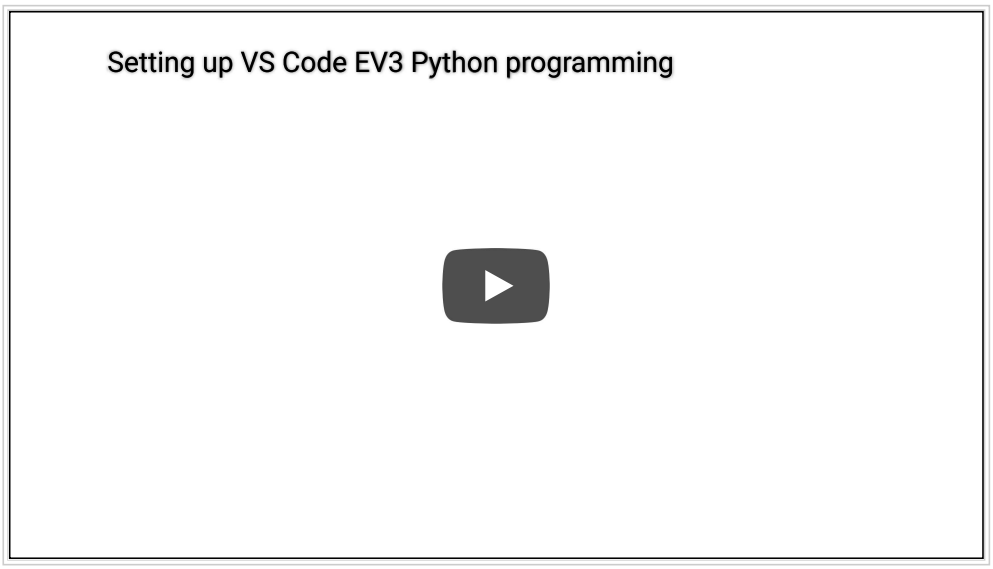
Setting up VS Code



You may prefer to **watch the video on YouTube**.

This is the script of the above 36 minute video:

This video explains how to set up your Lego EV3 robot and your computer so that you can begin using **Microsoft Visual Studio Code (VS Code)** to write Python scripts to control the EV3. The key is to use the **EV3 extension** which recently became available for VS Code. Using the extension, VS Code can download your script to the EV3 and run it there with the touch of a single key!

You should not watch this video until you have watched **the sister video** which shows how amazingly simple it can be to write and run EV3 Python scripts using VS Code, and you should not contemplate trying to write Python scripts for the EV3 unless you already have a good understanding of the basics of Python programming.

Writing Python scripts for the EV3 is *not* a good way to learn the basics of Python, but it is a good way to reinforce your Python skills while learning about robots.

Here's what you need to get started programming your EV3 with Python using VS Code.

You'll need to have Python installed on your computer, and I'll assume that's already the case since you shouldn't be writing scripts for the EV3 unless you already have a mastery of the basics of Python. You probably got Python from **https://www.python.org/downloads/** . You will certainly want to use Python 3, rather than Python 2 since Python 2 is not compatible with the VS Code EV3 Python programming workflow shown in this video.

These are the additional steps you will need to complete before you can use VS Code to write and run Python scripts for the EV3. In addition to following the instructions here you will also want to refer to ev3dev.org and my site ev3python.com.

Setting up in 12 easy steps

Total time could be less than 1 hour if all goes well!

1. Obtain a suitable microSD memory card.
2. Download the latest Linux Debian Stretch ev3dev image.
3. Download and install Etcher, a free utility that will allow you to flash the ev3dev image to the microSD card.
4. Use Etcher to flash the image to the card.

5. Insert the card into the EV3, boot the EV3, do some minor configuring and establish a connection to the computer via USB, WiFi, Bluetooth or Ethernet.
6. Download and install Microsoft Visual Studio Code (VS Code). This is a free multiplatform code editor, compatible with Windows, Mac OS and Linux.
7. Write and run some non-EV3 Python scripts.
8. Download and unzip the starter project.
9. Open VS Code, open the starter project folder and install two extensions.
10. Configure VS Code.
11. Connect VS Code to your EV3.
12. Write and run your first EV3 Python script!

Here are the steps in more detail:

1. Obtain a suitable microSD or microSDHC memory card

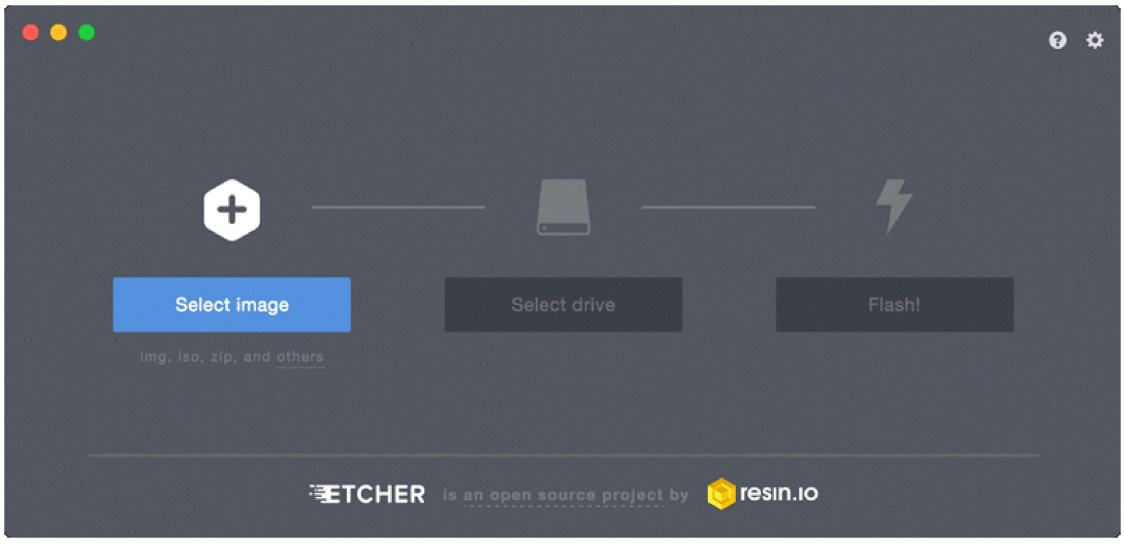
Obtain a **microSD** or **microSDHC** memory card with a capacity between 2GB and 32GB. **MicroSDXC** cards are not supported on the EV3. If you need to buy a card I suggest you buy an 8GB card from Amazon from a well-known manufacturer such as Sandisk or Kensington.

2. Download the latest Linux Debian Stretch ev3dev image

Note that the VS Code extension is compatible only with Stretch and not with Jessie versions of ev3dev. Currently, the Stretch images are not easy to find on ev3dev.org so go straight **HERE**: **Be sure to download the latest Stretch version and not a Jessie version**. Do not download the version offered prominently on ev3dev.org because that is a Jessie version and is not compatible with the VS Code Ev3 extension. Stretch versions are available for EV3, Raspberry Pi 1, Raspberry Pi 2 and Beaglebone, but only the EV3 is discussed in this video.The 27 July 2018 Stretch image seems to be 'broken' so please avoid that particular image. The download is about 250MB and may take a couple of minutes.

It would be a good idea to check every month or so whether a more recent image is available. If it is then using Etcher to flash the image to the SD card is easier and faster than trying to connect your EV3 to the internet and then updating the older operating system on the card – that's why this video does not explain how to connect your EV3 to the internet. If you use Etcher to flash a newer image to the card then any Python scripts on the card will be deleted but this should not be a problem because you have the same scripts on your computer and it's quick and easy to download them to the card again.

3. Download and install Etcher



Etcher is a free, open source, multi-platform (Windows, Mac OS, Linux) utility that will allow you to flash the ev3dev image to the microSD card. Download it from **https://etcher.io/** and install it on your computer.

4. Flash the image to the card

Insert the SD card in your computer's card reader (in the unlikely event that your computer does not have a card reader you will need to buy one). If you get a message that you need to format the card you can dismiss that for Etcher will format the card for you. Start Etcher, select the ev3dev Stretch image you just downloaded, navigate to your microSD card (be sure to select the correct card since **everything on the card will be erased during the flashing process**) and click 'Flash!'. The flashing and validation process may take a couple of minutes.

5. Insert the card into the EV3, boot the EV3, do some minor configuring and establish a connection to the computer

If the following instructions are not clear enough for you, please refer also to **ev3python.com** and ev3dev.org. With the EV3 turned off, insert the microSD card into the slot on the side of the EV3 marked 'SD'. The card is fairly easy to insert but more difficult to remove – when you want to remove the card you may find it helpful to use tweezers. Boot the EV3 by pressing the Enter (center) button for a few seconds. Note that the ev3dev operating system runs entirely off the microSD card – the EV3 firmware is not modified at all, so if you wish to return to using the standard Lego EV3 software all you need to do is power down the EV3, remove the card and reboot. When the boot process has finished, which takes a couple of minutes, the EV3's LED lights will glow green and the ev3dev interface called **Brickman** will be displayed on the EV3 display.

Brickman's main menu:

At the top of the Brickman interface is a header line where the EV3's IP address will be displayed, as well as indicators for WiFi or Bluetooth connections and an indication of the voltage of the EV3 battery. Under that are the main sections proposed by Brickman: File Browser, Device Browser, Wireless and Networks, Battery, Open Roberta Lab and About. What we need to do next is connect the EV3 to the computer via USB, WiFi, Bluetooth or Ethernet.

USB connection. This is perhaps the easiest. Simply connect a USB cable from your computer to the 'PC' socket on the EV3. You've probably done that before when you used the standard Lego EV software. The computer should then be connected to the EV3 no matter whether Brickman indicates 'Online', 'Connected' or 'Disconnected'. Even if Brickman indicates 'disconnected' that only means that there is no IPv4 connection to the computer - an IPv6 connection should automatically have been established between the EV3 and the PC. Note that this video is intended for EV3 Python beginners so I won't be explaining how to get the EV3 'online' i.e. connected to the internet via the computer. You really shouldn't need to connect the EV3 to the internet to use the VS Code workflow described in this video, and neither should you need to establish a Secure SHell or SSH connection between VS Code and the EV3. If you are a more advanced user and do want to use SSH then you should refer to **ev3python.com** and ev3dev.org.

Bluetooth connection. Bluetooth functionality is built in to the EV3 but if you are using a desktop computer you may need to purchase a Bluetooth dongle for the computer. For the setup procedure, refer to this page . Ignore the line at the bottom of that page that says you should connect via SSH – that shouldn't be necessary in our wonderful VS Code workflow.

On my Windows PC I find it trickier to establish a Bluetooth connection than a WiFi connection so I prefer to use a WiFi connection.

WiFi connection. To make a WiFi connection you will need to buy an adaptor if you do not already own one.

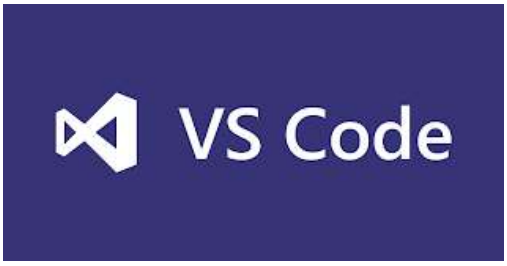
Unlike the standard Lego EV3 software which demands a very specific WiFi adaptor, the ev3dev software is compatible with a wide range of WiFi adaptors. Plug the adaptor into the EV3's USB socket, then, in the Brickman main menu, choose '**Wireless and Networks**'. Select '**Powered**' (press the Enter button) if it is not already selected. An indicator will come on in Brickman's header to indicate that the WiFi adaptor is powered. Then choose '**Start Scan**' and wait until available WiFi hotspots have been found. Choose the hotspot you wish to connect to then choose '**Connect**'. Press the Enter key to see the on-screen keyboard then enter your WiFi password. You can press the Back button if you make a mistake. Choose '**OK**' when you have finished, then choose 'Accept'. You should see '**Status: Connected**' or '**Status: Online**' – for our purposes it doesn't matter which you see. Now scroll down below the 'Forget' option and choose '**Network Connection**'. Turn on the option to '**Connect Automatically**'. You can move back up through the Brickman menu system by pressing the Back button.

Ethernet connection. If you want to use an Ethernet connection you must buy a USB to Ethernet adaptor such as this one.

You will plug the USB plug into the socket on the EV3 marked USB, and the Ethernet cable into your network hub. The nice thing about the Ethernet connection is that it should 'just work', with no need for any configuration.

Once you've established a connection from the EV3 to the computer, you've finished setting up the EV3. Now we'll install and set up Microsoft Visual Studio Code, and then we'll write and run our first EV3 Python script!

6. Download and install Microsoft Visual Studio Code



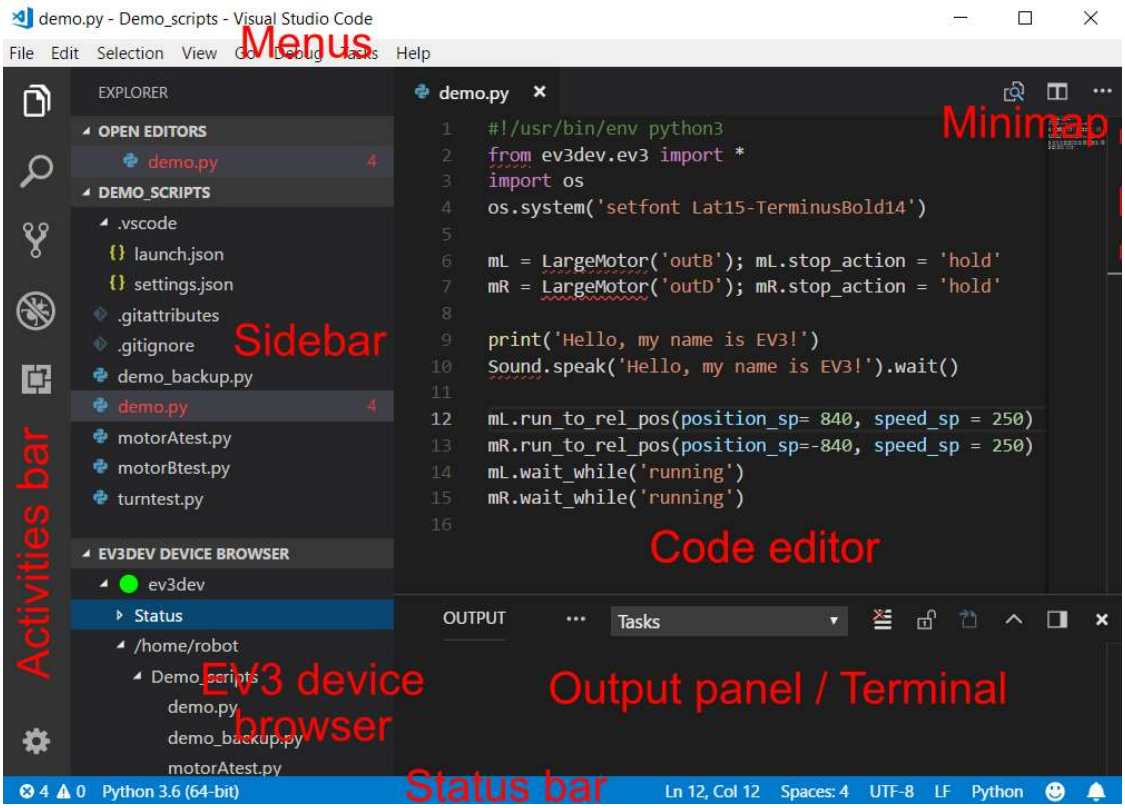
This is a free multiplatform code editor, compatible with Windows, Mac OS and Linux and not to be confused with Microsoft Visual Studio. Download and install Visual Studio Code (VS Code) from **https://code.visualstudio.com/Download**

For more help with getting started with VS Code, along with some helpful videos, see **https://code.visualstudio.com/docs** .

7. Write and run some non-EV3 Python scripts

These will run locally within VS Code. For non-EV3 Python scripts, you can either open a folder containing your script or scripts, or you can simply open a 'loose' Python script without opening the folder that contains it. However, since working with EV3 Python scripts requires that you open a folder containing your script or scripts it would be a good idea to have that habit even when you are working with non-EV3 scripts.

Start VS Code. You should recognise the various parts because you should have watched the **sister video** of this video before you started watching this video. If you haven't yet watched the **sister video**, be sure to watch it now before going any further.



Do **File> Open Folder**, navigate to a convenient location for a folder that will contain your non-EV3 Python scripts and click 'New Folder'. Name the new folder 'non-EV3 Python scripts' and click the '**Select Folder**' button. Assuming you are in Explorer view, the folder will open in the sidebar. The folder that holds your script or scripts is called the 'project folder'.

Do **File>New File** and before you even type any code save the file with the name **helloworld.py**. The reason to do that before typing the code is that as soon as VS Code recognises the extension for Python scripts it begins using helpful color coding. Type '**pr**' and notice how the Intellisense feature of VS Code proposes auto-completion to save you some typing. Choose the word you want and press Enter or Tab to confirm. After 'print' type an opening parenthesis and notice how Intellisense helps you in another way by giving you tips about the **print()** function. Finish typing the line so that it says **print('Hello, world')** and save your code again. Python is case-sensitive so, for example, **Print('Hello, world')** will not work. Be aware also that in Python 3 the **print()** function requires parentheses, which is not the case in Python 2. **Recall that you MUST use Python 3 and not Python 2 with the VS Code EV3 Python programming workflow**.

Run your script by right-clicking the script and choosing '**Run Python File in Terminal**'. The text will be printed to the terminal panel.

Let's make another non-EV3 Python script, this time one that uses the turtle module that hopefully is installed with your Python interpreter. This script will ask the user to input a number and then draw a polygon with that number of sides.

Do **File>New File**, save the file with the name **polygon.py** then type or copy/paste this script:

```
from time import sleep
import turtle
t = turtle.Pen()
sides=int(input('How many sides?'))
for n in range(sides):
    t.forward(100)
    t.left(360/sides)
sleep(6)
```

You can run your script by right-clicking the script and choosing '**Run Python File in Terminal**'. You may find that when you type the number into the terminal the graphics window disappears behind the VS Code window and that you cannot make it visible again. In that case make the VS Code window small and you should be able to see the polygon being drawn.

You can also run the script while in the Debug view. Before doing that it would be helpful to put a breakpoint on line 3 by clicking to the left of the line number such that a red dot appears. If you then choose **Debug>Start Debugging (or press the F5 key)** and choose '**Python**' then script execution will pause at line 3 and you can then repeatedly click the '**Step Over**' icon in the control bar to step through the script until it terminates.

Note that we have put two scripts in the same folder. That's fine for beginners and it makes it easy to copy and paste code between scripts, but advanced coders working on complex projects would not put unrelated files in the same project folder.

8. Download and unzip the starter EV3 python project

In order to be able to run your EV3 Python scripts on the EV3 from VS Code you MUST have open in VS Code a folder that contains your script and certain other files. The folder that holds your script or scripts is called the 'project folder'. As far as we are concerned as VS Code beginners, the project folder can also be called your 'workspace'. Don't mix EV3 Python scripts with non-EV3 Python scripts in the same folder because they need to be launched in different ways.

To get started, go to **https://github.com/ev3dev/vscode-hello-python** and follow the first step there to download the starter project called **vscode-hello-python**. That will actually download a zip file called **vscode-hello-python-master.zip**. Unzip that to a convenient location. In Windows, don't accept the proposed unzip location otherwise you will end up with one folder containing another one with exactly the same name. Instead, delete vscode-hello-python-master from the end of the address and also uncheck 'Show extracted files when complete. Unzipping will give you a folder called **vscode-hello-python-master** that contains various files and subfolders.

9. In VS Code, open the starter project folder and install the Microsoft Python extension and the EV3dev browser extension

In VS Code, do **File > Open Folder** and open the starter project folder called vscode-hello-python-master in VS Code. Click '**Show Recommendations**' when asked. Install the ev3dev-browser extension. You should not be following this tutorial unless you already have some basic knowledge of Python, so Python should already be installed on your computer. You can therefore go ahead and install the Python extension too. After installation completes, click '**Reload**' and '**Reload Window**'.

You should now see multiple files and folders listed as the contents of the vscode-hello-python-master folder. As a beginner in VS Code you will want to keep everything as simple as possible and I therefore suggest that you delete everything except the subfolder called **.vscode** and the Python script called **hello.py**. If you now open the subfolder called **.vscode** you will see that it contains four files: .gitignore, extensions.json, settings.json and launch.json. I suggest that you delete the files **.gitignore** and **extensions.json** since as a beginner you will not be using Git (an online source control system that is popular with more advanced users) and you no longer need the **extensions.json** file since you have just installed the two needed extensions.
Any other folder that you create containing EV3 Python scripts for use with VS Code should contain a copy of the .vscode folder that you have in the current project folder.

10. Configure VS Code

Let's take a look at the two remaining files in the **.vscode** folder.

Click on **settings.json** to open it in the code editor window. In fact two code editor windows will open. The left window shows the hundreds of **default settings** which govern the behaviour of VS Code. These default values cannot be changes but can be overridden by values that you provide. You can provide values either as '**User settings**' which will apply all the time that VS Code is running or as '**Workspace settings**' that are stored within the current folder and will only apply when that particular folder is open (I am assuming that as a beginner you will only ever one folder at a time, in which case the workspace is simply the open project folder). **User settings override default settings and workspace settings override user settings**. In the code editor on the right are the contents of the settings.json file and you can see that these are identified as 'workspace settings' since they belong to and apply to the open folder:

```
{
    "files.eol": "\n",
    "ev3devBrowser.download.exclude": "{**/.*,LICENSE,README.md}"
}
```

The first line specifies that lines of code are to end with a 'line feed (LF)' character, which is what EV3dev Python interpreter expects. ('eol' means 'end of line'.) If you type 'files.eol' into the search bar you will see that the default line ending is different, but the workspace setting overrides the default setting and gives us the line ending we need.

The second line specifies types of file that should NOT be downloaded to the EV3 when you later download and run your script. Everything in the project folder will be downloaded except for files excluded by this line.

Now let's try modifying a '**User setting**' – recall that 'User Settings' will apply every time you use VS Code, not just when you open a specific folder.

Click '**User Settings**' at the top of the right-hand code editor and you will see a number of user settings. We will add another one to override a so-called 'feature' of VS Code that most people find very annoying: the automatic closing of brackets. Carefully add this line "editor.autoClosingBrackets": false, …. No, wait! There is too much risk that you will do this wrongly, e.g. by missing the comma at the end. Recent versions of VS Code, such as the version I'm using (version 1.24.0 from June 2018) have a preview of a new settings editor which is much more foolproof and which will no doubt replace the older settings editor very soon. Click '**Try a preview of our new settings editor**' above the search box and make sure 'User Settings' is selected. Search for 'editor.autoc' and remove the checkmark from the 'Auto Closing Brackets' option. This change should be saved automatically.

Note that you can also open the settings, even if you do not have a settings file in the current folder, with **File>Preferences>Settings** or by clicking the gear icon in the bottom left of the activities bar.

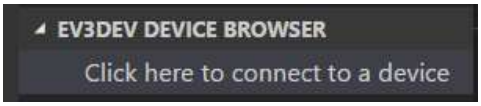
Now click the file **launch.json** file in the **.vscode** folder so that we can examine the contents of that file:

```
{
    "version": "0.2.0",
    "configurations": [
        {
            "name": "Download and Run",
            "type": "ev3devBrowser",
            "request": "launch",
            "program": "/home/robot/${workspaceRootFolderName}/hello.py"
        }
    ]
}
```

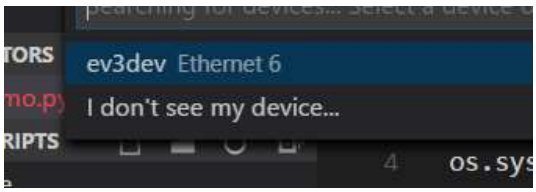
We can see that this file contains a single '**configuration**' called '**Download and Run**'. The line that we need to pay attention to is the one that mentions the Python script '**hello.py**'. This line determines which file will be run when we press the F5 key. If the line refers to a specific file then you will need to modify launch.json each time you want to use F5 to open a different file. We can modify the line so that F5 always downloads and runs the active file, whatever name that has. Carefully change '**hello.py**' to '**${relativeFile}**' so that the line now looks like this:
`"program": "/home/robot/${workspaceRootFolderName}/${relativeFile}"`
Then save the file.

11. Connect VS Code to your EV3

In a previous step you connected your EV3 to your computer, and now it's time to connect VS Code's ev3dev browser extension to the EV3. Make sure the Explorer view is chosen in the activities bar at left and you should see an 'EV3DEV DEVICE BROWSER' section at the bottom of the Explorer panel. Click where it says '**Click here to connect a device**' (you may need to click the header of the section in order to see those words).



You should then see at the top of the window the words '**I don't see my device**' and above that you should see a name that represents your EV3.



If there is such a name then click it and after a couple of seconds the connection to the EV3 should be established – a green dot will appear under the header of the EV3 Device Browser section. A red dot indicates a lost connection and an amber dot indicates that an attempt is being made to establish a connection.

If you do not see an EV3 name above the words 'I don't see my device' or if you are not able to connect to the named device then you have no choice but to click where is says 'I don't see my device'. You will then be asked to give a name to the connection you are trying to create – a name such as 'EV3 via USB' would be helpful. You will then be asked for the IP address of the EV3 which should be displayed at the top of the Brickman interface. When you enter information in this way your 'User settings' are modified. If you have tried to create a connection by inputting a name and IP address then you will see this has been incorporated into a User setting.

You can voluntarily disconnect VS Code from the EV3 by right-clicking the green dot and choosing 'Disconnect'. That right-click menu also gives you options of taking a screenshot, getting system info or opening an **SSH** (**Secure SHell**) terminal.

Note that VS Code will disconnect from the EV3 if you close the project folder or open a different folder.

If you are connected to the EV3 by a wireless connection (WiFi or Bluetooth) you may sometimes experience an involuntary disconnect. VS Code will not automatically try to reconnect if the connection is lost.

If you are disconnected from the EV3 and therefore see a red dot, then you can right-click the red dot and attempt to reconnect. You also have the option '**Connect to a different device**' which is rather misleading because this is also the option you would choose if you want to connect to the same device but via a different connection type (e.g. switching from USB to WiFi). Of course, connecting or reconnecting VS Code to the EV3 only works if there is a connection between the computer and the EV3.

12. Write and run your first EV3 Python script!

In a previous step you downloaded, unzipped, opened and modified a folder called vscode-hello-python-master that contains an EV3 Python script called **hello.py** and a subfolder called **.vscode**. The subfolder **.vscode** now contains two files that are needed in order to enable the running of EV3 Python scripts. If the folder **vscode-hello-python-master** is not still open then open it now. We are ready at last to try running an EV3 Python script!

The testing of a first EV3 Python script was demonstrated in the sister video – it would be a good idea to review that now. Click the script name **hello.py** in the folder listing in the sidebar and it will open in the code editor. We have carefully prepared the necessary helper files and established a connection from VS Code to the EV3 (the dot in the EV3 Device browser section should still be green) so launching the script now should be as easy as pressing the F5 key! It will usually take a few seconds for the script to begin running because the Python interpreter has to be started up on the EV3 each time. After a few seconds you should see the words 'Hello World!' appear on the LCD display of the EV3 in a large font and the words 'Hello VS Code!' should appear in the Output panel at the bottom of the VS Code window. The script should stop running on the EV3 after 5 seconds. If all went well for you then **CONGRATULATIONS**! You are well on the way to becoming a proficient EV3 Python programmer! If the script did not run correctly for you then please look back over the previous steps in the video and also consult **ev3python.com** and **ev3dev.org** for more help.

I am not going to analyse the code of this script line by line because it is not a typical EV3 script and because it is more complex than it needs to be. It is not typical because it does not make use of the **ev3dev.ev3 module** that is used by almost all EV3 scripts to enable the use of motors and sensors.

Let us therefore make a script that is shorter and more typical and which makes the motors attached to the EV3 work. Attach large motors to ports B and C of your EV3 and attach wheels to those motors. The script that we are about to make is the same as the script that I used in the sister video and it was analysed line by line in that video. It prints some text to the EV3's LCD screen in a large font, speaks some text and then attempts to make the robot rotate on the spot. Click **File>New File** and save the new file immediately into the vscode-hello-python-master folder with the name demo.py so that we can make use of color coding and Intellisense. Recall that it is fine to have up to about a dozen or so short EV3 Python scripts in the same project folder but no more than that because all the scripts in the folder are downloaded to the EV3 each time you run any script. Type or copy/paste the following code into the code editor then save the file.

```
#!/usr/bin/env python3
from ev3dev.ev3 import *
import os
os.system('setfont Lat15-TerminusBold14')
mL = LargeMotor('outB'); mL.stop_action = 'hold'
mR = LargeMotor('outC'); mR.stop_action = 'hold'
print('Hello, my name is EV3!')
Sound.speak('Hello, my name is EV3!').wait()
mL.run_to_rel_pos(position_sp= 840, speed_sp = 250)
mR.run_to_rel_pos(position_sp=-840, speed_sp = 250)
mL.wait_while('running')
mR.wait_while('running')
```

If this script fails to run properly when you press F5 then it may be because you did not type the code correctly - If you get an error message in the VS Code output panel then read the error message carefully - it will usually help you determine which line caused the script to fail.

Note that it is not possible to debug an EV3 by executing the script one line at a time as I demonstrated earlier for a non-EV3 script. That's because all we can do with EV3 scripts is simply

download them to the EV3 and launch them.

To stop an EV3 script that does not terminate by itself, press the Back button on the EV3. To stop a script from within VS Code, click the stop button in the control bar.

As an alternative to running EV3 scripts from VS Code by pressing F5, you can click the icon in the header of the EV3 Device Browser to download your script to the EV3 without running it, and then run it directly on the EV3 using the '**Brickman**' interface. In Brickman's main menu, choose 'File Browser' then navigate to the script you wish to run and press the Enter button.

In this video I have shown you how to prepare your EV3 for Ev3dev Python programming, how to set up VS Code on your computer and how to write and run EV3 Python scripts. Of course there is much more to say so I will release more videos about the wonderful VS Code EV3 Python programming workflow and will update my site **ev3python.com** regularly.

Before I go let me remind you of some of the reasons why the VS Code workflow is so superior to other ways of making and running EV3dev Python programs:

- In other workflows it is necessary to use Secure SHell (SSH) to mark each script as executable before it can be run. This is done automatically when you use the VS Code workflow.
- If you want to draw or write to the EV3's LCD screen then other workflows require that you use SSH to give control of the LCD screen to the script, otherwise the script fights with Brickman for control of the LCD. In the VS Code workflow control of the LCD is automatically given to the script while it is running, so no need to use SSH.
- In other workflows the motors will sometimes continue to run after a script has ended. This does not happen if you use the VS Code workflow.

Commenti

Non disponi dell'autorizzazione necessaria per aggiungere commenti.