Setting up the ESP8266 for Wireless Communication

Author: David Carvajal ([davidcarvajal@ufl.edu](mailto:davidcarvajal@ufl.edu))

The ESP8266 is a lightweight SoC with wireless communication options through a radio antenna. It contains its own microprocessor, which can be controlled through different programming options. The factory firmware supports simple setup through AT commands ([more info here](https://www.espressif.com/sites/default/files/documentation/4a-esp8266_at_instruction_set_en.pdf)), but in this implementation we decided to use Micropython, which is a trimmed down version of Python that operates similarly.

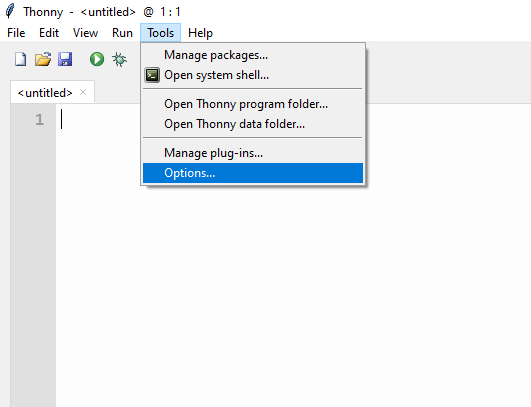
This setup guide assumes the ESP8266 you are using is already programmed with the Micropython firmware. For more information on installing the firmware check the “Micropython Flashing” guide or visit the [Micropython ESP8266 Tutorial/Setup](http://docs.micropython.org/en/v1.15/esp8266/tutorial/index.html).

For our purposes we believe the best IDE for programming and setup of the ESP8266 is the Thonny IDE. The download link and information for this IDE can be found on the [Thonny Website](https://thonny.org/).

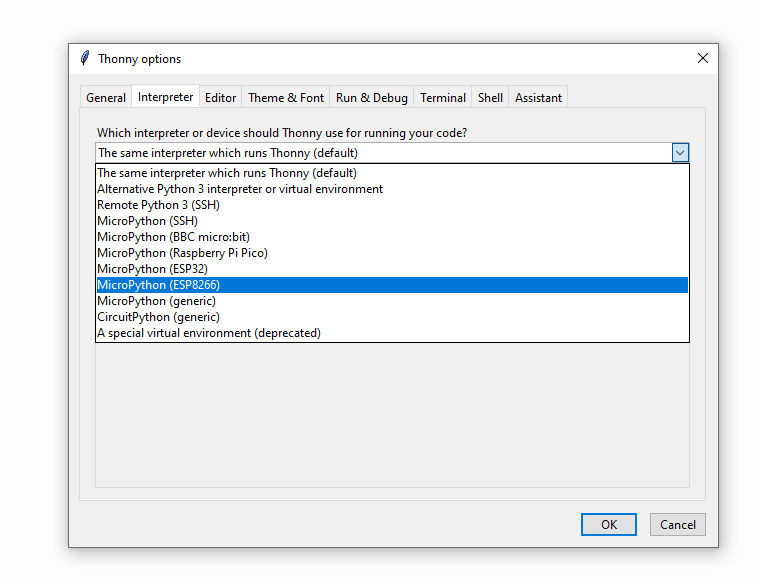
# Wired Setup

This first section will assume you have an ESP8266 with a brand new install of Micropython.

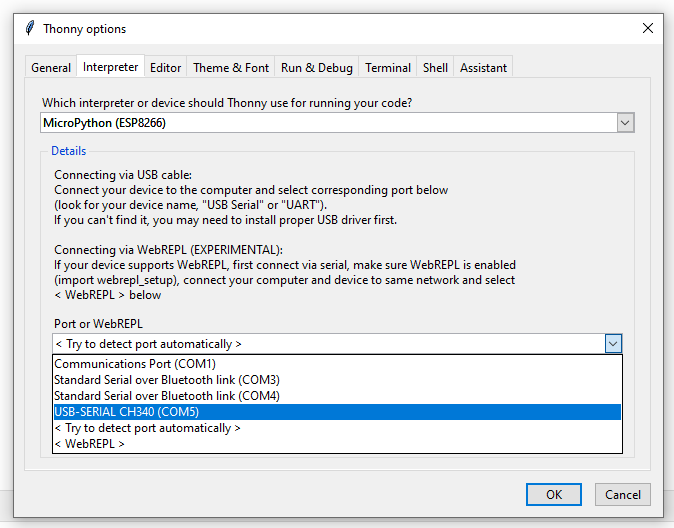
1. Open Thonny IDE and select the Tools tab and then options…



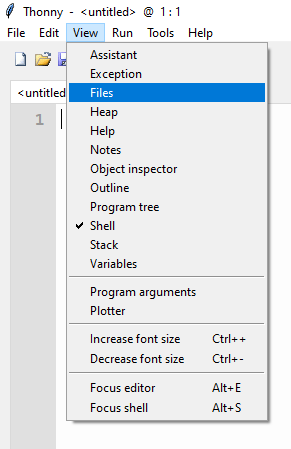
2.Under the Interpreter tab select MicroPython (ESP8266).



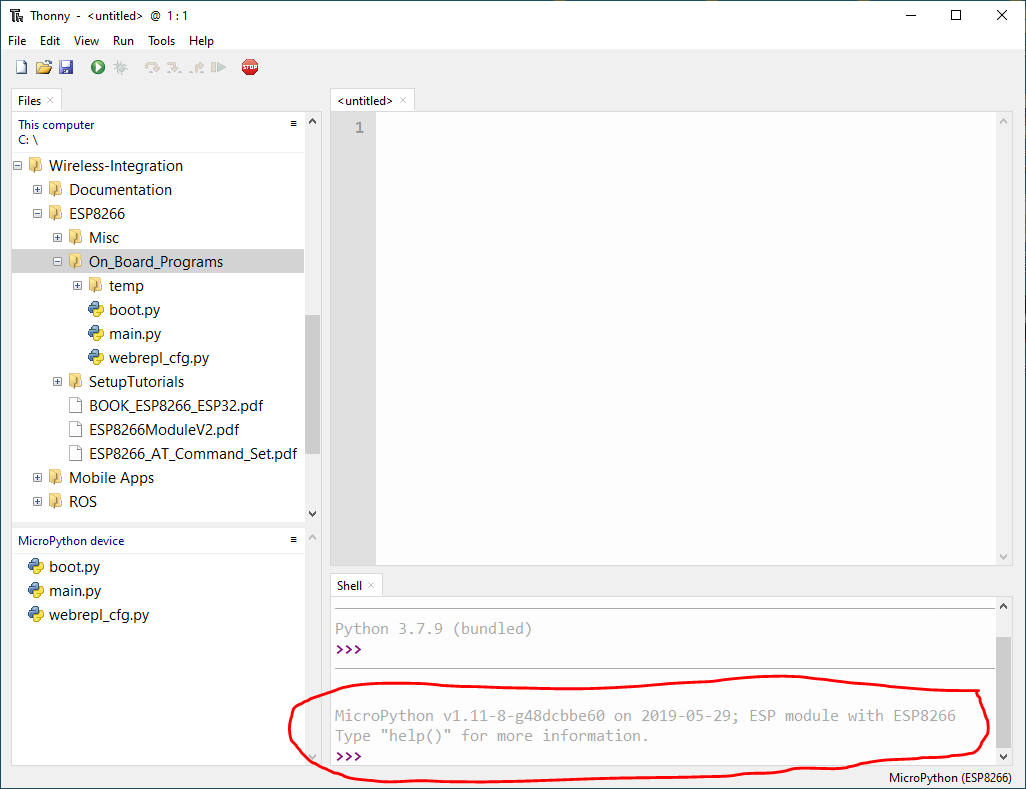
3. For now, select the COM port that corresponds to your ESP8266 Serial Connection, below it is COM5.



4. Under the View tab, select Files to open up a pane in the main screen which will show the ESP’s current files.

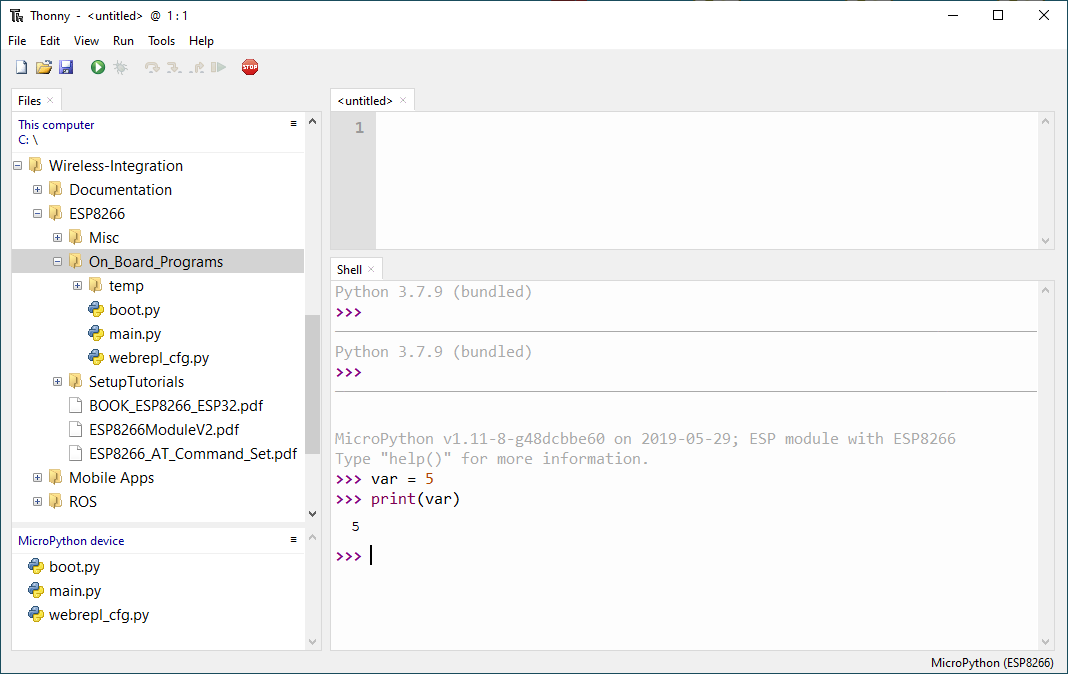


5. You should now connect your ESP8266 to your computer’s serial port, this can be done using an ESP8266 USB programmer or through any other UART to USB communication method. Your IDE should resemble the following:

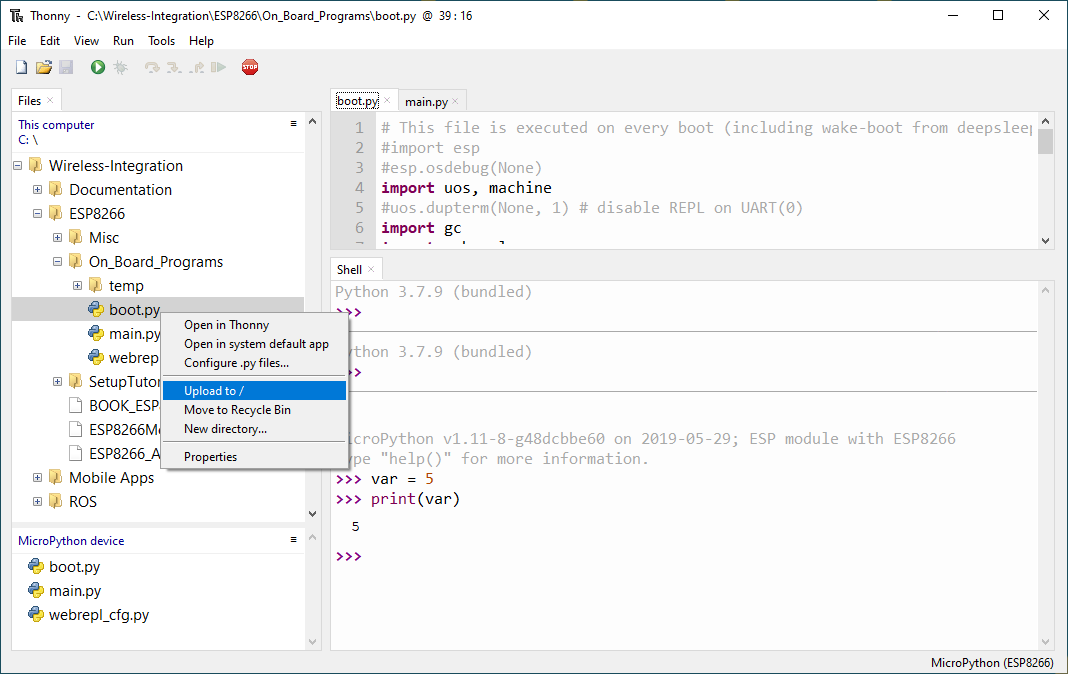


Look for the message at the bottom that shows a connection to an ESP8266 module has been made.

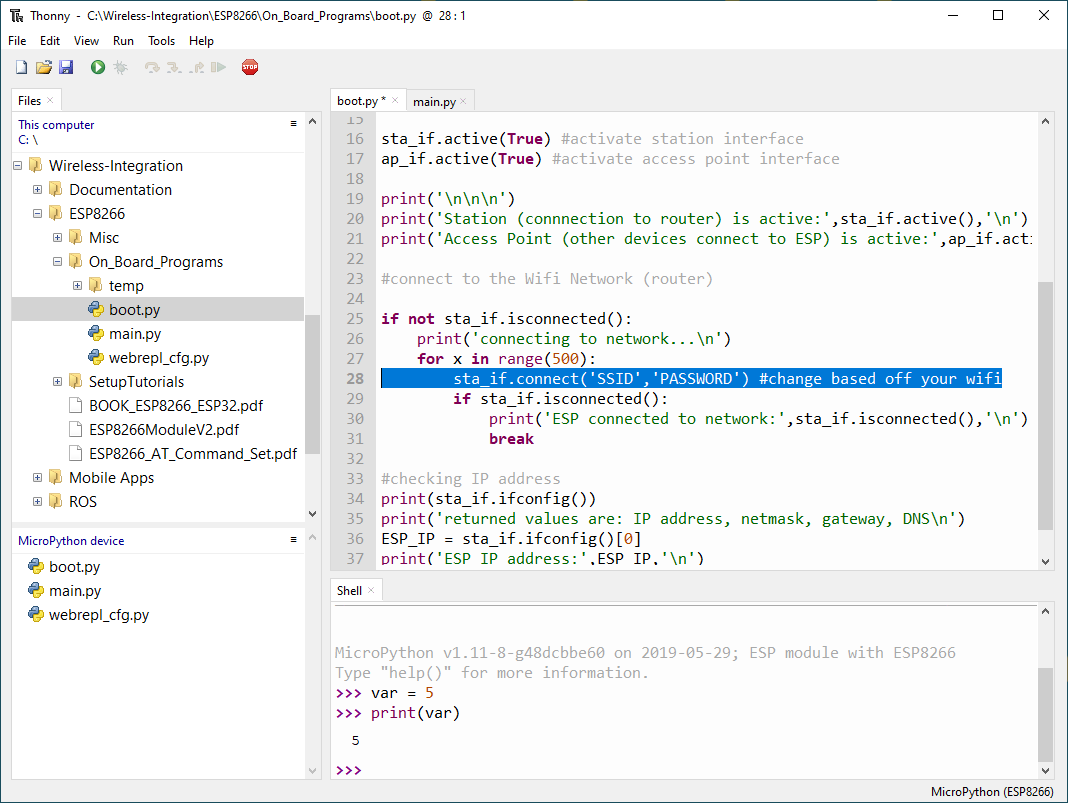
6. You are now free to use the shell to send commands to your ESP8266 module. Micropython is a subset of Python specialized for microcontrollers, so normal Python commands work fine.



7. In the same Wireless-Integration repo you can find our default board files under the folder ‘On\_Board\_Programs’. The ESP8266 will automatically run the ‘boot.py’ and ‘main.py’ programs on startup (for now do not worry about the ‘webrepl\_cfg.py’ file). You can right click on the files in the folder and select ‘Upload to/’ in order to send these files into your ESP8266. It is okay to override the existing files on the ESP8266.



8. These boot and main files are currently configured to automatically connect to WIFI, WEBREPL, and a Server. You will need to modify the boot file to include your WIFI network’s SSID and Password. Change the highlighted line to match your settings. Click the save icon in the top menu to re-upload the updated code onto your ESP8266.



9. Your ESP8266 is now configured to automatically connect to your WIFI network on startup. You are free to modify the boot to add other features, and your main can be programmed to suit your needs.

# Troubleshooting

* If you encounter issues while programming or your board seems to disappear, click the red STOP sign icon in the top menu bar. This will reset the connection to your ESP8266.
* If you need to stop a program use CTRL + C
* To reset your ESP8266 use the command: machine.reset() *(I recommend using the STOP Icon over this method)*

# WebREPL Setup

Your ESP8266 is capable of using a feature known as Web REPL. REPL stands for Read Evaluate Print Loop, and is the name given to the interactive MicroPython prompt that you can access on the ESP8266. Using the REPL is by far the easiest way to test out your code and run commands. Because you are able to access the REPL over Wifi, this allows you to modify and reprogram your ESP8266 while it is not connected to your computer. This is useful because you can avoid unplugging your ESP8266 from wherever it may be connected. All you need is to turn on your ESP8266, and have WebREPL configured.

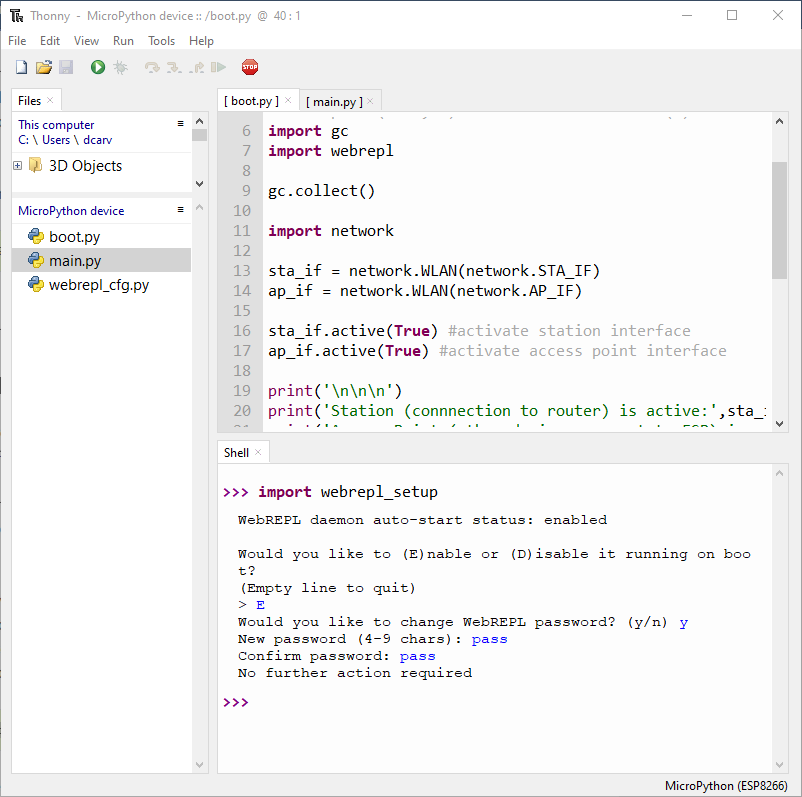
Luckily, the files provided in the above section are already configured to boot up WebREPL along with WIFI, but for your convenience, I will show you how to setup WebREPL from scratch. We will be using one of the Thonny IDE’s built in features that supports WebREPL, but other methods are available to access the REPL prompt. You can read more about these methods [here](http://docs.micropython.org/en/v1.15/esp8266/tutorial/repl.html).

1. First you need to enable WebREPL as part of your boot settings. To do this your ‘boot.py’ file must have the following lines of code:

> import webrepl

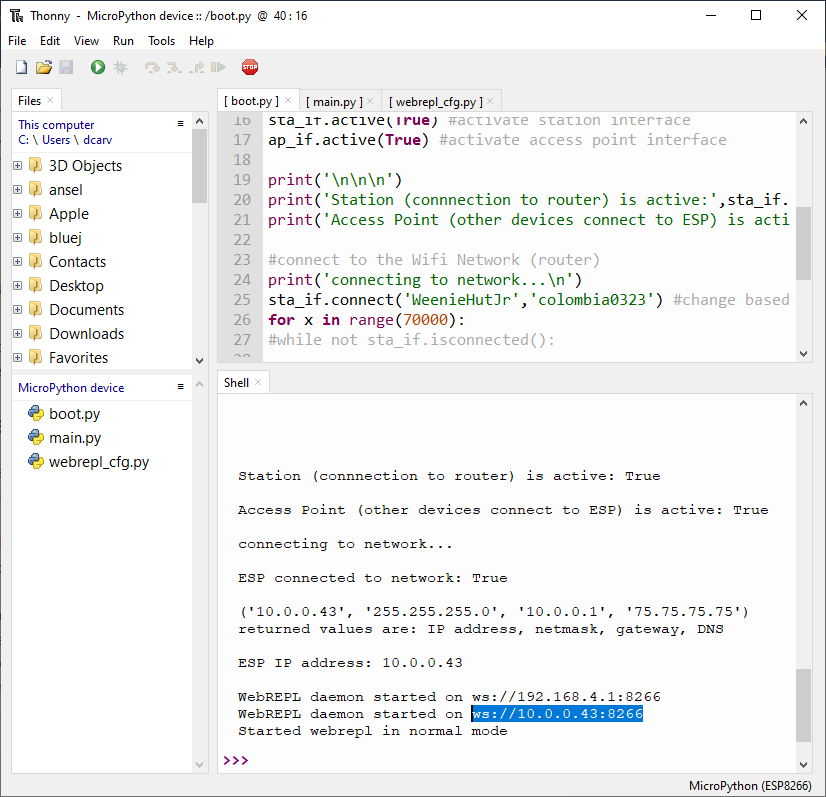
> webrepl.start()

2. There is also a one-time setup command you must run to initialize the WebREPL settings for the given ESP8266. The command is: > import webrepl\_setup



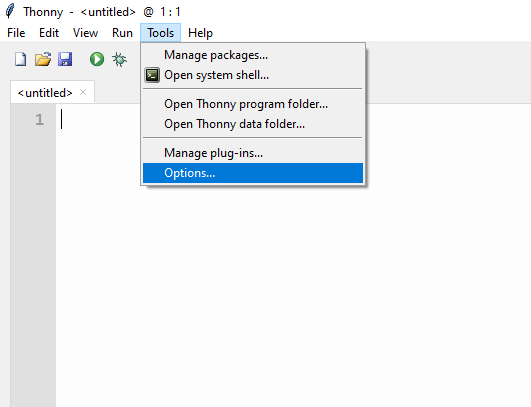
Follow the prompts to enable WebREPL on boot and set your own password. The password you choose will automatically be stored in a new file called ‘webrepl\_cfg.py’.

3. Now that WebREPL is enabled, rerun the ‘boot.py’ program and check the output messages to confirm WebREPL is starting up correctly. You should see output similar to the one below.

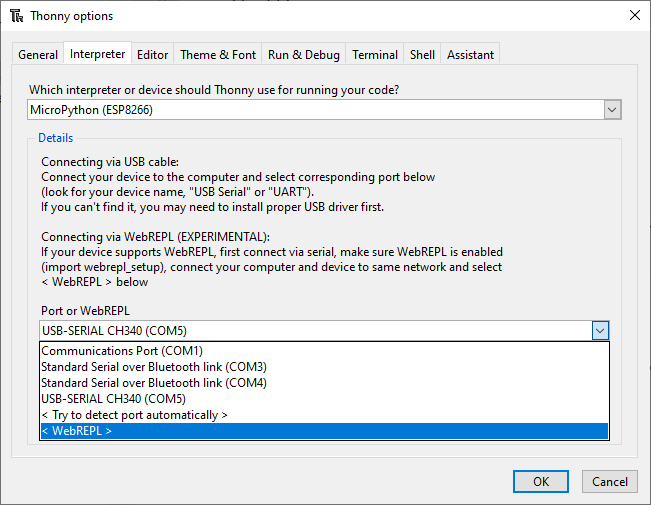


Take note of the IP address the WebREPL daemon started on (same as the ESP). This line is highlighted in the image above. We can also see that WebREPL started in normal mode, this means it is ready to communicate. Note: make sure that the ESP8266 and the computer you will be using to communicate with are both connected to the same Wifi network.

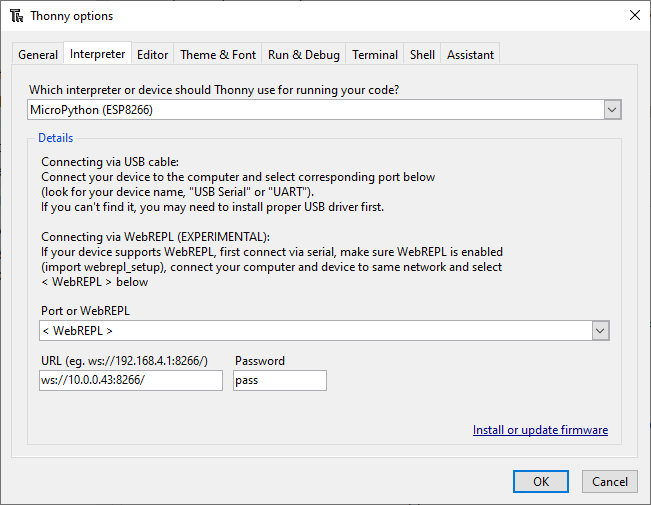
4. I will show you how to use a Thonny IDE feature to connect to the WebREPL prompt. First, open the tools tab again and select options.



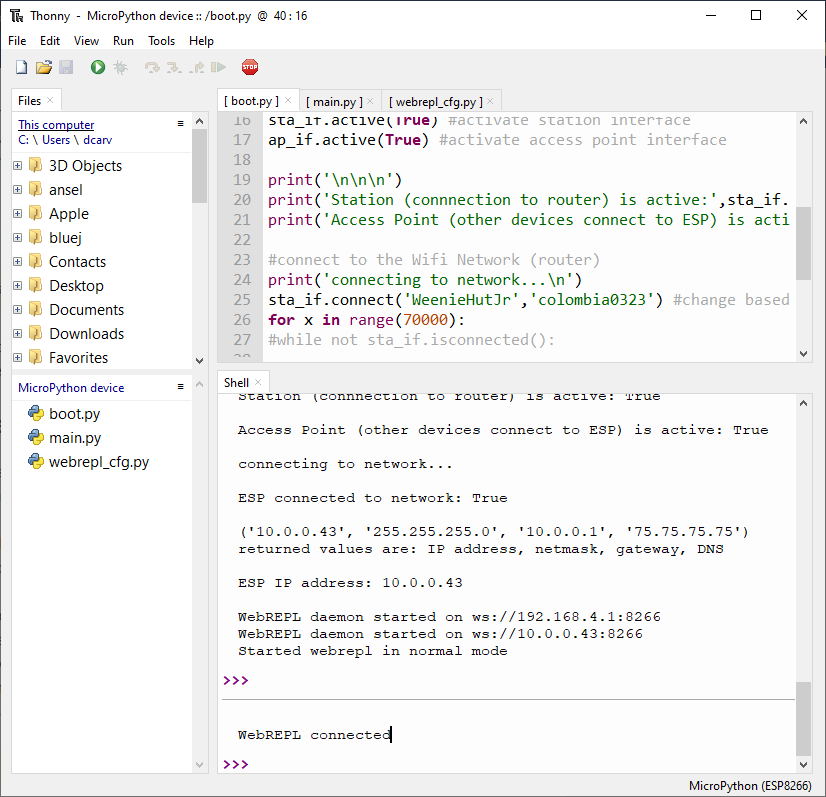
5. Under the Interpreter tab, choose the Port option for WebREPL



6. Under URL, enter the IP address the WebREPL daemon started on, and also enter the password you chose. Then press OK.



7. If the connection is successful, you will see ‘WebREPL connected’ as shown below.



8. If the message does not appear at first, press the stop sign icon to reset the ESP8266, this will usually fix any issues.

9. You are now free to program, access files, and run commands on the ESP8266 without any need for a serial connection. Apart from selecting WebREPL in your interpreter settings, you can use Thonny IDE exactly the same way as when you connect the ESP8266 through a serial port.

This concludes the guide on setting up communication with your ESP8266.