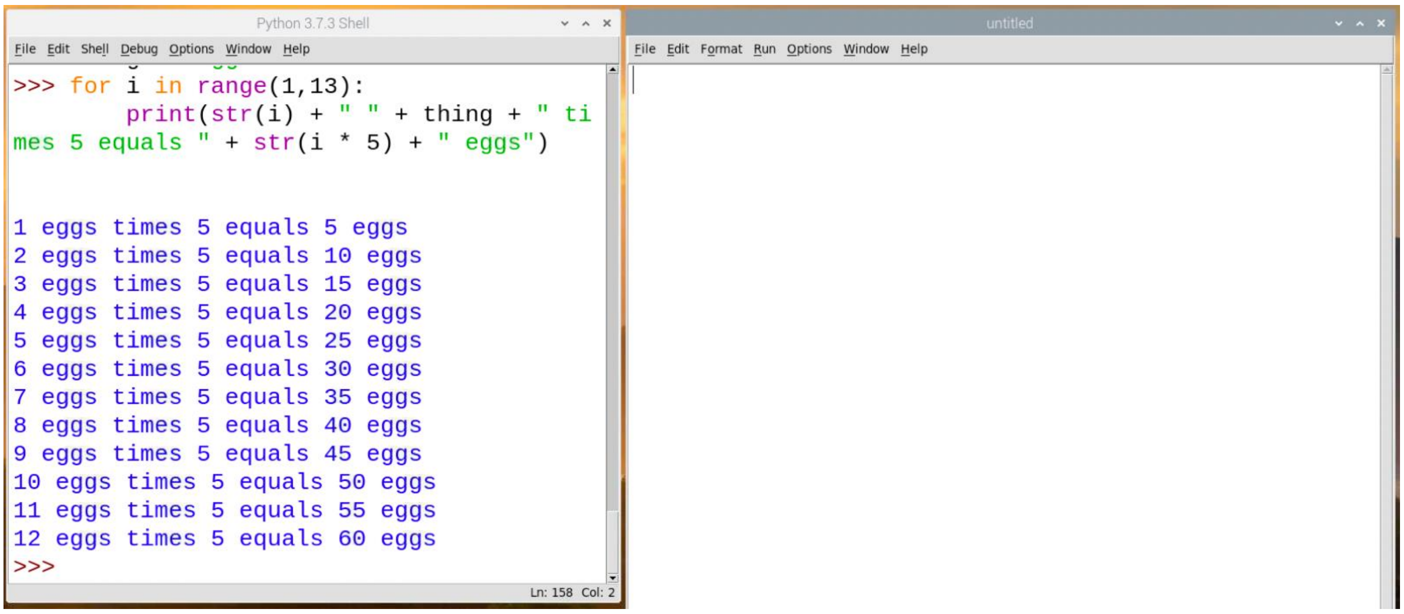
Up to now we have only been using the Python3 IDLE **Shell Window**. This allows us to enter simple statements and execute them. To write longer programs we need to use the **Editor Window**. The Editor Window allows us to write long programs, edit the programs more easily and save our programs.

8. Opening an Editor Window

- To open an Editor Window, click on the **File Menu** in the Python Shell and select **New File**.



- This will open an untitled document which is the Editor Window.
- To make things easier to see you can resize the Shell and Editor Windows so that they sit side by side.


Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
untitled
File Edit Format Run Options Window Help
Ln: 158 Col: 2

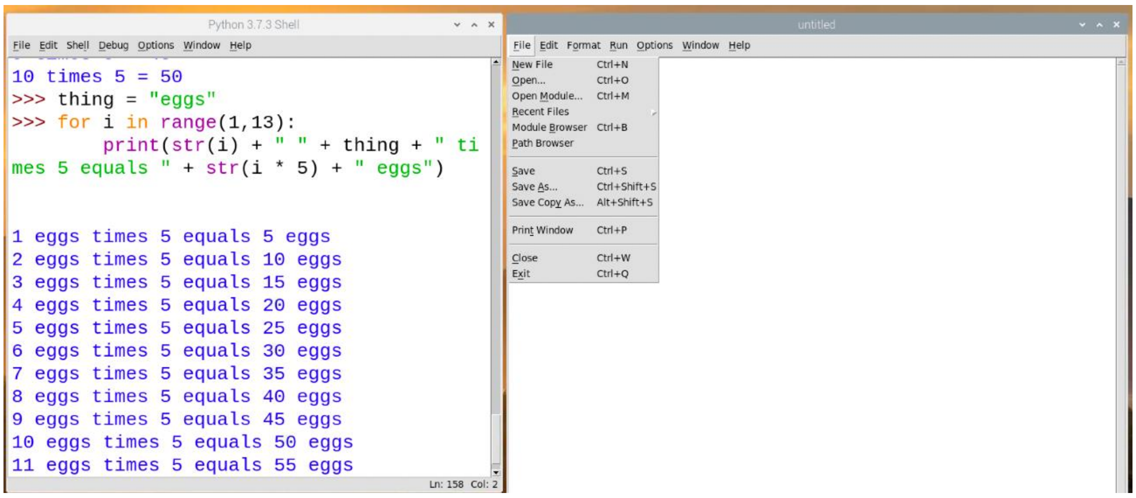
```
>>> for i in range(1,13):
        print(str(i) + " " + thing + " ti
mes 5 equals " + str(i * 5) + " eggs")


1 eggs times 5 equals 5 eggs
2 eggs times 5 equals 10 eggs
3 eggs times 5 equals 15 eggs
4 eggs times 5 equals 20 eggs
5 eggs times 5 equals 25 eggs
6 eggs times 5 equals 30 eggs
7 eggs times 5 equals 35 eggs
8 eggs times 5 equals 40 eggs
9 eggs times 5 equals 45 eggs
10 eggs times 5 equals 50 eggs
11 eggs times 5 equals 55 eggs
12 eggs times 5 equals 60 eggs
>>>
```

- When the Editor Window opens the default file is untitled. The computer is really prompting us to save our file and give it a name.

9. Saving a File in the Editor Window

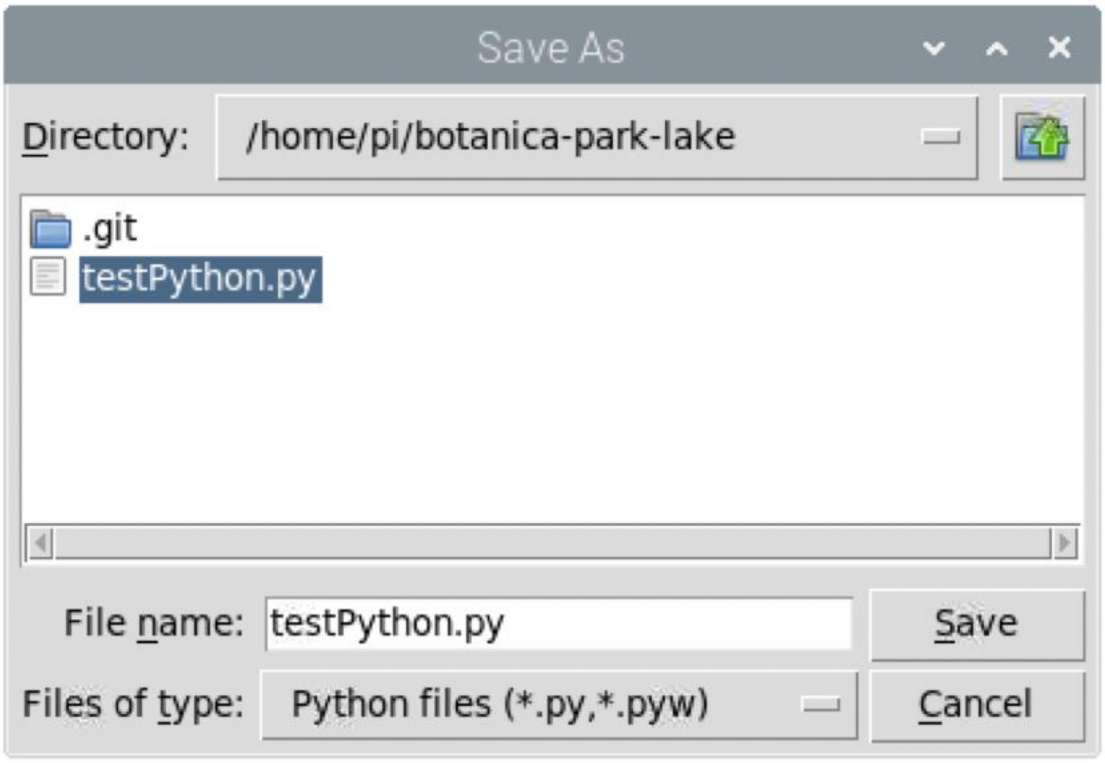
- To save a file in the Editor Window, select the File Menu and select Save As.

- Choose a suitable directory and file name for your new file.



```
10 times 5 = 50
>>> thing = "eggs"
>>> for i in range(1,13):
        print(str(i) + " " + thing + " times 5 equals " + str(i * 5) + " eggs")


1 eggs times 5 equals 5 eggs
2 eggs times 5 equals 10 eggs
3 eggs times 5 equals 15 eggs
4 eggs times 5 equals 20 eggs
5 eggs times 5 equals 25 eggs
6 eggs times 5 equals 30 eggs
7 eggs times 5 equals 35 eggs
8 eggs times 5 equals 40 eggs
9 eggs times 5 equals 45 eggs
10 eggs times 5 equals 50 eggs
11 eggs times 5 equals 55 eggs
```

- In my case, I've saved the file in the botanica-park-lake directory, and the file name is **testPython.py**
- You don't need to add the **.py file extension**. It will be added automatically.
- A file extension gives the computer some information about the application needed to run the file.



Save As

Directory: /home/pi/botanica-park-lake

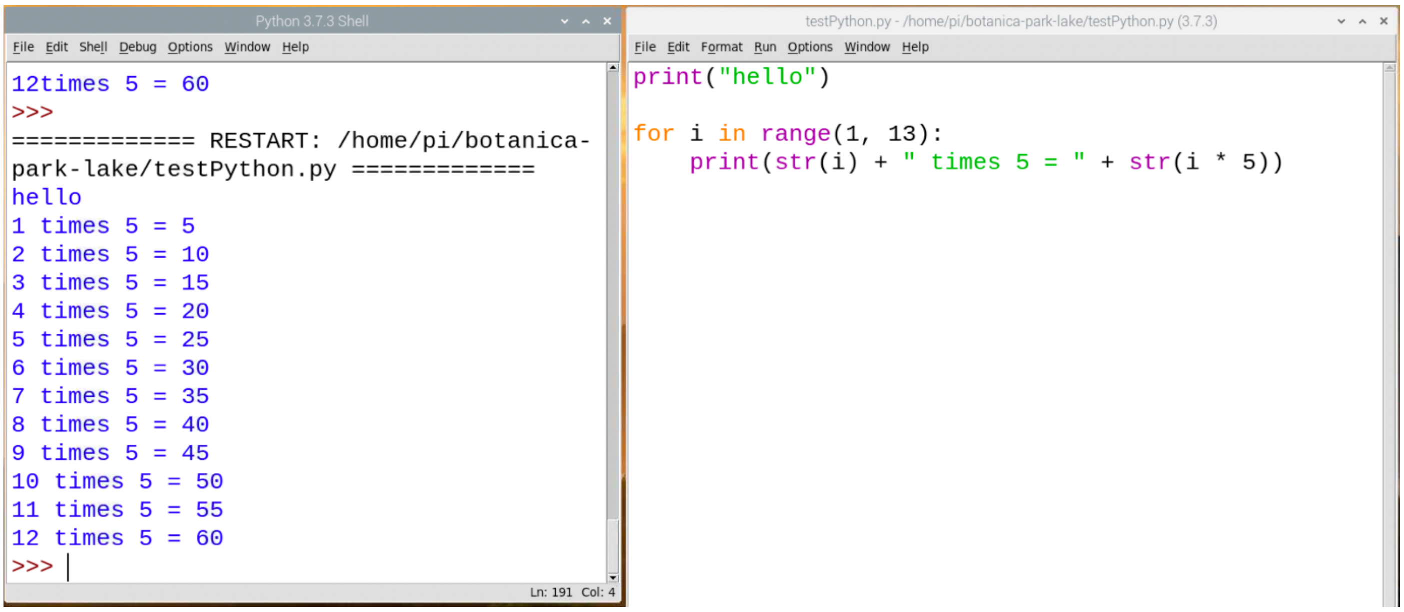
.git
testPython.py

File name: testPython.py

Files of type: Python files (*.py,*.pyw)

10. Running a File from the Editor Window

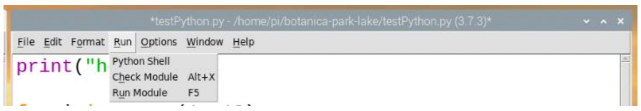
- Let’s write a small program in the Editor Window and then **Run** it.
- Enter the following code:


Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
Ln: 191 Col: 4
testPython.py - /home/pi/botanica-park-lake/testPython.py (3.7.3)
File Edit Format Run Options Window Help

```
12times 5 = 60
>>>
============= RESTART: /home/pi/botanica-park-lake/testPython.py =============
hello
1 times 5 = 5
2 times 5 = 10
3 times 5 = 15
4 times 5 = 20
5 times 5 = 25
6 times 5 = 30
7 times 5 = 35
8 times 5 = 40
9 times 5 = 45
10 times 5 = 50
11 times 5 = 55
12 times 5 = 60
>>>
```

```python
print("hello")

for i in range(1, 13):
    print(str(i) + " times 5 = " + str(i * 5))
```

- Save this code using the **File** Dropdown Menu in the Editor Window and select **Save**.
- To Run the file use the **Run** Dropdown and select **Run Module**.
- You always need to Save your program before running it.
- The output of your program will be director to the **Shell Window**.


testPython.py - /home/pi/botanica-park-lake/testPython.py (3.7.3)
File Edit Format Run Options Window Help
Python Shell
Check Module Alt+X
Run Module F5

```
print("h
```

- Congratulations. Now you can now write, edit and run more complex Python programs.