

**Course** : Diploma in Electronics & Computer Engineering (EGDF20)

**Module**  : Connected System Design Project (EGE205)

**Laboratory No**. : Lab 1a

**Laboratory Title** : Introduction: Getting Started with BeagleBone Black Wireless (BBBW) Board

**Objective** : To familiarize with BeagleBone Black Wireless (BBBW) Board, work on Linux shell

command and write some basic python program.

**Hardware**  : BBBW Board with USB Cable x1

: Micro SD Card loaded with Debian OS Image x1

**Contents**

1. Getting to know the BeagleBone Black Wireless (BBBW) Board
2. Setting up the BeagleBone Black Wireless (BBBW) Board
   1. Updating the BBBW with latest Debian OS Image
   2. Logging in to the BBBW board via Secure Shell (SSH)
3. Working on Linux Shell Command
   1. Basic Linux Shell Command
   2. Connecting the BBBW Board to the Internet using Linux Shell Command
4. Writing Python Program in the BeagleBone Black Wireless (BBBW) Board
   1. Using the Cloud9 IDE
   2. Python 101

# **Getting to know the BeagleBone Black Wireless (BBBW) Board**

**BeagleBone Black Wireless (BBBW) board** is a low-cost, community-supported development platform for developers and hobbyists. It boots in Linux under 10 seconds and get started on the development in less than 5 minutes with just a single USB cable. This popular open source BeagleBone Black Wireless (BBBW) board now comes with built-in onboard 802.11 b/g/n 2.4GHz Wi-Fi and Bluetooth wireless networking capability. It is one of the easiest to use and modify credit-card sized IoT Linux computer available.

**Hardware Specification:**

* Octavo Systems OSD3358 1GHz ARM® Cortex-A8 Processor
  + 512MB DDR3 RAM
  + 3D graphics accelerator
  + NEON floating-point accelerator
  + 2x PRU 32-bit microcontrollers
* 4GB 8-bit eMMC on-board flash storage
* USB client for power & communications
* USB host
* 802.11b/g/n and Bluetooth 4.1 + BLE
* HDMI
* 2x 46 pin headers

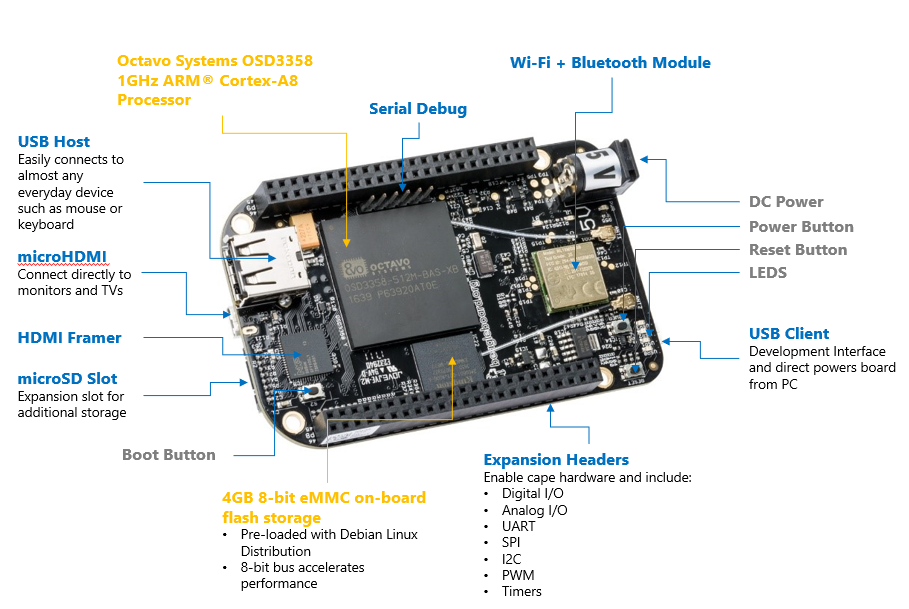


Figure 1a: BeagleBone Black Wireless (BBBW) Board

**Software Compatibility:**

* Debian with Cloud9 IDE on Node.js with BoneScript library.
* Third party support for Android and Ubuntu

# **Setting up the BeagleBone Black Wireless (BBBW) Board**

## Updating the BBBW with latest Debian OS Image

1. **Download** the latest Debian OS Image (*AM3358 Debian 10.3 2020-04-06 4GB eMMC IoT Flasher*) from <https://beagleboard.org/latest-images> as shown in the Figure below. It is a compressed sector-by-sector image of the SD card. The file downloaded will have an .img.xz extension with the size of approximately 500 MB. It may take up to 30 mins to download the Image.

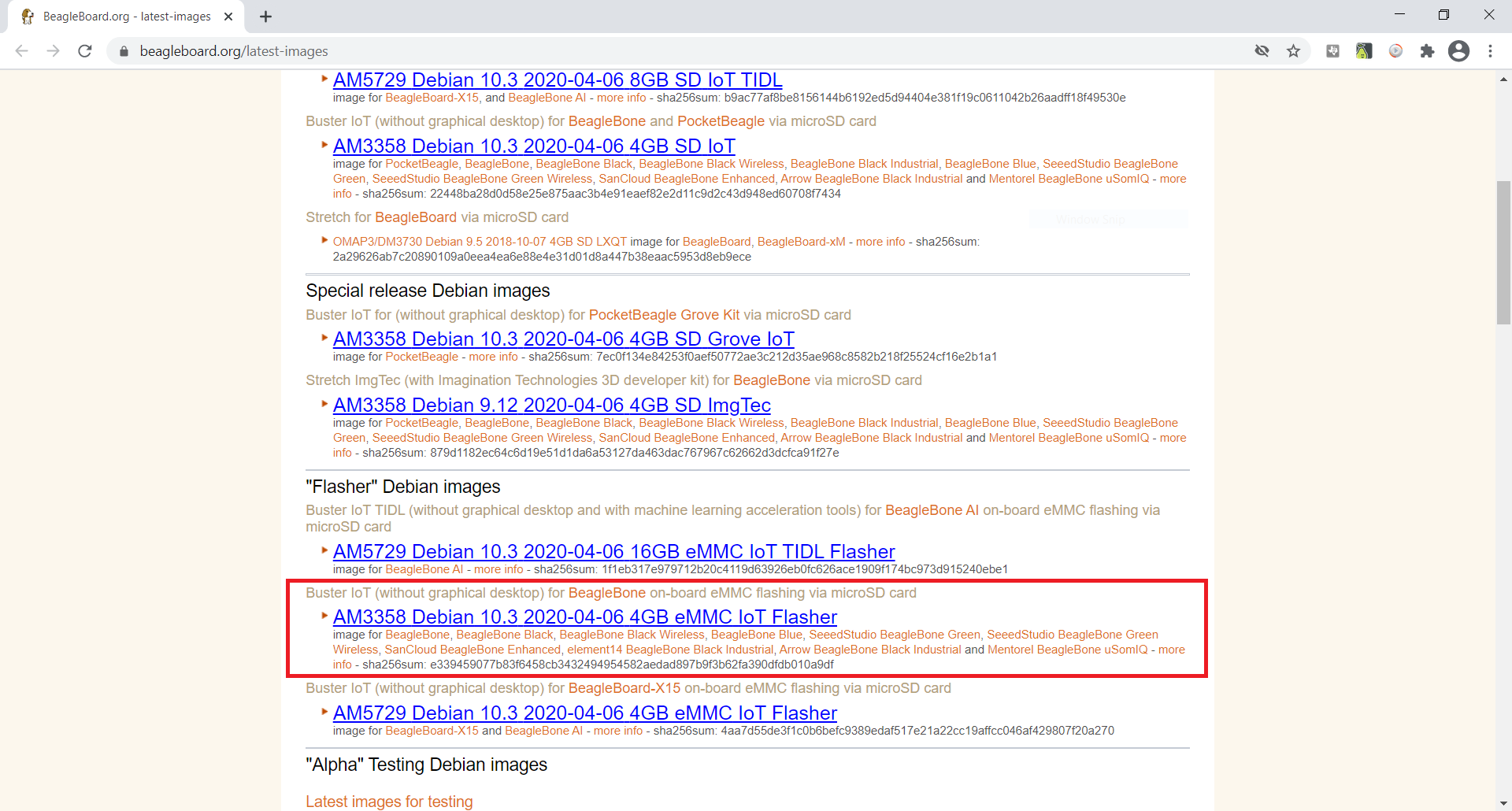


Figure 2.1a: Latest Debian OS Image Download Page

1. **Download** the balenaEtcher software from <https://www.balena.io/etcher> as shown in the Figure below. It is a SD card programming software. The file downloaded will have an .exe extension with the size of approximately 125 MB. It may take up to 5 mins to download the software.

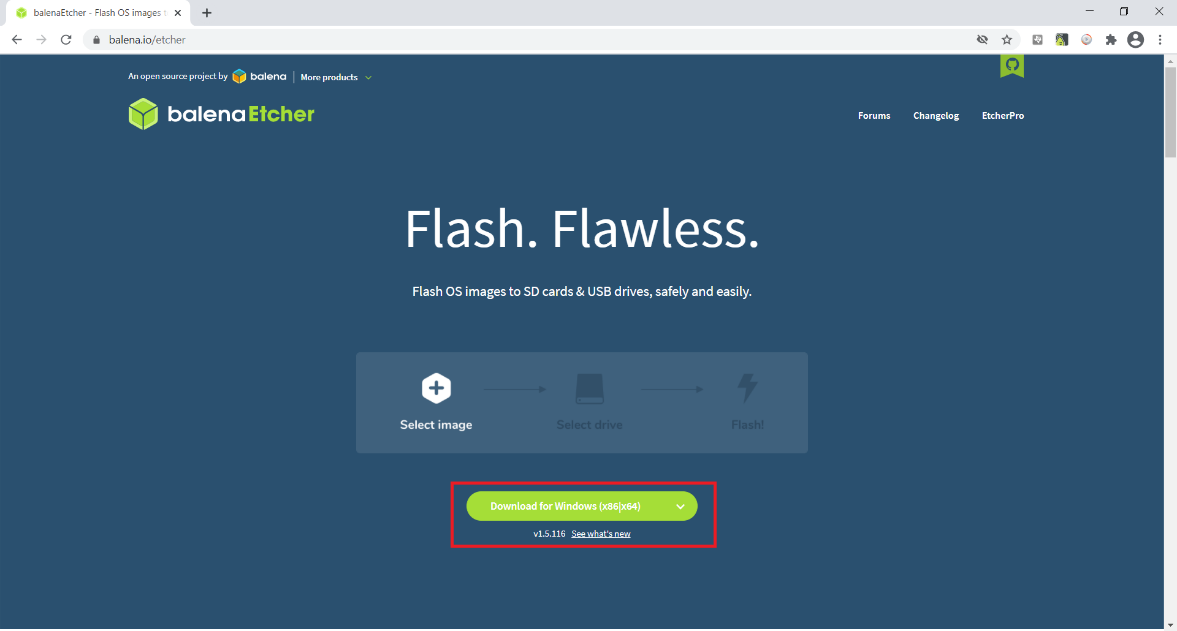


Figure 2.1b: BalenaEtcher Software Download Page

1. **Double clicking** on the downloaded .exe file to install the balenaEtcher software and a pop-up window is seen as shown in the Figure below. **Click** on the “I Agree” button to start the installation.

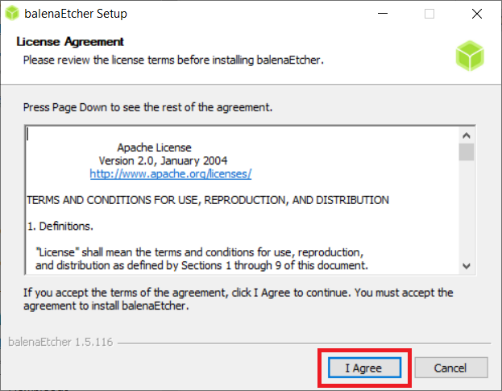


Figure 2.1c: BalenaEtcher Software Installation

1. The balenaEtcher software is launched upon completion of the installation as shown in the Figure below.

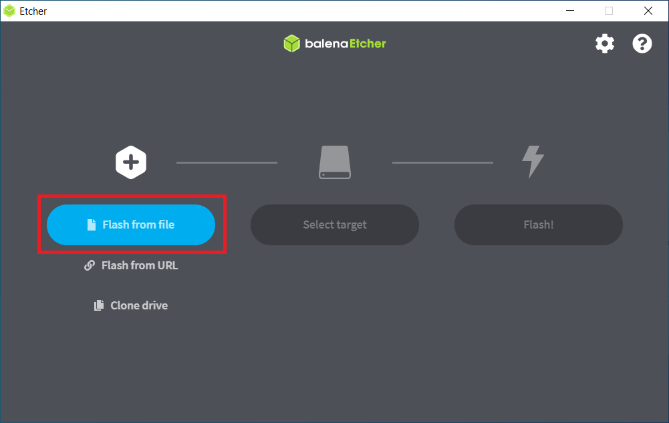


Figure 2.1d: BalenaEtcher Software

1. **Insert** a micro SD card to the computer's micro SD card slot or using a USB adapter to connect the micro SD card to the computer.
2. **Format** the micro SD card as prompted by Windows.
3. **Click** on the “Flash from file” button as shown in the Figure above (Figure 2.1d). **Locate** and **select** the latest Debian OS Image downloaded in Step 1 as shown in the Figure below.

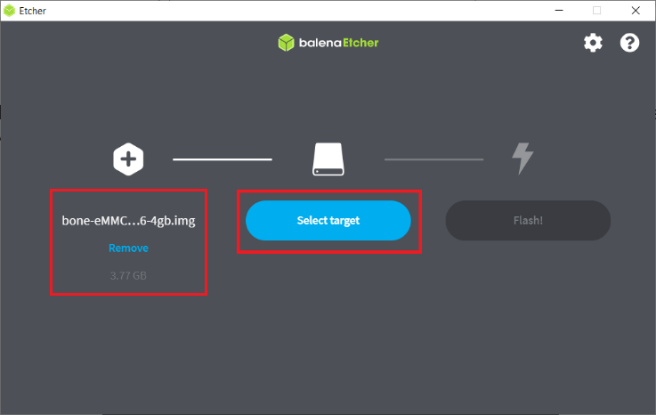


Figure 2.1e: Select the latest Debian OS Image

1. **Click** on the “Select target” button as shown in the Figure above (Figure 2.1e). **Select** the drive of the inserted micro SD card as shown in the Figure below.

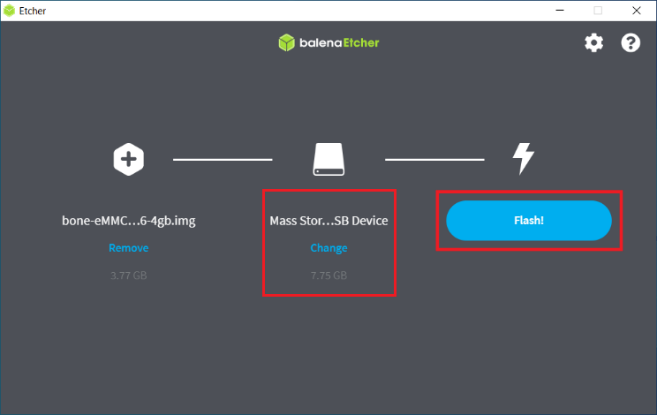


Figure 2.1f: Select the target

1. **Click** on the “Flash!” button as shown in the Figure above (Figure 2.1f) to start the image flashing process as shown in the Figure below. It may take up to 10 mins to complete the flashing process.

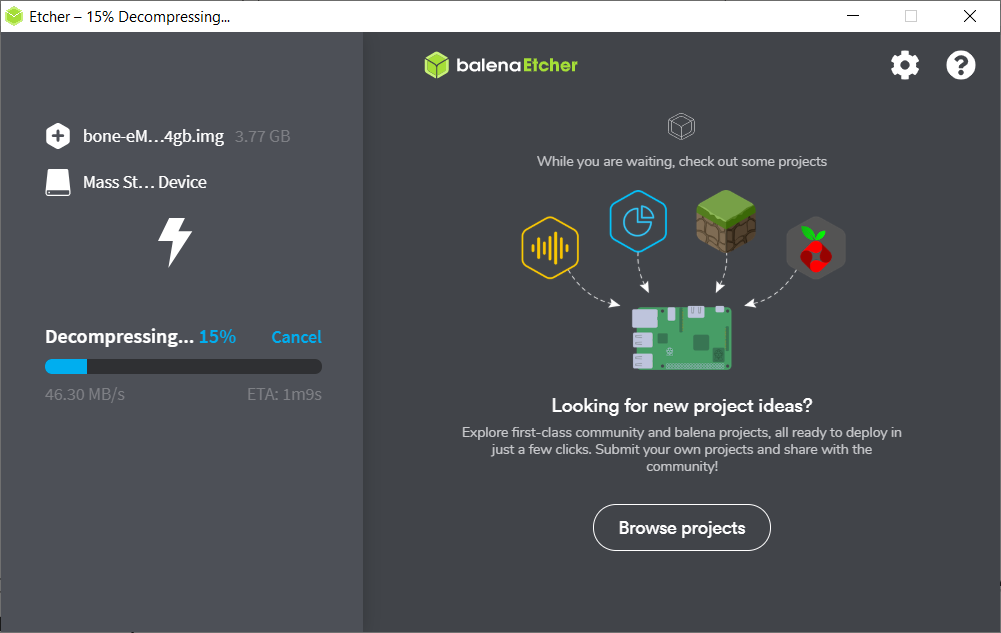


Figure 2.1g: Image Flashing in Progress

1. **Eject** the newly programmed micro SD card from the computer upon completion of the image flashing process.
2. **Insert** the newly programmed micro SD card into the micro SD slot located at the BBBW board (see Figure1a).
3. **Connect** the BBBW board to the computer using a USB cable as shown in the Figure below. The board is automatically powered-up through USB and will detect the present of the micro SD card. It will start flashing the image from the micro SD card to the on-board eMMC flash memory. The flashing may take up to 15 minutes. When the flashing is completed, all the 4 USRx LEDs will be steady off and the BBBW board is automatically powered down.

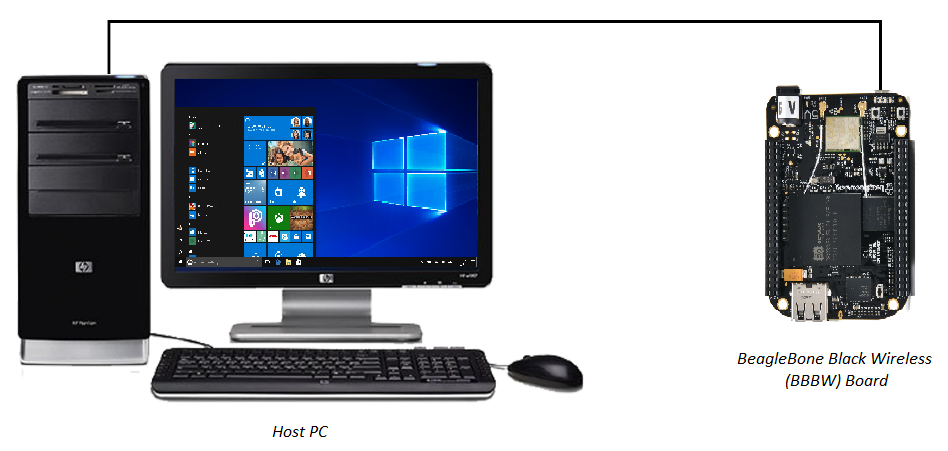


Figure 2.1h: Connecting BBBW to Computer

1. **Eject** the micro SD card from the BBBW board.
2. **Press** the on-board power button (see Figure 1a) to power up the BBBW board again. The BBBW board will start to boot using the latest OS Image found on the eMMC flash memory. BBBW is now operating with the latest Debian OS version!

## Logging in to the BBBW board via Secure Shell (SSH)

**Secure Shell** (SSH) is a method for secure remote login from one computer to another. Typical applications include remote command-line, login, and remote command execution. On Windows, PuTTY software will be used as the SSH client to remotely log into the BBBW board.

1. **Download** the PuTTy software -> <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html> as shown in the Figure below. **Choose** the 64-bit installer. The file downloaded will have an .exe extension with the size of approximately 500 KB. It should take less than a minute to download the software.

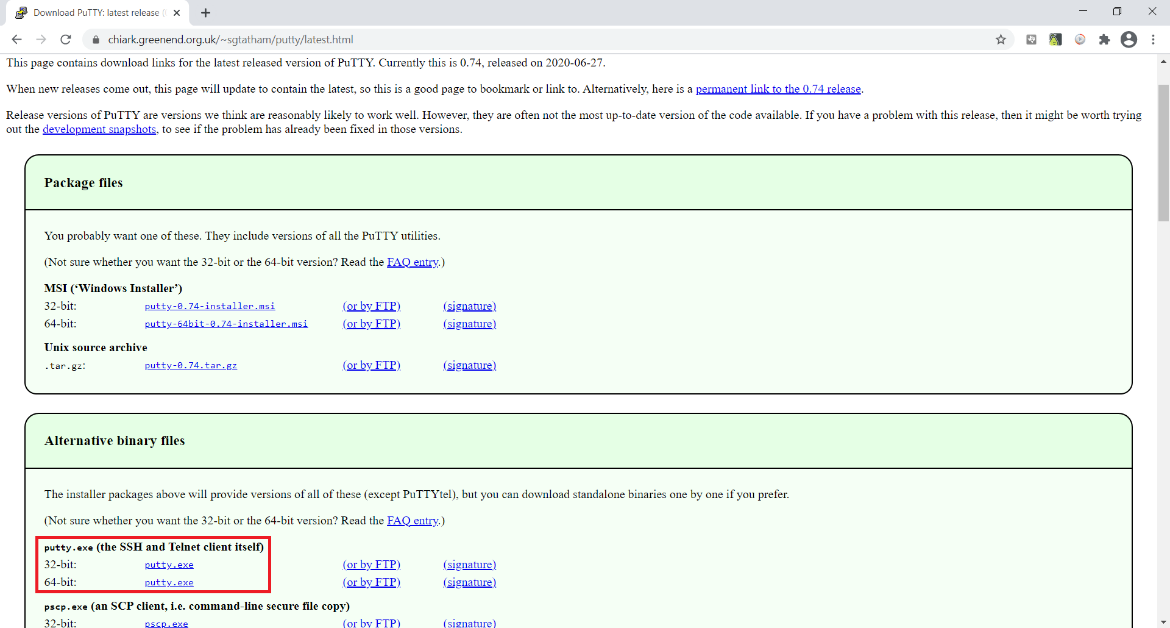


Figure 2.2a: Putty Software Download Page

1. **Double click** on the downloaded putty.exe file to run the software as shown in the Figure below.

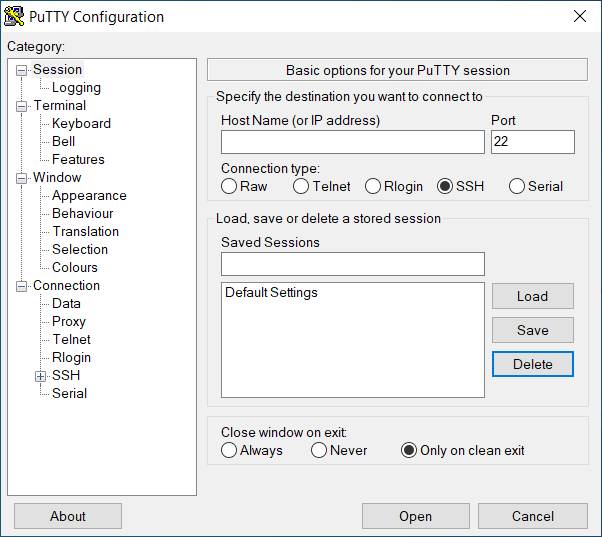


Figure 2.2b: Putty Software

1. **Configure** the SSH settings on the PuTTY software with the information as shown in the Figure below. Then, **click** on the “Open” button.

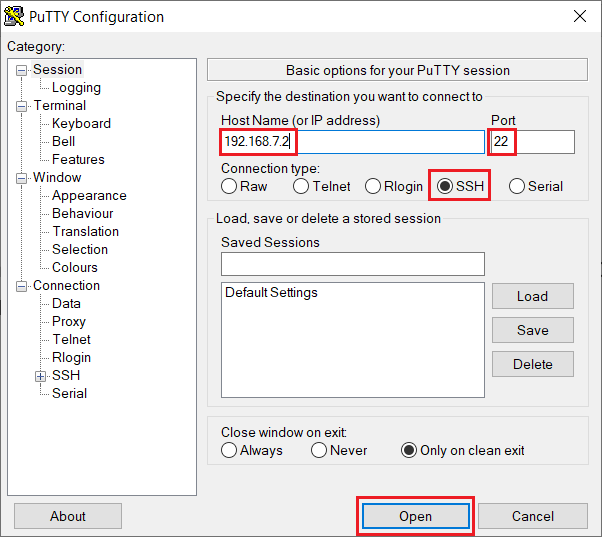


Figure 2.2c: Putty Software SSH Settings

1. **Click** on the “Yes” button if a security alert message pop up as shown in the Figure below.

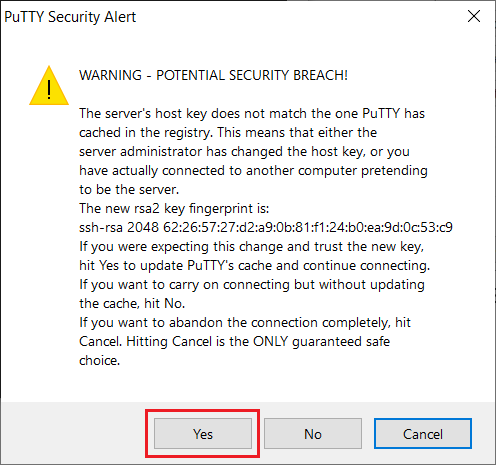


Figure 2.2d: Putty Security Alert

1. **Type** in the default username “**debian**” and **hit** the “Enter” key from the login prompt on the Putty screen as shown in the Figure below.



Figure 2.2e: Putty Login Prompt

1. **Type** in the default password “**temppwd**” and **hit** the “Enter” key when prompted for password as shown in the Figure below (*Note: Nothing will be shown on the terminal when keying in the password*).

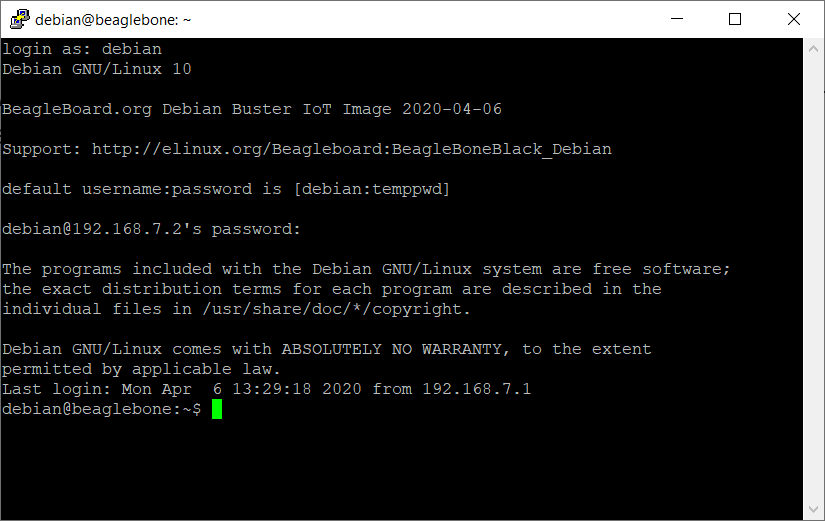


Figure 2.2f: Putty Password Prompt

1. The BBBW board has been successfully logged in through SSH!

# **Working on Linux Shell Command**

## Basic Linux Shell Command

The Shell is a program that takes commands from the keyboard and gives them to the operating system to perform. When logging in to an operating system, the standard shell is displayed and allows some common operations such as copy files or restart the system. Let's try some basic Shell commands and do something on the BBBW board.

1. **Type** in the command “**uname -r**” and **hit** the “Enter” key. This command returns the information of the Linux kernel version as shown in the Figure below.



Figure 3.1a: The “uname -r” Command

1. **Type** in the command “**date**” and **hit** the “Enter” key. This command returns the information of the current date of the BBBW board as shown in the Figure below.



Figure 3.1b: The “date” Command

1. **Type** in the command “**ls**” and **hit** the “Enter” key. The command “**ls**” stands for **l**i**s**t. It returns the information of the folders’ and files’ name located in the current working directory as shown in the Figure below.



Figure 3.1c: The “ls” Command

1. **Type** in the command “**pwd**” and **hit** the “Enter” key. The command “**pwd**” stands for **p**rint **w**orking **d**irectory. It returns the information of the current working directory as shown in the Figure below.



Figure 3.1d: The “pwd” Command

1. **Type** in the command “**mkdir WorkSpace**” and **hit** the “Enter” key. The command “**mkdir**” stands for **m**a**k**e **dir**ectory. It creates a folder with a name (case-sensitive). The name of the folder must be typed in followed by the command as shown in the Figure below.



Figure 3.1e: The “mkdir” Command

1. **Type** in the command “**ls**” and **hit** the “Enter” key again. It is observed that the folder named WorkSpace has been created as shown in the Figure below.



Figure 3.1f: The “ls” Command

1. **Type** in the command “**cd WorkSpace**” and **hit** the “Enter” key. The command “**cd**” stands for **c**hange **d**irectory. It switches between the folders. The name of the folder must be typed in followed by the command as shown in the Figure below.



Figure 3.1g: The “cd” Command

1. **Type** in the command “**pwd**” and **hit** the “Enter” key again. It is observed that the current working directory has been changed to /home/Debian/WorkSpace from /home/Debian as shown in the Figure below.



Figure 3.1h: The “pwd” Command

1. **Type** in the command “**cat > sample.txt**” and **hit** the “Enter” key. The command “**cat**” is derived from its function to con**cat**enate files. It is one of the most basic commands that is used to read, write, and append data to files in Shell. It is observed that a file named sample.txt is created and opened as shown in the Figure below.

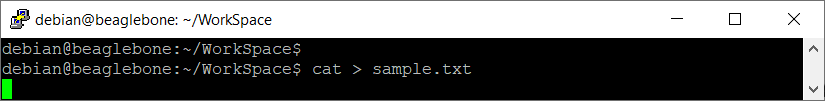


Figure 3.1i: The “cat” Command

1. **Type** in “**I love NYP**” and **hit** the “Enter” key. This is to input a line of text in the file sample.txt as shown in the Figure below.

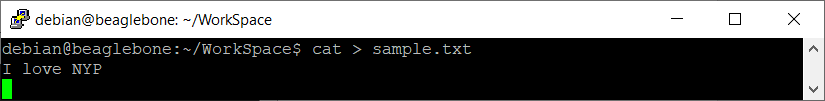


Figure 3.1j: Entering Text into a File

1. **Press** CTRL+D (hold down Ctrl Key and hit the “d” key). This will save the content and exit the file sample.txt as shown in the Figure below.

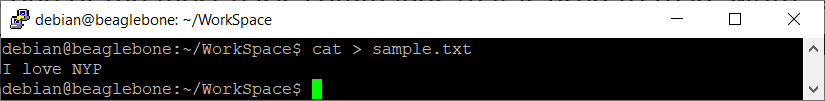


Figure 3.1k: Exiting the File

1. **Type** in the command “**more sample.txt**” and **hit** the “Enter” key. The command “**more**” returns the contents of file sample.txt as shown in the Figure below.

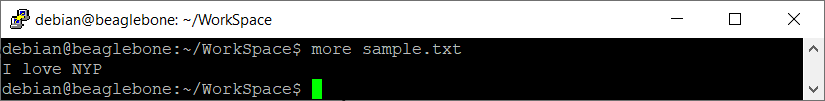


Figure 3.1l: The “more” Command

1. **Type** in the command “**ls**” and **hit** the “Enter” key. It is observed that the file sample.txt is created in the WorkSpace folder as shown in the Figure below.



Figure 3.1m: The “ls” Command

1. **Type** in the command “**rm sample.txt**” or “**rm /home/debian/WorkSpace/sample.txt**” and **hit** the “Enter” key. The command “**rm**” stands for **r**e**m**ove. It can be used to delete any file by typing the filename or filename along with path followed by the command. It is observed that the file sample.txt is deleted as shown in the Figure below.

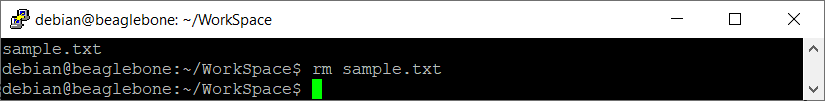




Figure 3.1n: The “rm” Command

1. **Type** in the command “**ls**” and **hit** the “Enter” key again. It is observed that the file sample.txt is no longer exist in the WorkSpace folder as shown in the Figure below.

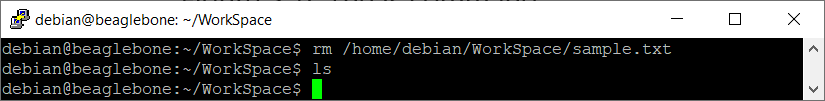


Figure 3.1o: The “ls” Command

1. Congratulations! You have successfully picked up some of the basic Linux Shell commands.

## Connecting the BBBW Board to the Internet using Linux Shell Command

**ConnMan** is an internet connection manager for embedded devices running the Linux operating system and **Connmanctl** is a ConnMan command line interface (CLI) that can handle most simple network connections. It can enable/disable any technology that exists on the system such as Wi-Fi, display a list of services available, connect to/disconnect from any unsecured networks etc.

1. **Log in** to the BBBW Board through SSH using the default username “**debian**” and password “**temppwd**” as shown in the Figure below.

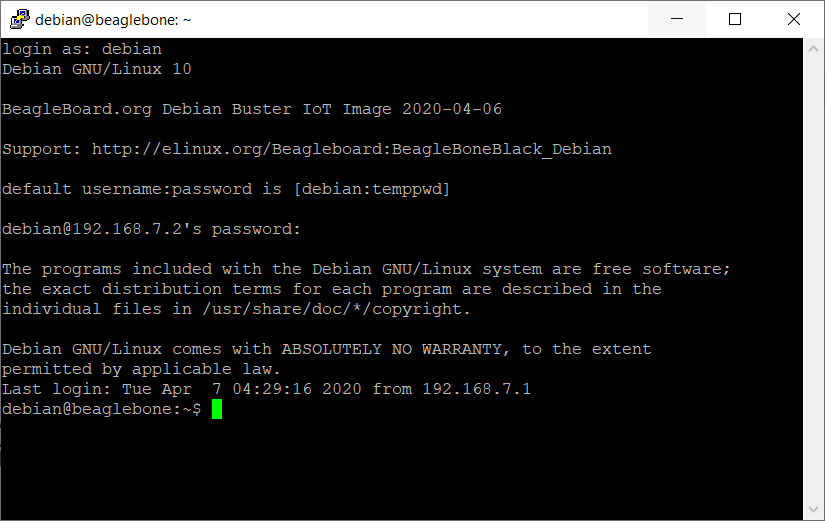


Figure 3.2a: Logging in to the BBBW Board

1. **Type** in the command “**sudo connmanctl**” and **hit** the “Enter” key. **Type** in the password “**temppwd**” and **hit** the “Enter” key again. The command “**Sudo**” stand for **S**uper **U**ser **Do**. It allows the program to run with the security privileges of another user. It is observed that “**connmanctl>**” indicates that the **Connmanctl** CLI has been successfully accessed as shown in the Figure below (*Note: Ignore the Error on getting VPN connection*)

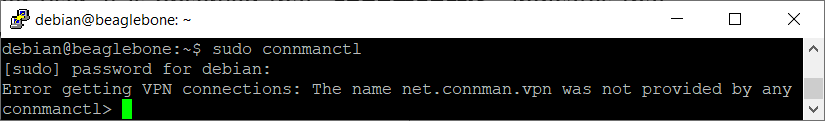


Figure 3.2b: Accessing Connmanctl CLI

1. **Type** in the command “**scan wifi**” and **hit** the “Enter” key. It is observed that the message “**Scan completed for wifi**” is returned as shown in the Figure below.



Figure 3.2c: The “scan wifi” Command

1. **Type** in the command “**services**” and **hit** the “Enter” key. It is observed that the surrounding detected Wi-Fis are listed as shown in the Figure below.

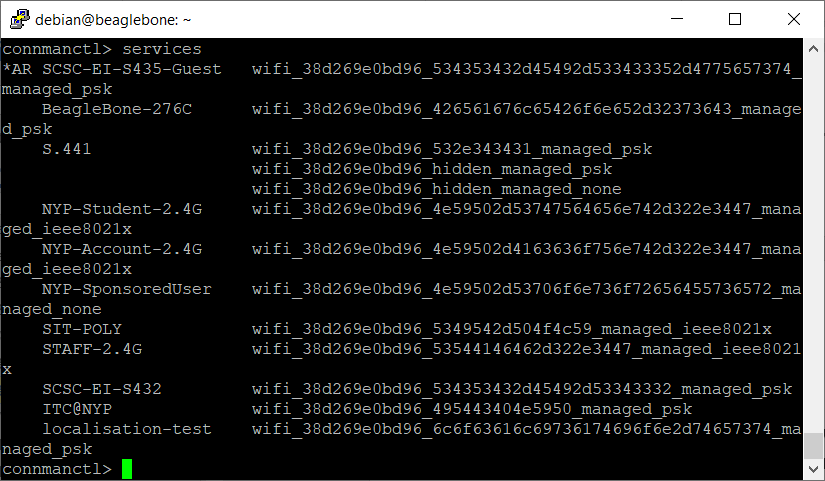


Figure 3.2d: The “services” Command

1. **Type** in the command “**agent on**” and **hit** the “Enter” key. It is observed that the message “**Agent registered**” is returned as shown in the Figure below.



Figure 3.2e: The “agent on” Command

1. **Type** in the command

“**connect wifi\_38d269e0bd96\_534353432d45492d533433352d4775657374\_managed\_psk**” and **hit** the “Enter” key. **Type** in the Wi-Fi passphrase “**helloworld**” and **hit** the “Enter” key again. This is to connect to the Wi-Fi with the SSID of S435-CSDL\_dev. It is observed that the message “**Connected wifi\_38d269e0bd96\_534353432d45492d533433352d4775657374\_managed\_psk**” is returned as shown in the Figure below.

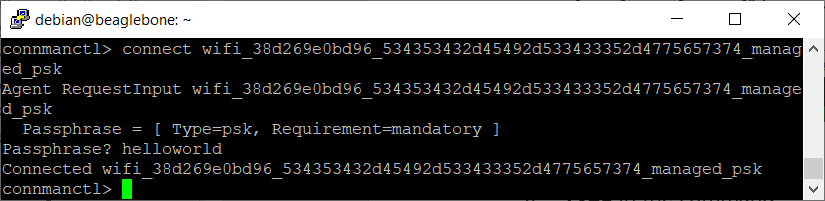


Figure 3.2f: The “connect” Command

1. **Type** in the command “**quit**” and **hit** the “Enter” key. This is to exit the **Connmanctl** CLI as shown in the Figure below.



Figure 3.2g: The “quit” Command

1. **Type** in the command “**ping -c 1 www.google.com**” and **hit** the “Enter” key. It is observed that the message “**64 bytes from 172.217.194.103: icmp\_seq=1 ttl=53 time=50.5 ms**” is returned indicating that the BBBW board has been successfully connected to the internet as shown in Figure the below.

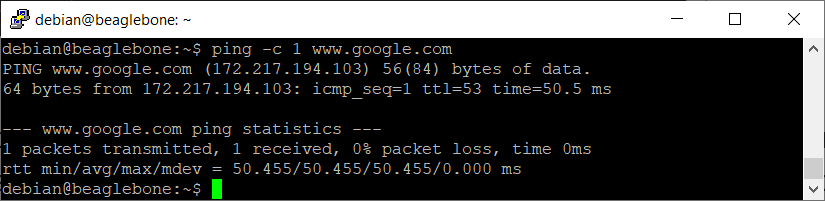


Figure 3.2h: The “ping” Command

# **Writing the Python Program in the BeagleBone Black Wireless (BBBW) Board**

## Using the Cloud9 IDE

**Cloud9 IDE** is an open-source web-based programming platform that supports several programming languages. This great piece of software comes installed on the BBBW board by default. To access the software in the BBBW board, a web browser and SSH connection is required.

1. **Ensure** that the BBBW board is powered up and connected to the computer through a USB cable.
2. **Open** the web browser (preferably Chrome browser) and **type** in “**http://192.168.7.2:3000**” in the address bar as shown in the Figure below.

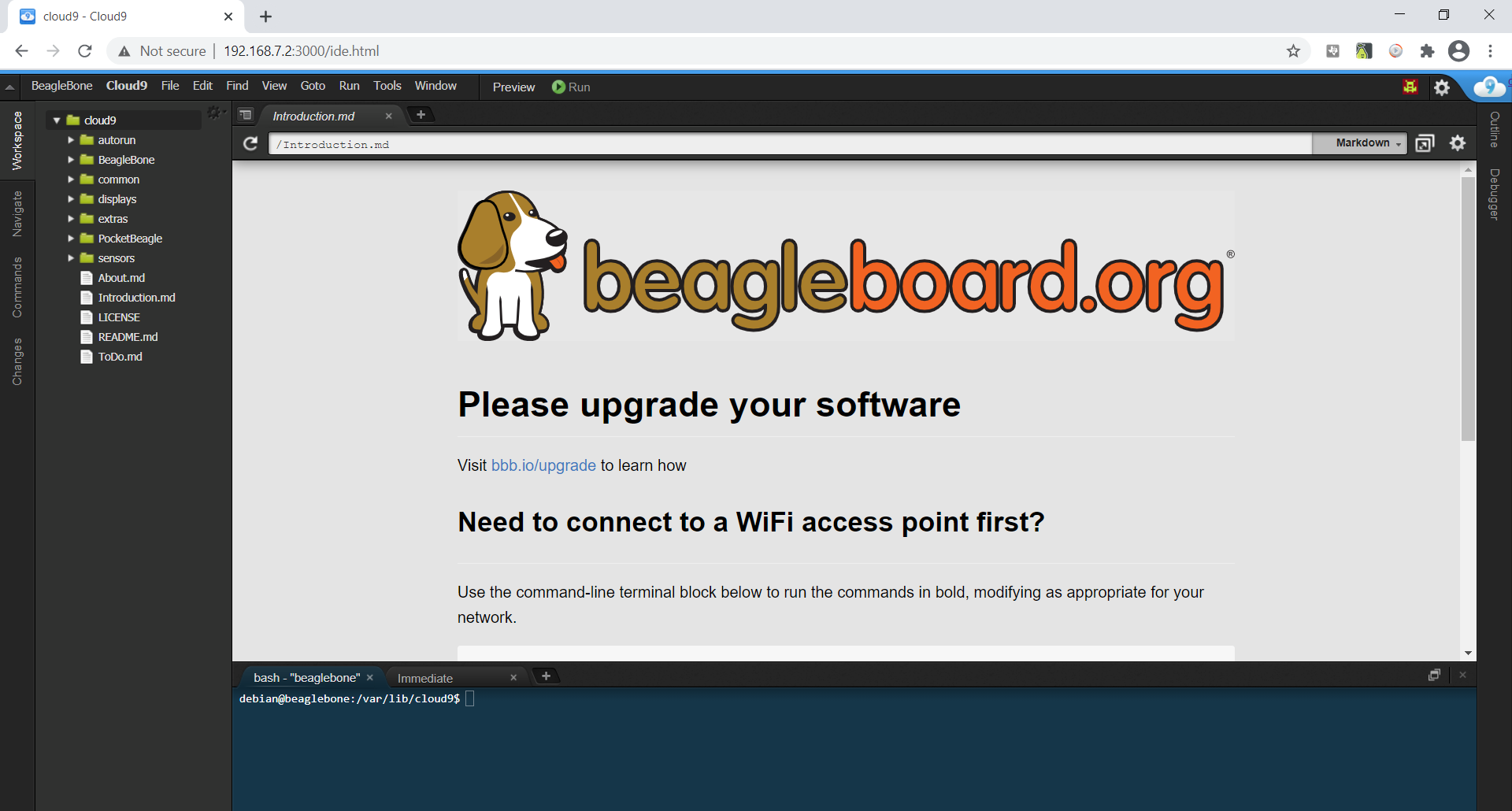


Figure 4.1a: Cloud9 IDE

1. There are six main sections of the Cloud9 IDE shown in the Figure below.
2. **Menus Tab**: The menus in the Cloud9 IDE are organized in a very familiar way to any computer application: File, Edit, Find, View, Goto, Run, Tools and Window.
3. **Workspace**: A place where all working folders and files are being accessed. Everything is organized in a hierarchy.
4. **Editor**: A place where codes are being written. The editor highlights the functions according to the syntax of the programming of the file opened.
5. **Console**: When a script or piece of code is executed, the console prints the output of the application. Those messages are commonly used for debugging purpose.
6. **Debugger**: The debugger is the perfect way to see exactly what is happening when a script is running. Breakpoints can be created so that code runs only to a certain line. The status of the code and which values are stored the variables can also be observed.
7. **Terminal**: Instead of using PuTTy software, it provides an alternative to control the BBBW board directly from the web browser, to update or install new software, move files, and perform other commands.

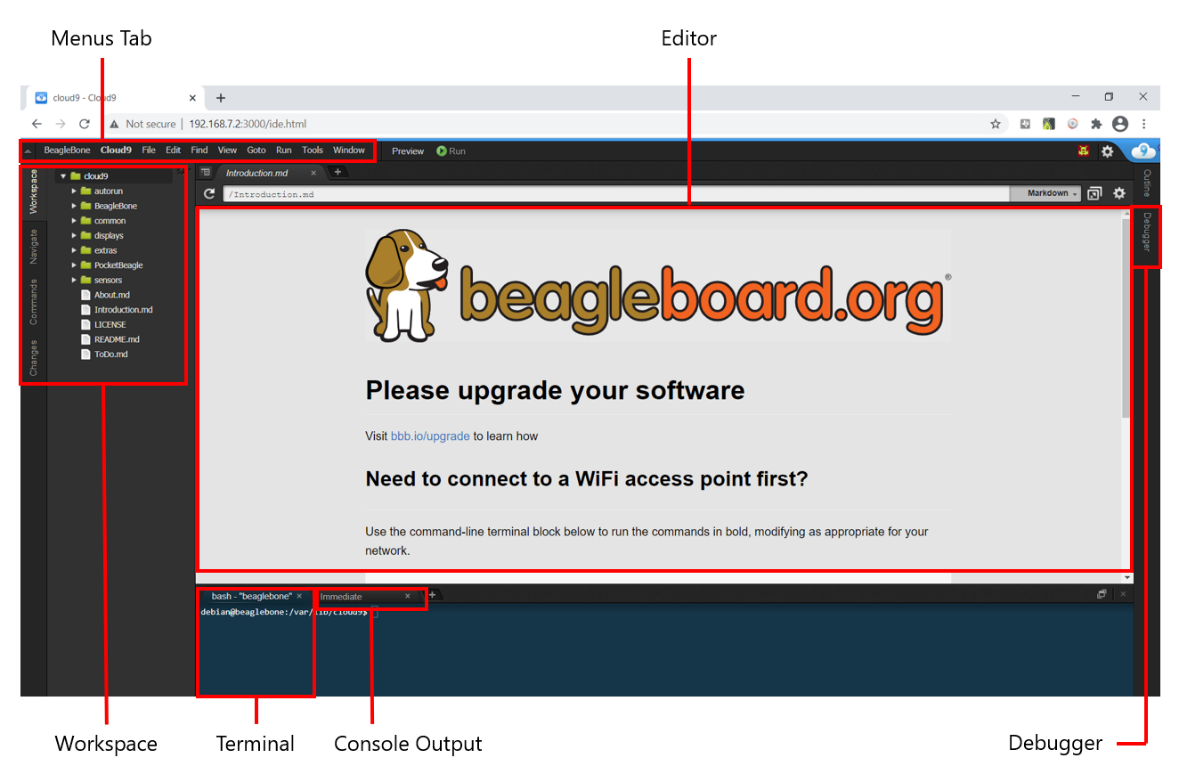


Figure 4.1b: Cloud9 IDE Main Sections

1. **Right click** on the cloud9 folder under the Workspace section and **select** the “**New Folder**” from the drop-down menu to create a new folder. **Name** the folder as “**MyFirstPythonProject**” as shown in the Figure below.

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

Figure 4.1c: Creating a New Folder

1. **Right click** on the newly created folder “**MyFirstPythonProject”** and **select** the “**New File**” from the drop-down menu to create a new python file. **Name** the file as “**MyFirstPythonProgram.py**” as shown in the Figure below.

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

Figure 4.1d: Creating a New File

1. **Double click** on the newly created file “**MyFirstPythonProgram.py**” and **enter** the following code “**print("Hello, World!")**” into the file under the Editor section as shown in the Figure below.

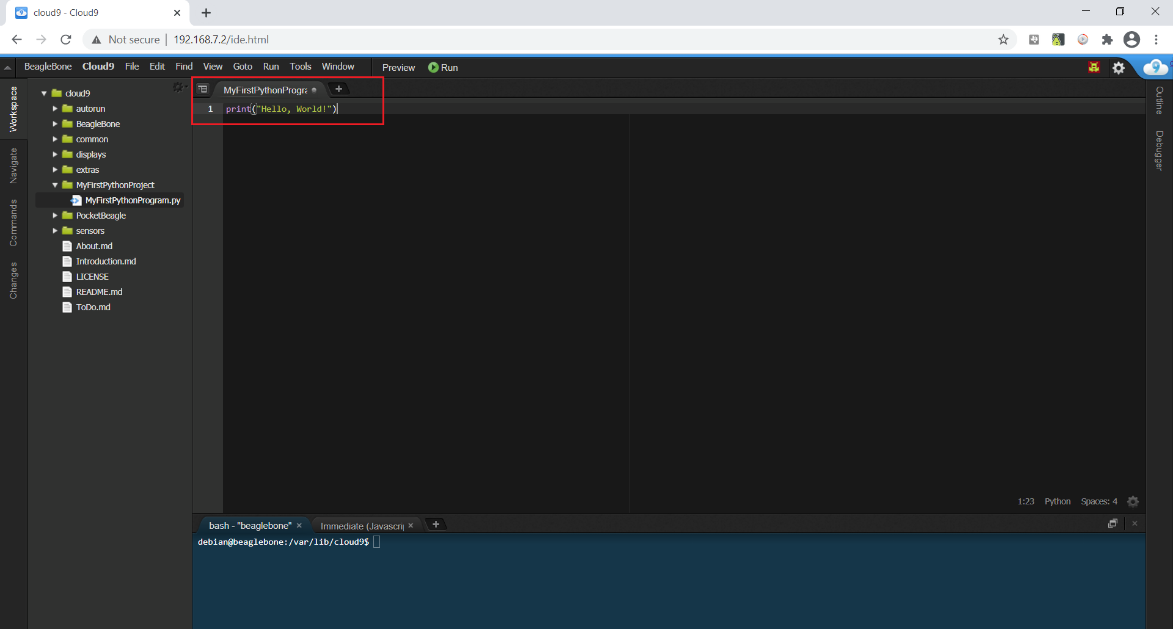


Figure 4.1e: Entering Python Code

1. **Click** on the “**File**” under the Menu Tab section and select “**Save**” from the drop-down menu to save the file as shown in the Figure below.

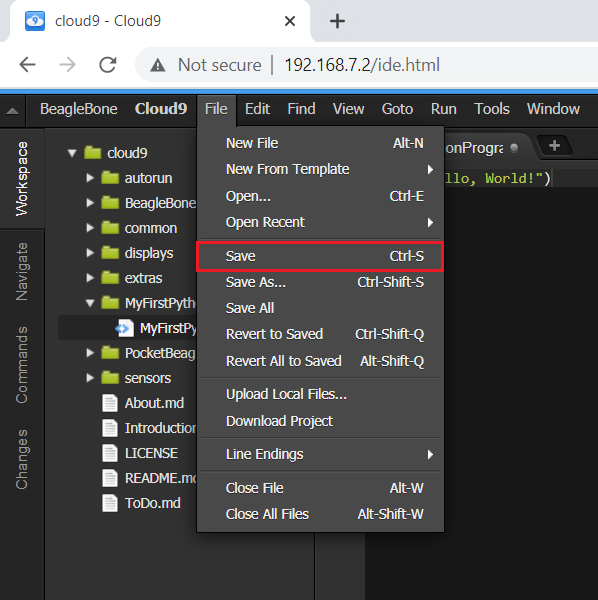


Figure 4.1f: Saving the file

1. **Click** on the “Run” button located beside the Menu Tab to execute the file. It is observed that the text “**Hello, World!**” is printed at the Console section as shown in the Figure below.

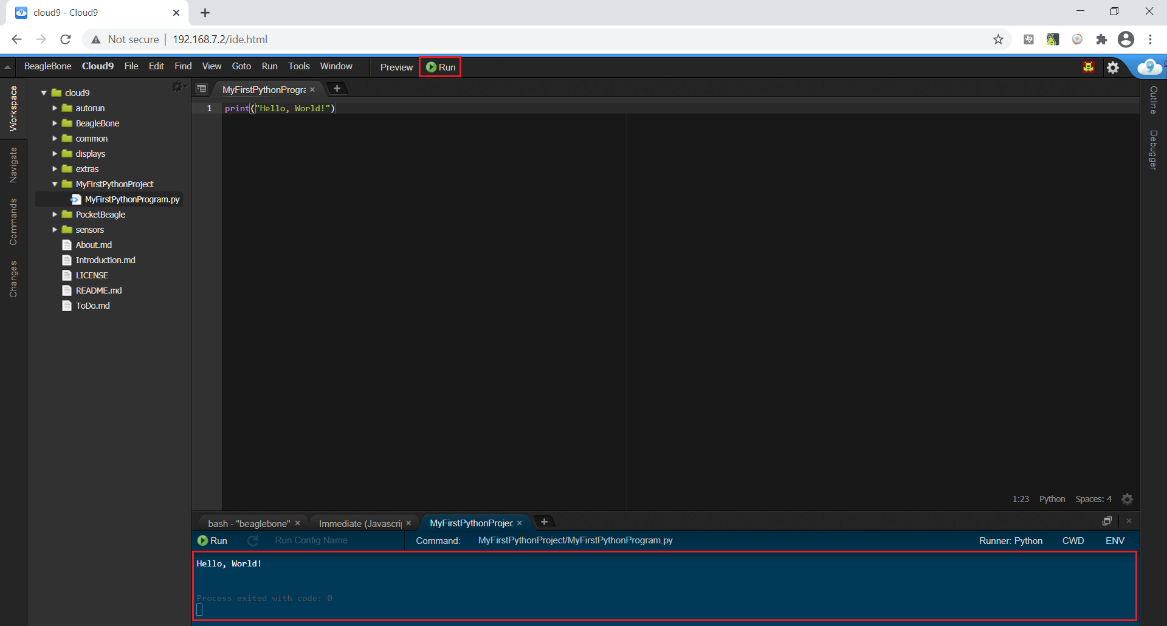


Figure 4.1g: Executing the file

1. You have successfully created your first python program using the Cloud9 IDE!

## Python 101

**Variable Types**

**Python** is completely object oriented. Every **variable** in Python is an object. No declaration of variables and their types is needed before using them.

For **numbers**, Python supports two types which is integers (whole numbers) and floating-point numbers (decimals).

For **Strings**, it is defined using single quote (‘ ‘) or a double quotes (“ “). The difference between the two is that using double quotes allows apostrophes in a sentence to be included without terminating the strings.

1. **Enter** the following code into the Cloud9 IDE.

|  |
| --- |
| myint = 8  myfloat = 8.0  mystring = "It's John Birthday!"  print("Integer: %d" % myint)  print("Float: %f" % myfloat)  print("String: %s" % mystring) |

1. **Click** on the “Run” button to execute the code.
2. **Observe** and **Compare** the program output at the console output window with your teammates and consult your lecturer for advice if it is not the same.
3. **Write** the program output in the white box below for future reference if needed.

|  |
| --- |
| *Right click and select “New comment” to insert your program as a comment.* |

**List & Tuple**

**Lists** are used to store multiple items within a **square** bracket in a single variable. List items are **ordered**, **changeable**, and **allow duplicate** values.

**Tuples** are used to store multiple items within a **round** brackets in a single variable. Tuple items are **ordered**, **unchangeable**, and **allow duplicate** values.

1. **Enter** the following code into the Cloud9 IDE.

|  |
| --- |
| mylist = ["Alex", "David", "Peter", "David"]  mytuple = ("Alex", "David", "Peter", "David")  mylist[3] = "John"  print(mylist)  mytuple[3] = "Jackson"  print(mytuple) |

1. **Click** on the “Run” button to execute the code.
2. **Observe** and **Compare** the program output at the console output window with your teammates and consult your lecturer for advice if it is not the same.
3. **Write** the program output in the white box below for future reference if needed.

|  |
| --- |
| *Right click and select “New comment” to insert your program as a comment.* |

**Basic Arithmetic and Strings Operators**

Python support arithmetic operators such as addition, subtraction, multiplication, and division operators. All the arithmetic operators can be used with numbers. On top of that, Python also supports concatenating strings using the addition operator.

1. **Enter** the following code into the Cloud9 IDE.

|  |
| --- |
| num1 = 18  num2 = 7  num3 = 2  result = (num1 % num2) \* num3  output = "Result" + " " + "="  print("%s %d" % (output, result)) |

1. **Click** on the “Run” button to execute the code.
2. **Observe** and **Compare** the program output at the console output window with your teammates and consult your lecturer for advice if it is not the same.
3. **Write** the program output in the white box below for future reference if needed.

|  |
| --- |
| *Right click and select “New comment” to insert your program as a comment.* |

**Conditions Statement (if… else…)**

Python uses Boolean logic to evaluate conditions. The Boolean values True and False are returned when an expression is compared or evaluated. Notice that variable assignment is done using a single equal operator "**=**", whereas comparison between two variables is done using the double equals operator "**==**". The "not equals" operator is marked as "**!=**".

The "**and**", "**or**" and “**not**” Boolean operators can be used together with the condition statement to allow building complex Boolean expressions. The "**in**" operator could also be used to check if a specified object exists within an iterable object container.

1. **Enter** the following code into the Cloud9 IDE.

|  |
| --- |
| num1 = 6  num2 = 2  result = 0  operator = "multiply"  if operator in ["multiply", "divide"]:  if num1 == 6 and num2 == 2:  result = num1 \* num2  print("The result is %d" % result)  else:  result = num1 / num2  print("The result is %d" % result) |

1. **Click** on the “Run” button to execute the code.
2. **Observe** and **Compare** the program output at the console output window with your teammates and consult your lecturer for advice if it is not the same.
3. **Write** the program output in the white box below for future reference if needed.

|  |
| --- |
| *Right click and select “New comment” to insert your program as a comment.* |

**Loops Statement (for & while loop)**

There are two types of loops in Python, for and while.

For loops can iterate over a sequence of numbers using the "**range**" functions. The range function returns a sequence of numbers of that specified range. While loops repeat as long as a certain Boolean condition is met.

“**break**” is used to exit a for loop or a while loop, whereas “**continue**” is used to skip the current block and return to the "for" or "while" statement.

1. **Enter** the following code into the Cloud9 IDE.

|  |
| --- |
| count = 0  for x in range(10):  if x % 2 == 0:  continue  print(x)  print("Number 1,3,5,7,9 are printed!")  while True:  print(count)  count += 1  if count >= 5:  print("Number 0,1,2,3,4 are printed!")  break |

1. **Click** on the “Run” button to execute the code.
2. **Observe** and **Compare** the program output at the console output window with your teammates and consult your lecturer for advice if it is not the same.
3. **Write** the program output in the white box below for future reference if needed.

|  |
| --- |
| *Right click and select “New comment” to insert your program as a comment.* |

**Function**

**Function** is a convenient way to divide code into useful blocks, allowing code to be arranged in order, make it more readable, reuse it and save some time. Also, function is a key way to define interfaces so programmers can share their code.

Function in python is defined using the block keyword “**def**”, followed with the function's name as the block's name. Function may also receive arguments (variables passed from the caller to the function) and may return a value to the caller, using the keyword “**return**”.

1. **Enter** the following code into the Cloud9 IDE.

|  |
| --- |
| def MyFirstFunction():  print("Hello From My First Function!")  def MyFirstFunctionWithInputs(Name, Greeting):  print("Hi %s!, %s" % (Name, Greeting))  def Sum(a, b):  return a + b  MyFirstFunction()  MyFirstFunctionWithInputs("Johnson", "Happy New Year!")  x = Sum(8,9)  print("8 + 9 = %d" % x) |

1. **Click** on the “Run” button to execute the code.
2. **Observe** and **Compare** the program output at the console output window with your teammates and consult your lecturer for advice if it is not the same.
3. **Write** the program output in the white box below for future reference if needed.

|  |
| --- |
| *Right click and select “New comment” to insert your program as a comment.* |

*Congratulations! You have successfully completed the Lab1a. Good job! Take a good break and stay tune for next lab. More exciting lab exercises coming to you!*