# Exercise Manual
for
CE/CZ1003
Introduction to Computational Thinking

Exercise #1:
Introduction to Raspberry Pi

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
NANYANG TECHNOLOGICAL UNIVERSITY

Ex. #1 - Introduction to RPi

Learning Objectives

In this course, students will learn the various concepts of computational thinking through programming exercises performed on the Raspberry Pi platform. This laboratory exercise is to introduce students to the Raspberry Pi (RPi) single-board computer, and how to use its text-based (Linux's bash) commands to explore the environment of the RPi.

Intended Learning Outcomes

At the end of this exercise, you should be able to

- Operate the Raspberry Pi board through its Graphic User Interface and through its Terminal mode interface.
- Write basic programs in Python language and C language to observe the difference in running an interpreted language and a compiled language based programs.

Equipment and accessories required

i) Raspberry Pi 3 Model B (RPi3) board with Sense HAT add-on display module/board.
ii) HDMI monitor, USB keyboard and USB mouse.
iii) A USB power source to power the RPi3 board (E.g. Power Bank, Adaptor or USB port of a desktop computer).

---

1. Introduction

The Raspberry Pi (RPi) is a small single-board computer developed in UK by the Raspberry Pi Foundation to promote the teaching of basic computer science in schools. It is a low cost computer board (about S$50) but comes with powerful processor (that are also used in some of the latest smartphones).

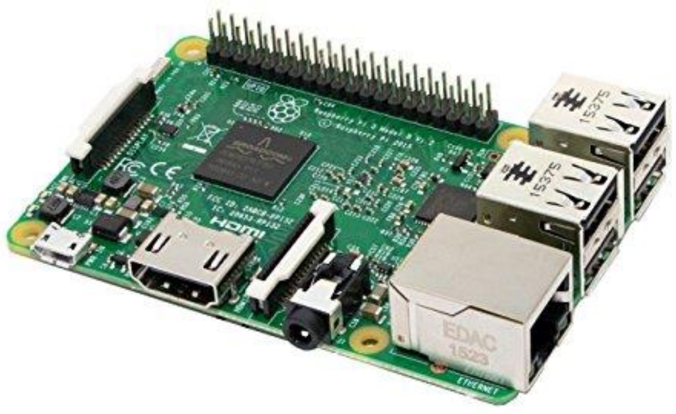


Fig.1 - Raspberry Pi 3 Model B (RPi3)

For the RPi3 model that is used in this exercise, the board also support two wireless interfaces (i.e. connections), Wifi and Bluetooth, which means that it can also be accessed using your wireless devices such as Notebook and Smartphone.

As an educational kit, you can add other devices to the RPi board, typically in the form of add-on modules. In this course, you will be using the **Sense Hat** add-on module on the RPi board, such that you can code programs to display messages and images on the module, as will be seen later.

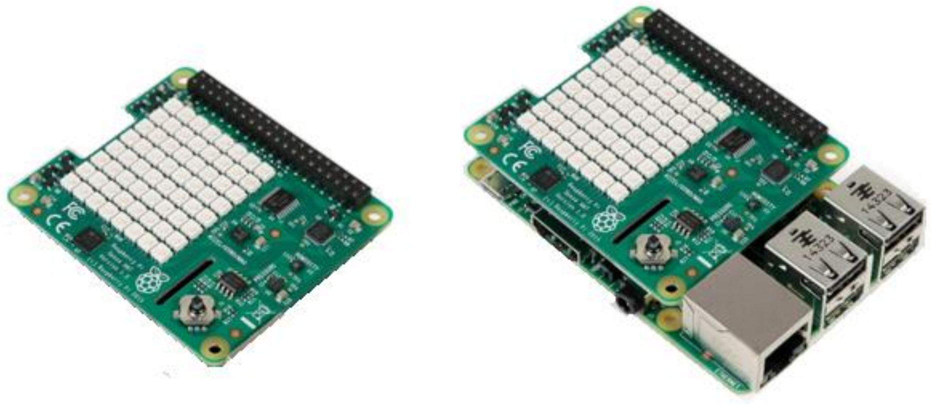
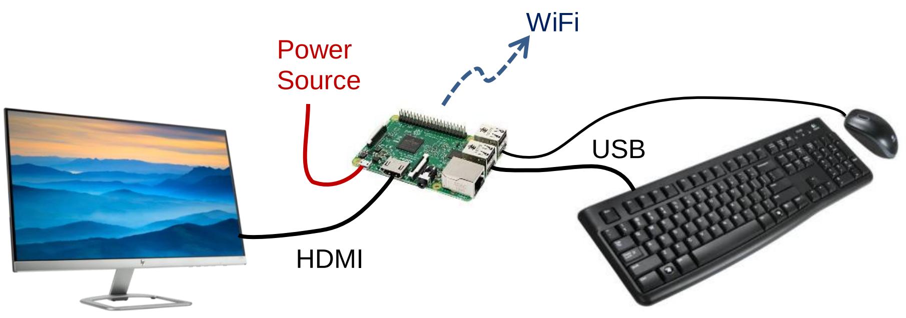


Fig. 2 - Sense Hat module and its interface on RPi3

The RPi can be set up as a standalone computer by connecting a monitor, keyboard and mouse through its various interfaces.



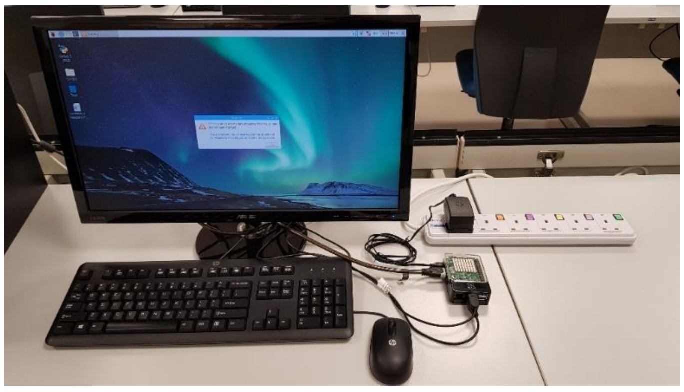


Fig. 3 - Standalone setup of RPi3

The operating system (OS) used on the RPi is the **Raspbian OS**, which is based on the open source Linux OS. (An OS is the software used to control and operate a computer, such as the Microsoft's Windows OS and Apple's macOS for their notebooks, and Google's Android and Apple's iOS for their smartphones)

2 Familiarization of the RPi's User Interface and useful commands

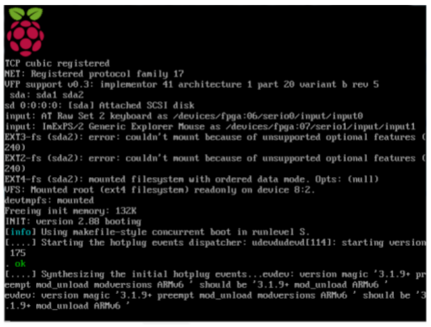
You will now go through a sequence of tasks designed to get you familiarized with the operation of the RPi as a standalone computer, in particular, certain commands that are frequently used on RPi.

Remark : Wiring connection in step 2.1 and 2.2 (Figure 3) may already connected

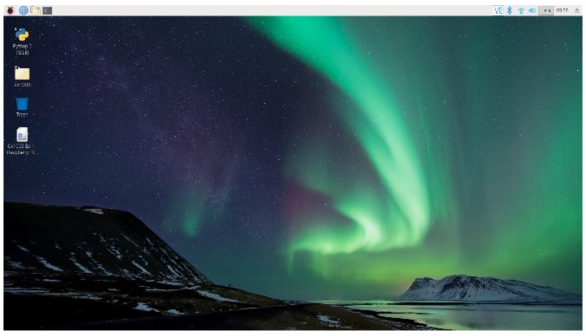
2.1 Connect the RPi3 board with the keyboard, mouse and monitor as shown in Figure 3.

2.2 Apply the power source to the RPi board (which in this case is by connecting the power adaptor USB cable to Rpi and ***power on the adaptor***)

- You will see a series of messages appearing on the monitor, which eventually leads to a graphic based display screen – the Desktop screen.

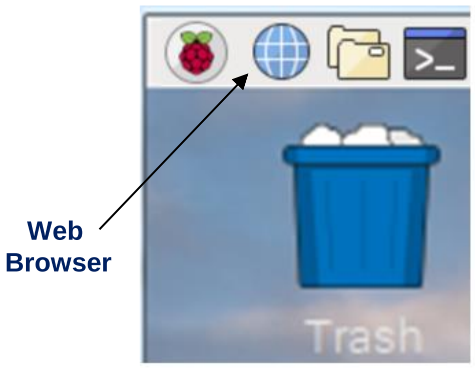


```
TCP cubic registered
NET: Registered protocol family 17
VFP support v0.3: implementor 41 architecture 1 part 20 variant b rev 5
 sda: sda1 sda2
sd 0:0:0:0: [sda] Attached SCSI disk
input: AT Raw Set 2 keyboard as /devices/fpga:06/serio0/input/input0
input: ImExPS/2 Generic Explorer Mouse as /devices/fpga:07/serio1/input/input1
EXT3-fs (sda2): error: couldn't mount because of unsupported optional features (
240)
EXT2-fs (sda2): error: couldn't mount because of unsupported optional features (
240)
EXT4-fs (sda2): mounted filesystem with ordered data mode. Opts: (null)
VFS: Mounted root (ext4 filesystem) readonly on device 8:2.
devtmpfs: mounted
Freeing init memory: 132K
INIT: version 2.88 booting
[info] Using makefile-style concurrent boot in runlevel S.
[....] Starting the hotplug events dispatcher: udevdudevd[114]: starting version
 175
. ok
[....] Synthesizing the initial hotplug events...evdev: version magic '3.1.9+ pr
eempt mod_unload modversions ARMv6 ' should be '3.1.9+ mod_unload ARMv6 '
evdev: version magic '3.1.9+ preempt mod_unload modversions ARMv6 ' should be '3
.1.9+ mod_unload ARMv6 '
```



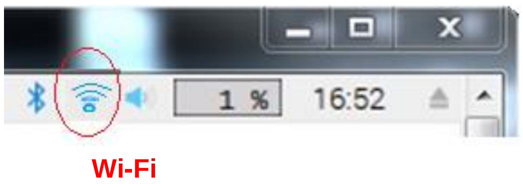
2.3 This is the Graphic User Interface (GUI) based Desktop provided by Raspbian Operating System. There are several 'icons' available on the Desktop that allow you to interact with the RPi3 system.

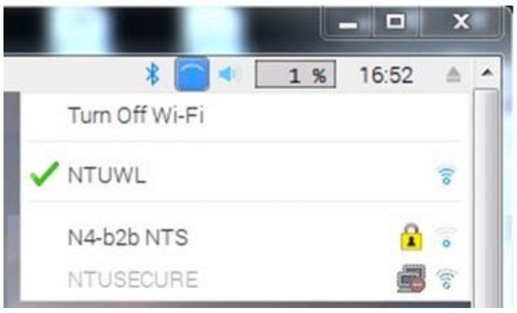
- The icons on the top left corner of the Desktop screen can be used to access the various functions available on RPi, including a Web browser program that you are going to use in a while.



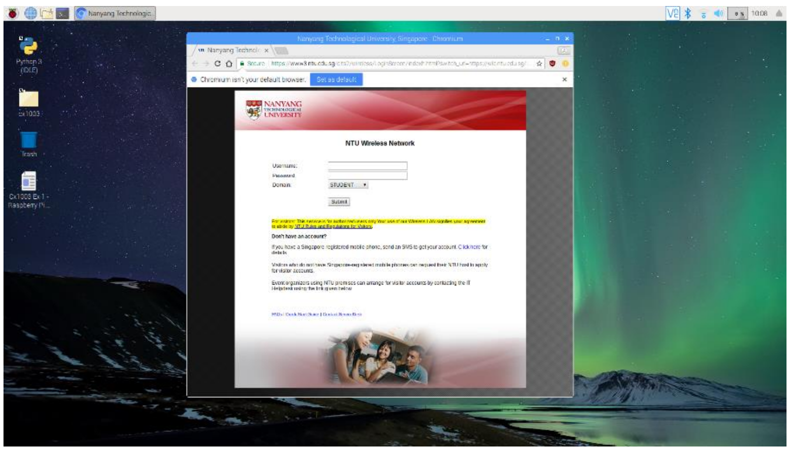
2.4 To start with, you are going to connect the RPi to NTU wireless network, and then log in to the NTULearn's website to access the various relevant documents needed for doing the Cx1003 hands-on exercises – including this exercise manual.

- On the top right corner of the Desktop, click the Wi-Fi icon, and there should be the 'NTUWL' connection that you can select.





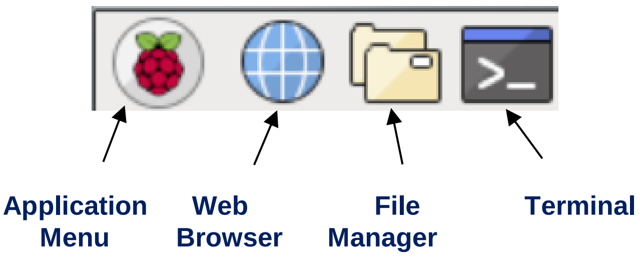
- Next, launch the web browser program at the top left corner of the Desktop. Enter 'www.ntu.edu.sg' in the URL address bar, and the NTU network login interface should appear.



- Enter your NTU account's detail, and if everything is OK, the RPi board should now be connected to the NTU's network.

- You can now log on to the NTULearn website, 'ntulearn.ntu.edu.sg' to retrieve this exercise manual, and continue with the rest of the hands-on exercises.

2.5 Continue with the other icons shown on the top left corner of the Desktop screen, these icons can be used to access the various functions available on RPi.

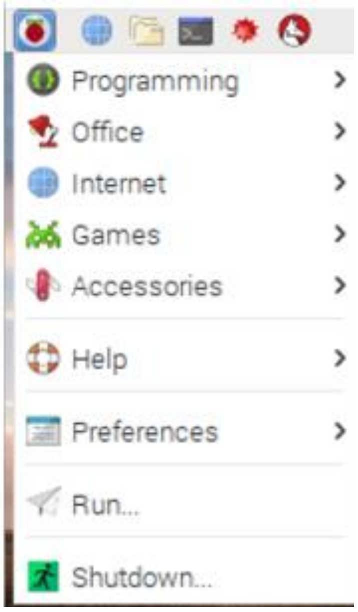


- Click on the **Application Menu** icon - you will be provided with a list of options - programs and functions available on RPi, which you can use to operate the RPi. Note that there is a **Shutdown** function which is used to properly turn off the RPi at the end of the exercise.

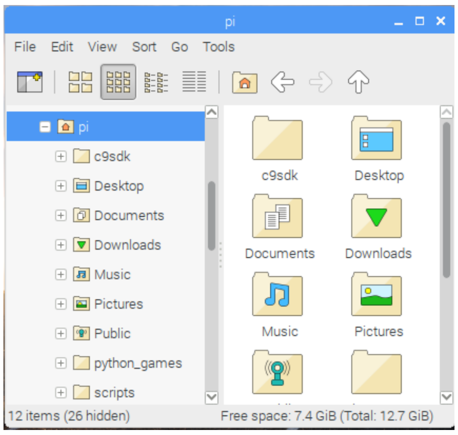
- The **Web Browser** is normally used for surfing internet, which you will use as needed during the various hands-on exercises in this course.

- **File Manager** is used to explore the directories and files stores on the RPi.

- **Terminal** is a program that is usually used to enter text commands and run other programs, as will be seen later.



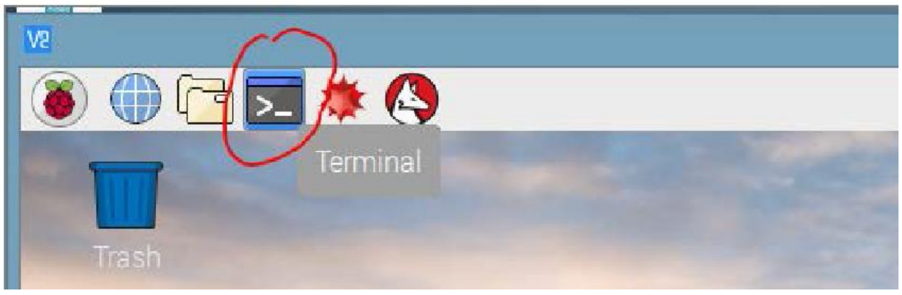
2.6 Click on the **File Manager** icon. You will be presented with a window showing the various folders/directories and files available on the RPi.
- You can further explore the various folders by clicking on each of them to see their contents.
- Click the **'Desktop'** folder and explore its content.



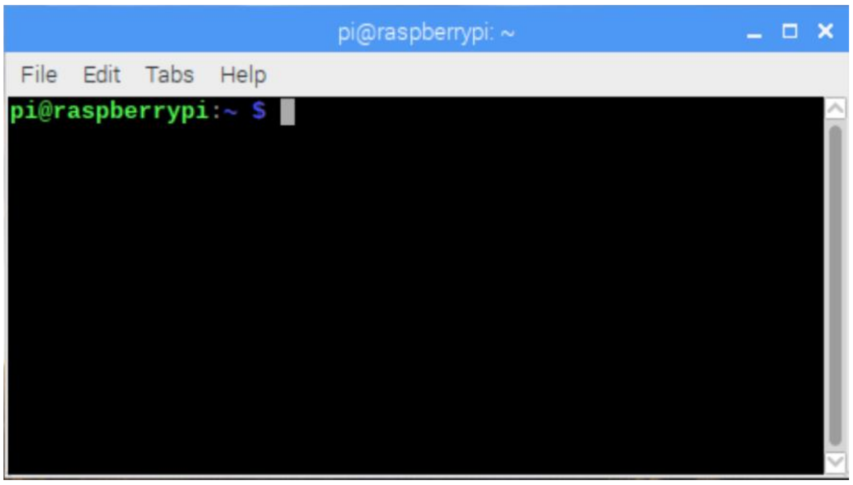
3. Finding information using commands in Terminal mode

While the GUI based Desktop is convenient and user friendly, there are other things that you can do with RPi, many of which can only be accessed by typing text-based commands through the Terminal. In practice, using Terminal mode is the most 'powerful', flexible and quickest way to accomplish many things on RPi.

3.1 Click on the **Terminal** icon to open a terminal window.



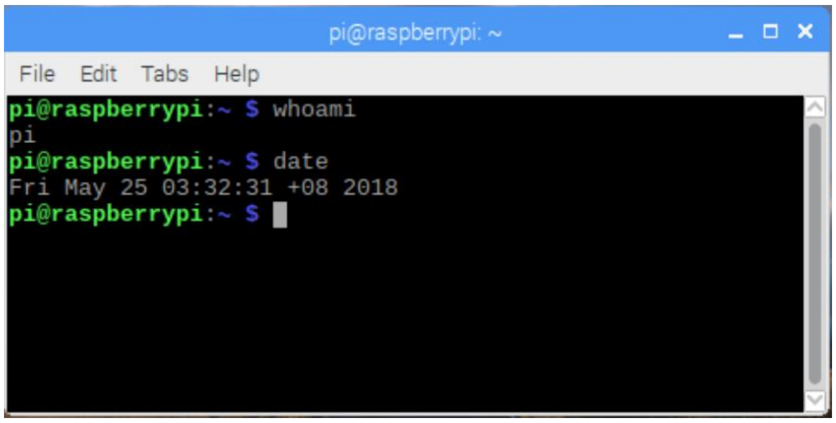
- Resize the terminal window so it's of a reasonable size for you to type into.



```
pi@raspberrypi:~ $
```

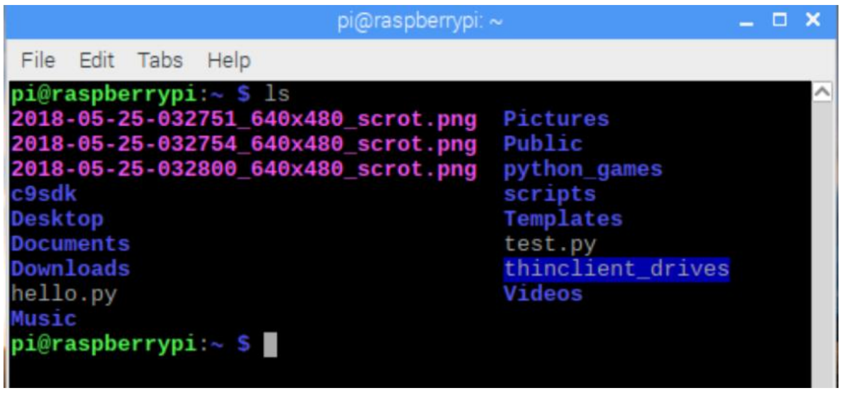
3.2 Type the following commands into the terminal and observe what is shown for each of them.

- **whoami**
- **date**



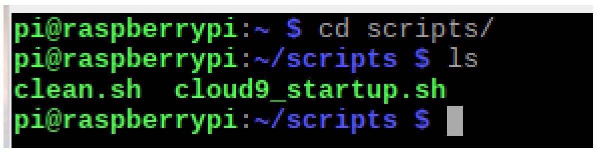
```
pi@raspberrypi:~ $ whoami
pi
pi@raspberrypi:~ $ date
Fri May 25 03:32:31 +08 2018
pi@raspberrypi:~ $
```

3.3 You can view the contents in the current directory by using the list command **ls**. This will show the various files and sub-directories available in the current directory. (Try using "**ls -a**" and "**ls -l**" for different listing output)



```
pi@raspberrypi:~ $ ls
2018-05-25-032751_640x480_scrot.png  Pictures
2018-05-25-032754_640x480_scrot.png  Public
2018-05-25-032800_640x480_scrot.png  python_games
c9sdk                                scripts
Desktop                              Templates
Documents                            test.py
Downloads                            thinclient_drives
hello.py                             Videos
Music
pi@raspberrypi:~ $
```

3.4 You can change to a different directory, such as to the `scripts` directory using the command: `cd scripts`

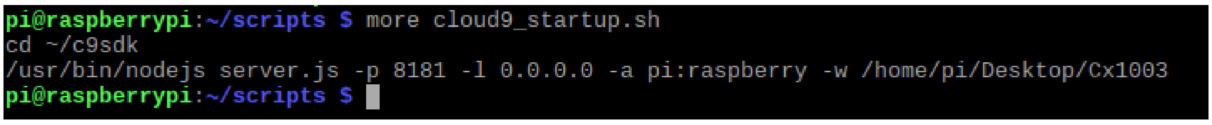


```
pi@raspberrypi:~ $ cd scripts/
pi@raspberrypi:~/scripts $ ls
clean.sh  cloud9_startup.sh
pi@raspberrypi:~/scripts $
```

- View the contents of this directory (using the `ls` command again).

3.5 You can view the content of the file by using the command `more`

- Type the command `more cloud9_startup.sh`
- The content of this file is then shown on the screen.



```
pi@raspberrypi:~/scripts $ more cloud9_startup.sh
cd ~/c9sdk
/usr/bin/nodejs server.js -p 8181 -l 0.0.0.0 -a pi:raspberry -w /home/pi/Desktop/Cx1003
pi@raspberrypi:~/scripts $
```

Helpful tip: The <Tab> key is very useful when typing command in the terminal mode, such as to select a file in a directory without entering the full name of the file. E.g., you can execute the full command by merely typing `more clo`<Tab>. Try it.

3.6 You can check what directory you are currently working in

- Type the command `pwd`

3.7 You can change to the directory above the current directory as follows

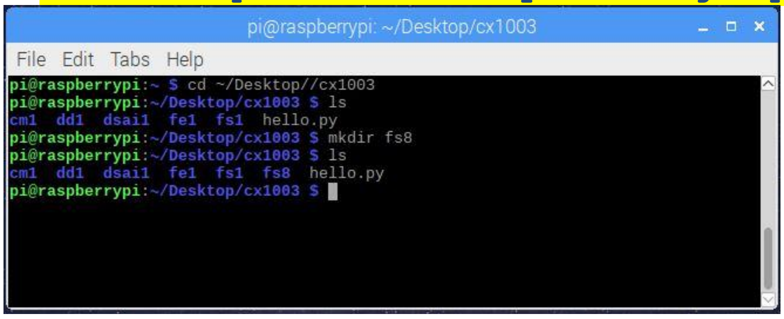
- Type the command `cd ..`

There are many other commands and things that you can do in terminal mode, which you will learn in due course, but we will now move on to do a bit of programming exercises on the RPi board.

3.8 Before you move on, **create your own lab group folder first**, as the Raspberry Pi is shared, it would be neater to keep all your program files in your own group folder.

- **Open a Terminal window**, and change directory to Desktop/cx1003 using
  `cd ~/Desktop/cx1003`
- Check to see if your lab group folder already exist using
  `ls`
- If your lab group folder does not exist, create it by
  `mkdir +++`

`***** Remark: replace +++ with your lab group number`



```
pi@raspberrypi:~ $ cd ~/Desktop//cx1003
pi@raspberrypi:~/Desktop/cx1003 $ ls
cm1  dd1  dsai1  fe1  fs1  hello.py
pi@raspberrypi:~/Desktop/cx1003 $ mkdir fs8
pi@raspberrypi:~/Desktop/cx1003 $ ls
cm1  dd1  dsai1  fe1  fs1  fs8  hello.py
pi@raspberrypi:~/Desktop/cx1003 $
```

4. Interpreted language versus Compiled language

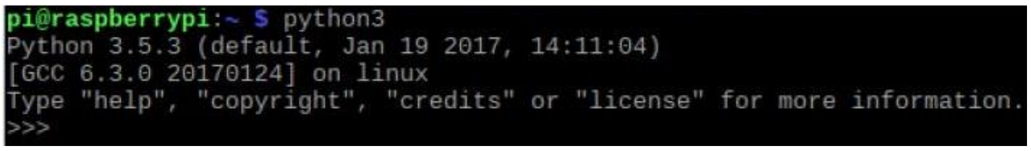
You have learnt in Topic 1 of the recorded video about the different type of programming languages, which can be further separated into interpreted language, such as Python and compiled language such as C. In this exercise you are going to write a simple program using each of these two languages and observe how each of them get executed.

4.1 Python is an ‘interpreted language’. You write the code and run the program. There are two ways to run a python code, the Script mode and the Interactive mode. You will use the Interactive mode here which is clearer for illustrating the working of an interpreter.

- Open a terminal window in RPi and launch the Python **interpreter, python3** by entering

```
python3
```

- You will see the prompt **>>>** display on screen

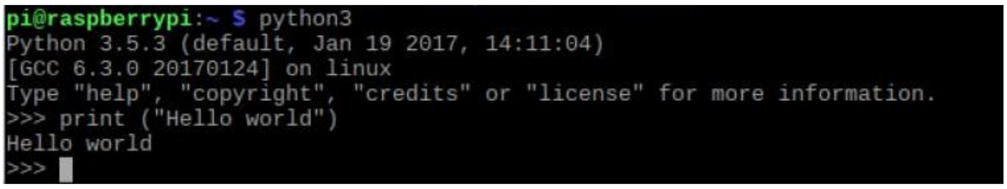


```
pi@raspberrypi:~ $ python3
Python 3.5.3 (default, Jan 19 2017, 14:11:04)
[GCC 6.3.0 20170124] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

- Key in the following Python code, and press *Enter*

```
print (“Hello world”)
```

You will see that the message is shown on the screen immediately in the next line.



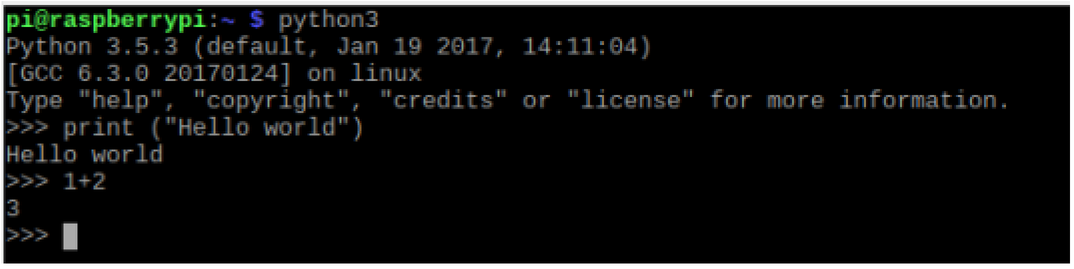
```
pi@raspberrypi:~ $ python3
Python 3.5.3 (default, Jan 19 2017, 14:11:04)
[GCC 6.3.0 20170124] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print ("Hello world")
Hello world
>>>
```

And that is – the code that you entered has been executed by the interpreter python3.

- Next key in the following and press *Enter*

```
1+2
```

You will notice that the code is immediately executed by the interpreter as soon as you press the *Enter* key.



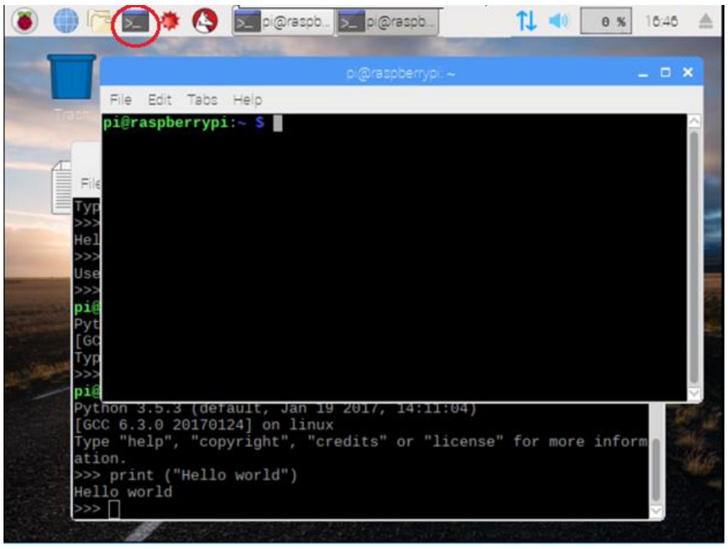
```
pi@raspberrypi:~ $ python3
Python 3.5.3 (default, Jan 19 2017, 14:11:04)
[GCC 6.3.0 20170124] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print ("Hello world")
Hello world
>>> 1+2
3
>>>
```

(In practice, we will usually use the Script mode as described in Appendix A, which is a more practical way to develop a python program)

4.2 Let us now write a program in C to do exactly the same thing. Unlike the Python program, you need to first create a file to key in the program code in C. In here we will use a command line text editor program called **nano**.

- To do so, you can either use the existing Terminal (i.e. the one still runs the Python3 interpreter), or you can open another terminal by clicking the **Terminal** icon.

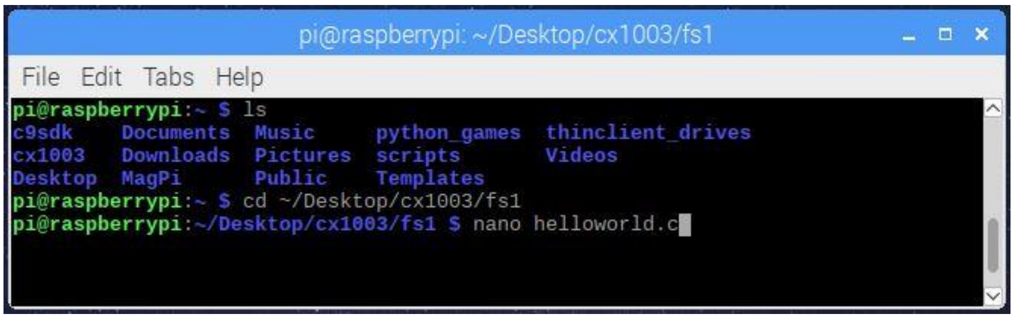
- We will open another Terminal here. (This is to also demonstrate the multi-tasking ability of the RPi, i.e. its ability to run multiple programs at the same time).



```
pi@raspberrypi:~ $
```

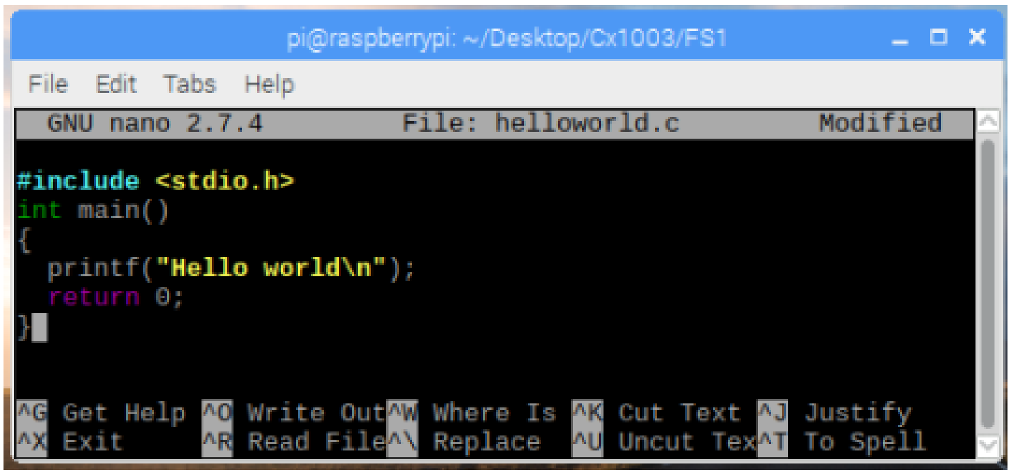
```
Python 3.5.3 (default, Jan 19 2017, 14:11:04)
[GCC 6.3.0 20170124] on linux
Type "help", "copyright", "credits" or "license" for more inform
ation.
>>> print ("Hello world")
Hello world
>>>
```

- First, change to the subdirectory of **your lab group** (e.g. fs1 in the screenshot below). Then use the text editor program, **nano** to create a file calls **`helloworld.c`** using the command as shown below



```
pi@raspberrypi:~ $ ls
c9sdk     Documents  Music      python_games  thinclient_drives
cx1003    Downloads  Pictures   scripts       Videos
Desktop   MagPi      Public     Templates
pi@raspberrypi:~ $ cd ~/Desktop/cx1003/fs1
pi@raspberrypi:~/Desktop/cx1003/fs1 $ nano helloworld.c
```

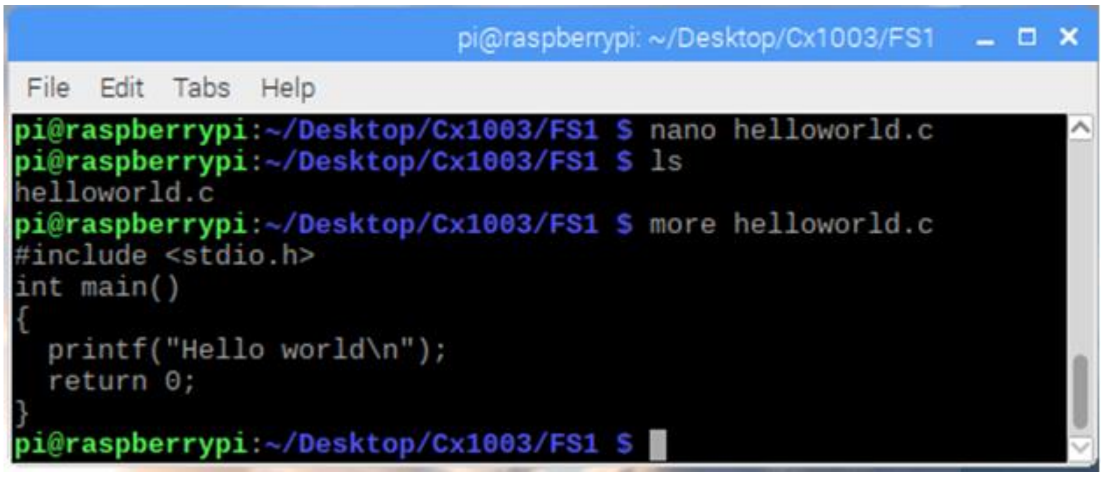
- Key in the program code as shown below



```
  GNU nano 2.7.4              File: helloworld.c              Modified

#include <stdio.h>
int main()
{
  printf("Hello world\n");
  return 0;
}

^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify
^X Exit      ^R Read File ^\ Replace   ^U Uncut Tex ^T To Spell
```

- Save the program and close the **nano** text editor by pressing the keys **Ctrl-X** (i.e. *Ctrl* key together with the *X* key), type "**y**" when prompted to 'Save modified buffer?', press [**Enter**] key to confirm save.

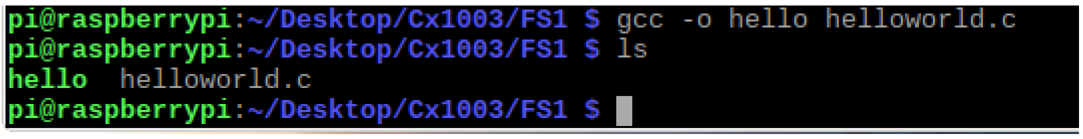
- What you have done so far is you have created a C program file. You can check the existence of this file and see its contents using the commands that you have learnt earlier.



```
pi@raspberrypi:~/Desktop/Cx1003/FS1 $ nano helloworld.c
pi@raspberrypi:~/Desktop/Cx1003/FS1 $ ls
helloworld.c
pi@raspberrypi:~/Desktop/Cx1003/FS1 $ more helloworld.c
#include <stdio.h>
int main()
{
  printf("Hello world\n");
  return 0;
}
pi@raspberrypi:~/Desktop/Cx1003/FS1 $
```

- In order to execute this C program, you will need to first **compile** it. To do so, you will need to run a compiler, and here you will use the **gcc compiler**.

- Type the gcc command as shown below. If everything is OK, then a new file **`hello`** will be created as shown - this is the executable file created by the compiler.

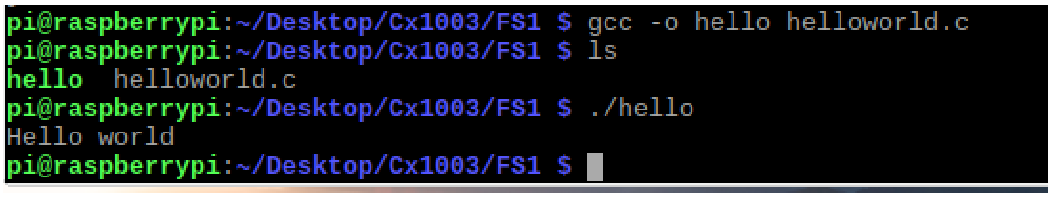


```
pi@raspberrypi:~/Desktop/Cx1003/FS1 $ gcc -o hello helloworld.c
pi@raspberrypi:~/Desktop/Cx1003/FS1 $ ls
hello  helloworld.c
pi@raspberrypi:~/Desktop/Cx1003/FS1 $
```

- If instead, you see other message(s) and cannot find the hello file, it means that you probably have made a mistake somewhere, either in your code or in the command that you typed. This is then a great opportunity for you to learn about 'debugging' – to find out what is/are wrong and fix the problem(s).

- Once you have successfully generate the executable program, type the command as shown below to run it

  `./hello`



```
pi@raspberrypi:~/Desktop/Cx1003/FS1 $ gcc -o hello helloworld.c
pi@raspberrypi:~/Desktop/Cx1003/FS1 $ ls
hello  helloworld.c
pi@raspberrypi:~/Desktop/Cx1003/FS1 $ ./hello
Hello world
pi@raspberrypi:~/Desktop/Cx1003/FS1 $
```

- If everything is OK, then the message will be printed on screen to show the output of your program.

At this juncture, you should review what you had done in these two simple programming exercises. Make sure that you understand the various steps you used to create and compile the C program to produce an executable file.

5. Displaying message on Sense Hat Module

Instead of displaying message on screen, you can also output message to the Sense Hat module with the RPi-Sense Hat setup. To provide a preview of how the Sense Hat module will be used in subsequent hands-on exercises, you will now code a Python program to display a message on the Sense-Hat module.

- First select the **python3 Terminal** that you opened earlier on. (If you had closed the python3 terminal, you can open a new terminal window, and type "python3" to invoke a pyhton3 terminal again.)

- Type the following 3 lines of code in the python3 Terminal, and observe the display on the Sense HAT module.

```
>>>from sense_hat import SenseHat
>>>sense = SenseHat()
>>>sense.show_message("Hello NTU")
```

- You can terminate the running interpreter program by pressing the two keys, *Crtl+D*

6. Summary

In the hands-on exercises here, you have learnt how the RPi can be setup as a standalone computer. You have also learnt some of the basic commands used to navigate the RPi system. You have tried writing two basic programs, in Python language and C language to appreciate the difference in interpreted language and compiled language.

******* Skip the following 'shutdown' step if you have time to continue on the Appendix**

Lastly, issue the following command in the Terminal to properly shut down the RPi computer (and goggle to find out what the command means).

```
sudo shutdown -h now
```

!! Please wait for about 10 seconds for the Rpi to complete the shutdown process before power off the Rpi power adaptor, otherwise system file corruption may occur.

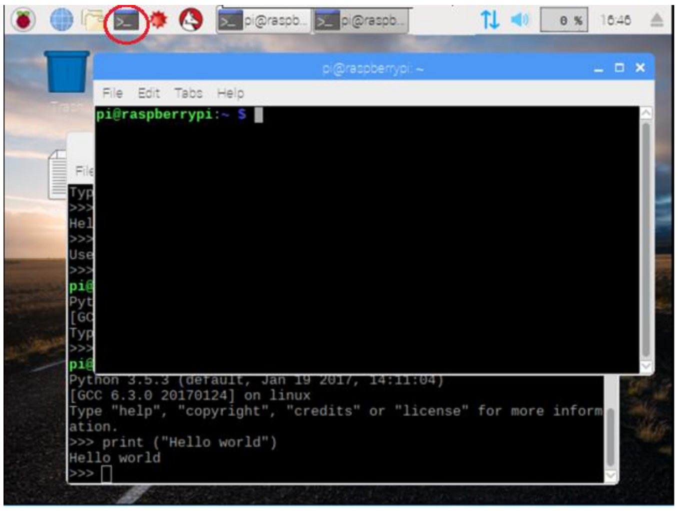
Appendix

(Please *power on the Rpi again* if you had shutdown the Rpi previously)

A. Python in Script Mode

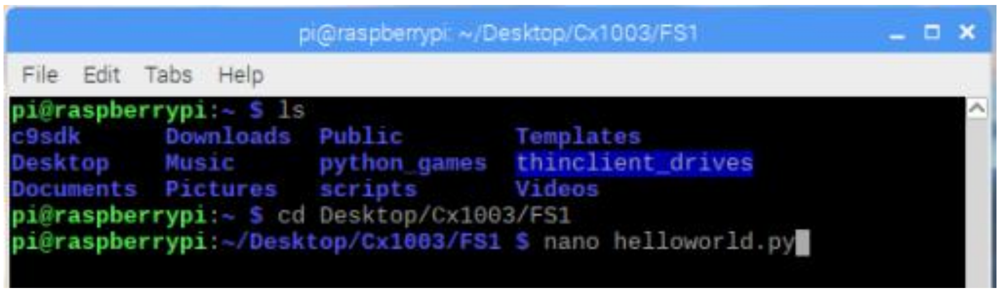
When we write python program for real-world application, we will usually store the python program code in a file, also known as script. We then run the interpreter to execute the script file.

- Open a new Terminal window, and change directory to the subdirectory of your own lab group (e.g. fs1). Eg : **cd ~/Desktop/cx1003/fs1**



```
pi@raspberrypi:~ $ 
```

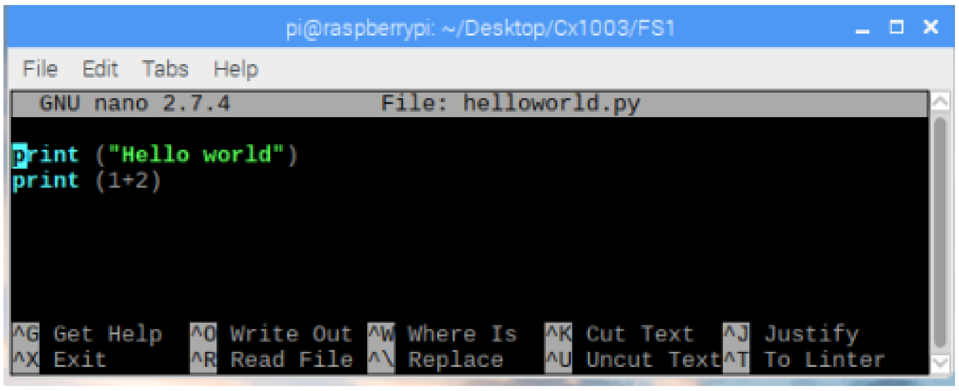
```
Python 3.5.3 (default, Jan 19 2017, 14:11:04)
[GCC 6.3.0 20170124] on linux
Type "help", "copyright", "credits" or "license" for more inform
ation.
>>> print ("Hello world")
Hello world
>>> 
```



pi@raspberrypi: ~/Desktop/Cx1003/FS1

```
pi@raspberrypi:~ $ ls
c9sdk      Downloads  Public        Templates
Desktop    Music      python_games  thinclient_drives
Documents  Pictures   scripts       Videos
pi@raspberrypi:~ $ cd Desktop/Cx1003/FS1
pi@raspberrypi:~/Desktop/Cx1003/FS1 $ nano helloworld.py
```

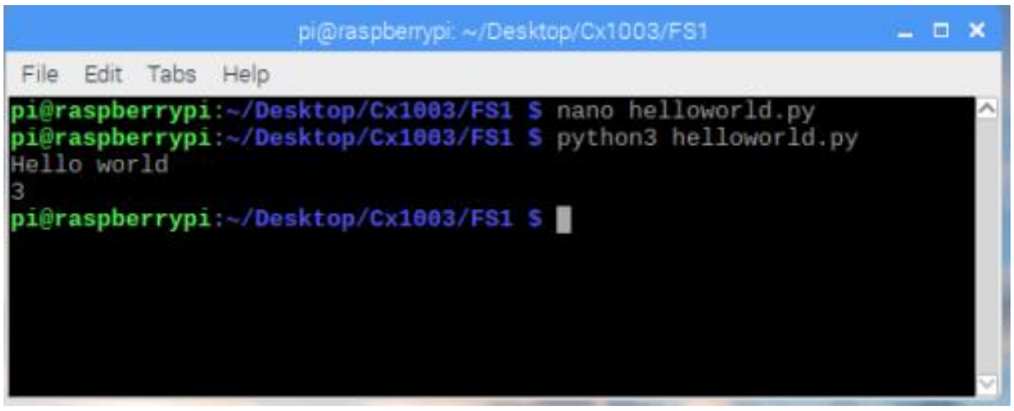
- Use the **nano** text editor, create a file calls **helloworld.py** as shown above
- Key in the code as shown below and save the file by pressing the keys **Ctrl and X**. (i.e. *Ctrl* key together with the *X* key), type "**y**" when prompted to 'Save modified buffer?' press [**Enter**] key to confirm save.
- This is the file that contains the program with code written in Python.



pi@raspberrypi: ~/Desktop/Cx1003/FS1

```
  GNU nano 2.7.4              File: helloworld.py

print ("Hello world")
print (1+2)




^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text ^T To Linter
```

- You can now use the python3 interpreter to run the python program, and you should see the messages shown on the screen.

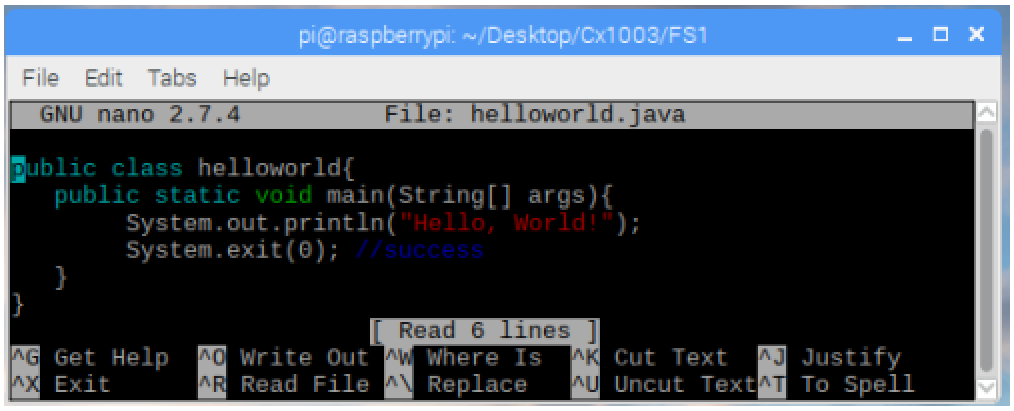


```
pi@raspberrypi:~/Desktop/Cx1003/FS1 $ nano helloworld.py
pi@raspberrypi:~/Desktop/Cx1003/FS1 $ python3 helloworld.py
Hello world
3
pi@raspberrypi:~/Desktop/Cx1003/FS1 $
```

B. Basic Java Program

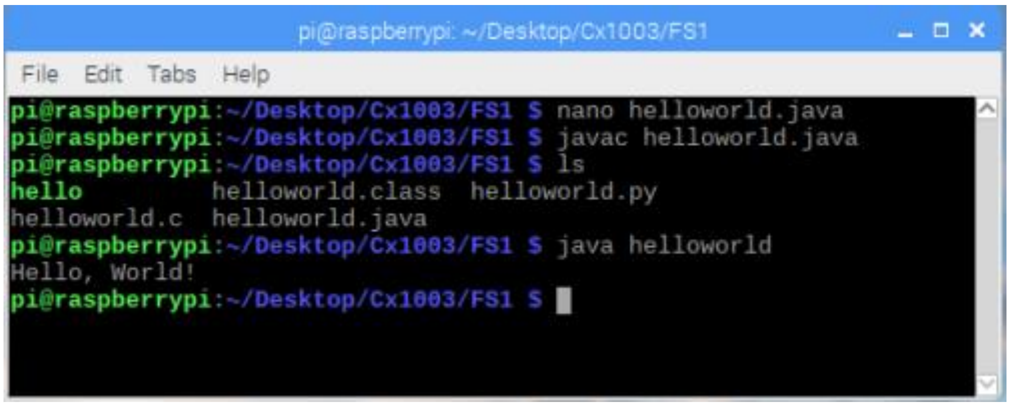
In contrast to Python and C, Java uses both a compiler and an interpreter to run its program. The following exercise shows the basic steps to develop and run a basic Java program.

- Use nano to create a text file ***helloworld.java*** and key in the code as shown below.



```
  GNU nano 2.7.4              File: helloworld.java

public class helloworld{
   public static void main(String[] args){
         System.out.println("Hello, World!");
         System.exit(0); //success
    }
}
                                [ Read 6 lines ]
^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text ^T To Spell
```

- Next compile the java program by using the ***javac*** compiler as shown below



```
pi@raspberrypi:~/Desktop/Cx1003/FS1 $ nano helloworld.java
pi@raspberrypi:~/Desktop/Cx1003/FS1 $ javac helloworld.java
pi@raspberrypi:~/Desktop/Cx1003/FS1 $ ls
hello           helloworld.class  helloworld.py
helloworld.c    helloworld.java
pi@raspberrypi:~/Desktop/Cx1003/FS1 $ java helloworld
Hello, World!
pi@raspberrypi:~/Desktop/Cx1003/FS1 $
```

- A java bytecode file ***helloworld.class*** will be created if there is no error.
- The bytecode file can then be executed by using the Java interpreter, using the following command

```
java helloworld
```

  The corresponding result message will then be shown on screen.

C. Basic Python game – Have fun !!

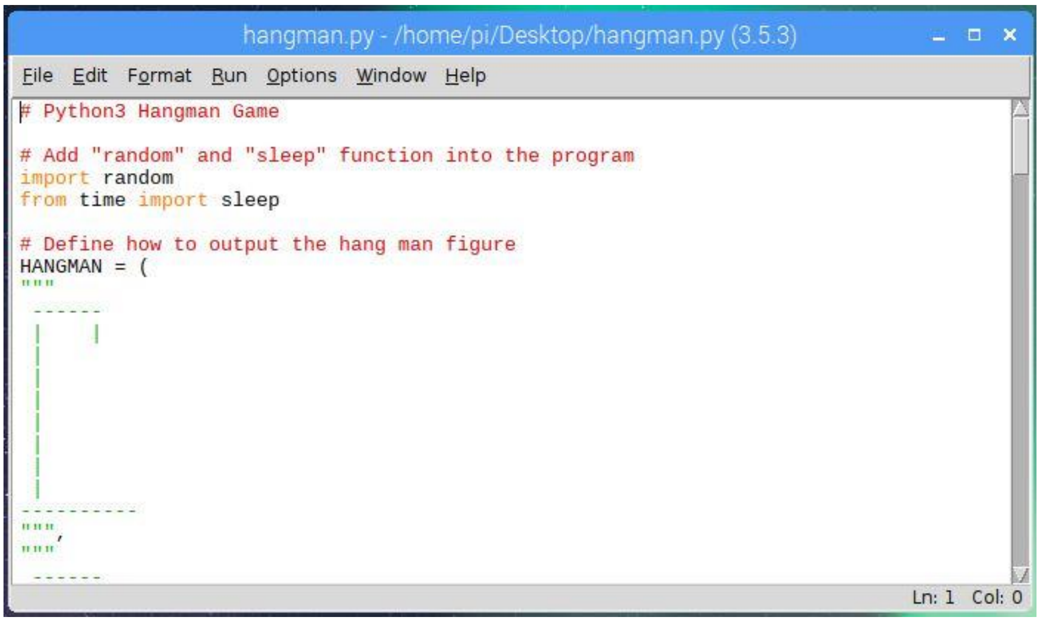
It's time to play some game and take a look on how a Python program is coded

- Launch terminal window, and copy the **hangman.py** program to the own group folder (eg Desktop\cx1003\+++)

  Please replace the "+++" with your lab group number eg fs1, if you have not created your lab group folder before, please refer to page 8 step 3.8 for folder creation steps.

```
cp /usr/local/hangman.py ~/Desktop/cx1003/+++/hangman.py
```

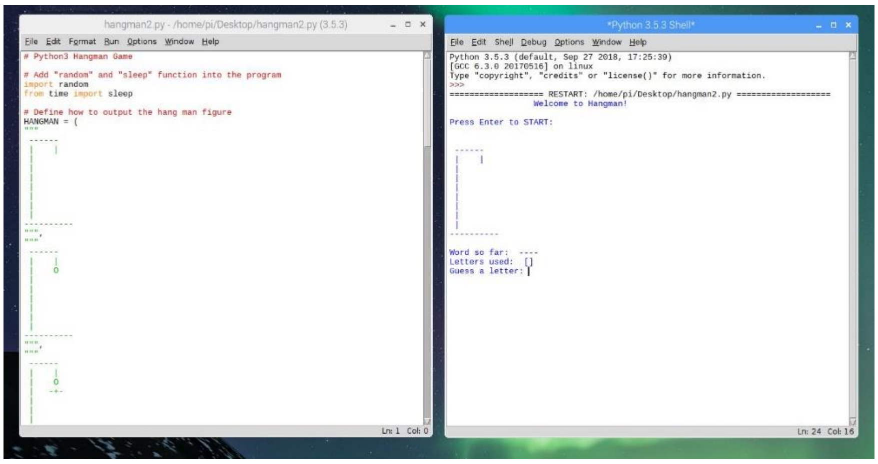
- On your **Rpi Desktop**, double click the "cx1003" folder, go to your own group folder, and double click on "hangman.py" file to open it with **Python 3 IDLE editor**. Tries to take a look at the codes, it is not difficult to understand what the codes are doing.


hangman.py - /home/pi/Desktop/hangman.py (3.5.3)
File Edit Format Run Options Window Help
Ln: 1 Col: 0

```
# Python3 Hangman Game

# Add "random" and "sleep" function into the program
import random
from time import sleep

# Define how to output the hang man figure
HANGMAN = (
"""
 ------
 |    |
 |
 |
 |
 |
 |
 |
 |
----------
""",
"""
 ------
```

Ex. #1 - Introduction to RPi

- To execute the code, select “Run” “Run Module F5” from the ***editor menu bar***, this will invoke a new python shell window where you can interact with the game. (Remarks : you can use the <F5> function key to execute the code as well)

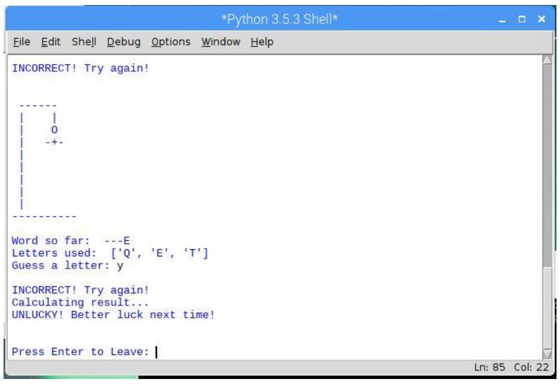


```
# Python3 Hangman Game

# Add "random" and "sleep" function into the program
import random
from time import sleep

# Define how to output the hang man figure
HANGMAN = (
"""
 ------
 |    |
 |
 |
 |
 |
 |
 |
----------
""",
"""
 ------
 |    |
 |    O
 |
 |
 |
 |
 |
----------
""",
"""
 ------
 |    |
 |    O
 |   -+-
 |
```

```
Python 3.5.3 (default, Sep 27 2018, 17:25:39)
[GCC 6.3.0 20170516] on linux
Type "copyright", "credits" or "license()" for more information.
>>>
=================== RESTART: /home/pi/Desktop/hangman2.py ===================
                 Welcome to Hangman!

Press Enter to START:

 ------
 |    |
 |
 |
 |
 |
 |
 |
----------

Word so far:  ----
Letters used:  []
Guess a letter:
```

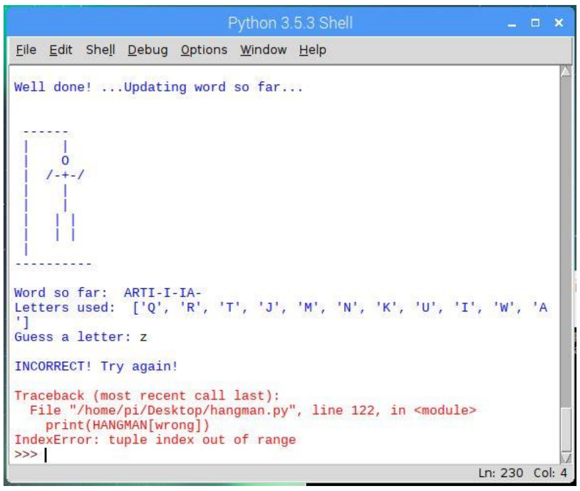
- Tries to play the game a few times, you will find that there are some errors in the codes :

  - The program will end too early before the whole hang man process

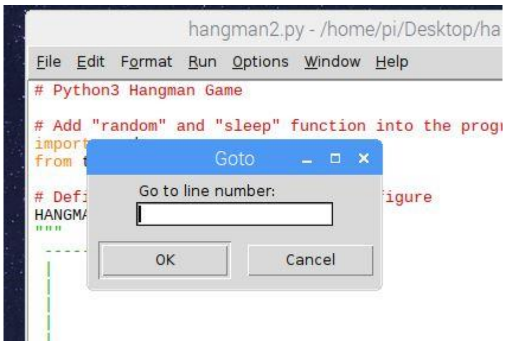


```
INCORRECT! Try again!


 ------
 |    |
 |    O
 |   -+-
 |
 |
 |
 |
 |
----------

Word so far:  ---E
Letters used:  ['Q', 'E', 'T']
Guess a letter: y

INCORRECT! Try again!
Calculating result...
UNLUCKY! Better luck next time!


Press Enter to Leave:
```

  - The program will end with program error message if you complete the hang man process. (Remark : You can use “Edit” “Go to Line” on the python editor to see the line of code that give the error)



```
Well done! ...Updating word so far...


 ------
 |    |
 |    O
 |  /-+-/
 |    |
 |    |
 |   | |
 |   | |
 |
----------

Word so far:  ARTI-I-IA-
Letters used:  ['Q', 'R', 'T', 'J', 'M', 'N', 'K', 'U', 'I', 'W', 'A
']
Guess a letter: z

INCORRECT! Try again!

Traceback (most recent call last):
  File "/home/pi/Desktop/hangman.py", line 122, in <module>
    print(HANGMAN[wrong])
IndexError: tuple index out of range
>>>
```



hangman2.py - /home/pi/Desktop/ha
File Edit Format Run Options Window Help
Goto
Go to line number:
OK
Cancel

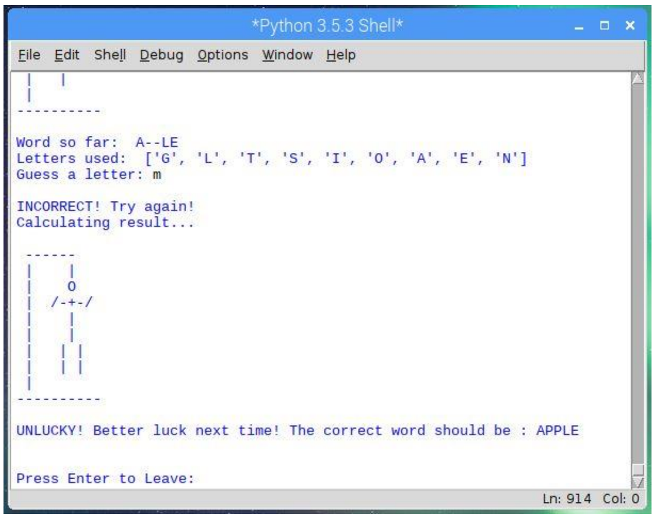
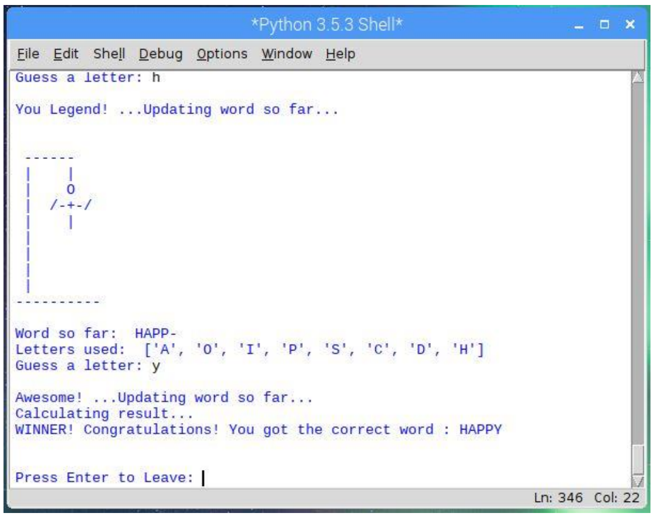
Ex. #1 - Introduction to RPi

- Take a look at the code, and try to modify it to eliminate the error and make the game more functional, a possible end result is as shown below

  Hints :

  - Line 106 : WORDS = ("CODE", "ARTIFICIAL")   *** you can add more words
  - Line 110 : MAX_WRONG = len(word) – 1
  - Line 122 : print(HANGMAN[wrong])

    You may need to add a separate counter for the hangman figure instead of using the “wrong” counter
  - You may need to insert a few more lines of code

Possible game output - *losing*	Possible game out - *winning*
	

Python 3.5.3 Shell
File Edit Shell Debug Options Window Help
Ln: 914 Col: 0

```
 |    |
 |
----------

Word so far:  A--LE
Letters used:  ['G', 'L', 'T', 'S', 'I', 'O', 'A', 'E', 'N']
Guess a letter: m

INCORRECT! Try again!
Calculating result...

  ------
 |     |
 |     O
 |   /-+-/
 |     |
 |     |
 |    | |
 |    | |
 |
----------

UNLUCKY! Better luck next time! The correct word should be : APPLE


Press Enter to Leave:
```

Python 3.5.3 Shell
File Edit Shell Debug Options Window Help
Ln: 346 Col: 22

```
Guess a letter: h

You Legend! ...Updating word so far...


  ------
 |     |
 |     O
 |   /-+-/
 |     |
 |
 |
 |
 |
----------

Word so far:  HAPP-
Letters used:  ['A', 'O', 'I', 'P', 'S', 'C', 'D', 'H']
Guess a letter: y

Awesome! ...Updating word so far...
Calculating result...
WINNER! Congratulations! You got the correct word : HAPPY


Press Enter to Leave: |
```

- No worry if you are unable to modify the code, this is just the start of your programming journey, just enjoy and have fun. Very soon you will be able to perfect the code with ease.

- Well done!, you have completed lab 1, please perform a proper shutdown for the Raspberry Pi using the following command in terminal window.

```
sudo shutdown -h now
```

!! Please wait for about 10 seconds for the Rpi to complete the shutdown process before power off the Rpi power adaptor, otherwise system file corruption may occur.