Lego Spike Prime and Python

Desires

- Use python to program a Spike Prime robot, specifically for advanced FLL teams
- Use advanced IDE features like intellisense, code completion, classes, code sharing (git/github), debugging, etc
- We’d like a way to have some common code that we can share with all team members. Things like gyroturn and gyrodrive. We’d also like the common code to take care of the “wiring” of the robot (what is wired into each port). Spike Python requires setting up the ports for the drive motor pairs and we’d like to eliminate possible errors by doing this once in the common code.
- We also need a way to get all programs onto a single competition robot for tournaments.

Our Team Setup

We have several base robots. Each one is carefully built and wired exactly the same as the other base robots. Each kid on the team will have a robot and a laptop for programming.

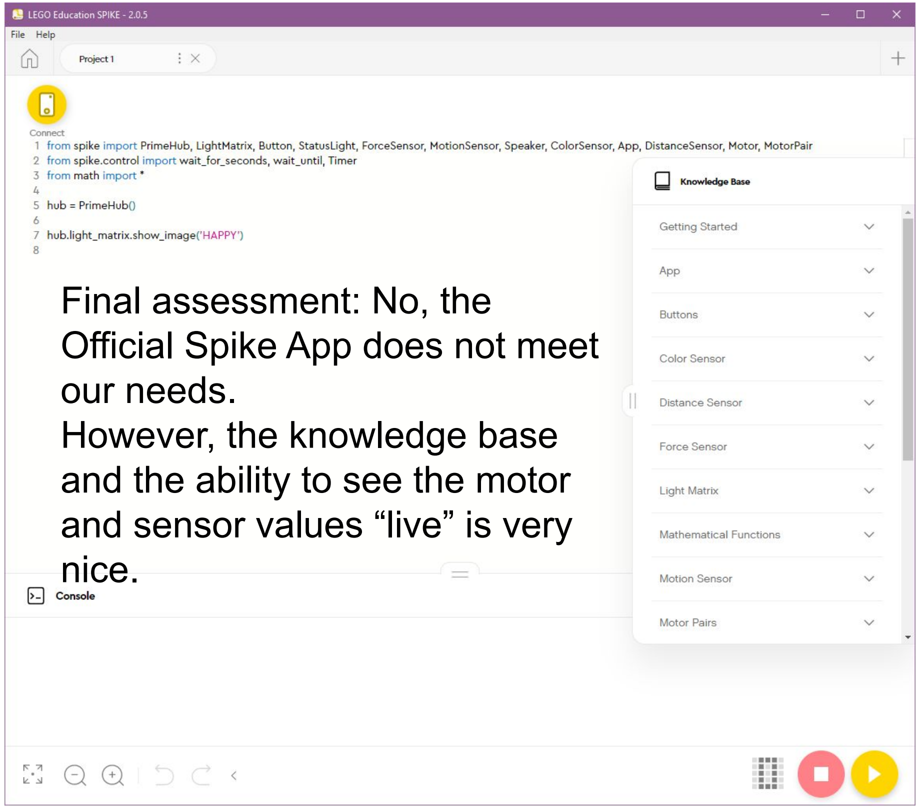
Each kid on the team is responsible for building the mechanical attachment/solution for whatever mission they are working on AND the programming for that mission.

Ultimately we will need to get all programs on to the “competition” robot, and we normally use some sort of master program to control the running of the mission programs.

This way any shared code between robots should run without any worries of one robot being wired differently.

Does the Official Spike App Meet our Needs?

- For sure we can use Python in the Spike app to program the hub/robot, but what about the other "normal" IDE features?
- Intellisense/Code completion - No
- Code Sharing - No (just copy/paste)
- Classes - Yes, but all in one file
- However, the ability to see the motors and sensors "live" is nice–We'll use that
- Debugging (checkpoints, variable watches, etc) - No



```
from spike import PrimeHub, LightMatrix, Button, StatusLight, ForceSensor, MotionSensor, Speaker, ColorSensor, App, DistanceSensor, Motor, MotorPair
from spike.control import wait_for_seconds, wait_until, Timer
from math import *

hub = PrimeHub()

hub.light_matrix.show_image('HAPPY')
```

Final assessment: No, the Official Spike App does not meet our needs.
However, the knowledge base and the ability to see the motor and sensor values "live" is very nice.

What Could be Better?

- We can use VS Code to program the robot. This will give us (almost) all of the normal IDE features we are looking for
- We can use a very nice Lego Spike Prime plugin for VS code to manage the connection to the hub
- We can use a program called ampy to upload our custom python modules to the hub. This will give us the ability to use a Python class file with custom methods (similar to myBlocks from the EV3)
- We can still use the Spike app to view the live status of the motors and sensors (and to rename the hub–VERY IMPORTANT for middle school kids!)
- Unfortunately, there isn’t a way to debug a micropython program from another computer, which is a limitation for all micropython controllers.
- Final assessment: VS Code will meet almost all of our needs!

Install the Software

- **Spike App** (https://education.lego.com/en-us/downloads/spike-app/software)
- **Python** (technically not needed, but it includes pip, which is needed to install ampy) (https://www.python.org/downloads/)
- **VS Code** (https://code.visualstudio.com/download)
- **Spike plugin for VS Code** (https://marketplace.visualstudio.com/items?itemName=PeterStaev.lego-spikeprime-mindstorms-vscode)
- **ampy** (https://learn.adafruit.com/micropython-basics-load-files-and-run-code/install-ampy)
- **git** (https://git-scm.com/download/win)

Micropython vs Python

MicroPython is a programming language largely compatible with Python 3. When you write a python program in the Spike app, you run it by uploading it to the hub, and the python interpreter on the hub executes the program. That’s why technically we don’t need to install Python on the computer before programming the hub–the hub doesn’t use the python installed on the computer.

What this also means is that if you want to “import” a special library, just having that library on your computer isn’t good enough. It has to be on the hub “at run time”. When your program is running, all of the imports have to be on the hub. The imports for spike are already on the hub. It’s any **other** imports you will need to ensure are in place.

Imports

A typical python (and micropython) program may start with a couple of import lines. Spike programs will normally have imports such as

```
import base_robot
import sys
from spike.control import wait_for_seconds, wait_until, Timer
from spike.operator import greater_than, greater_than_or_equal_to, \
    less_than, less_than_or_equal_to, equal_to, not_equal_to
```

The imports for sys, spike.control and spike.operator are already on the hub. But in this case, we have also written a custom module named base_robot.py (the first import listed above) and we have manually uploaded that file to the hub using ampy, so all of these imports will work on the hub.

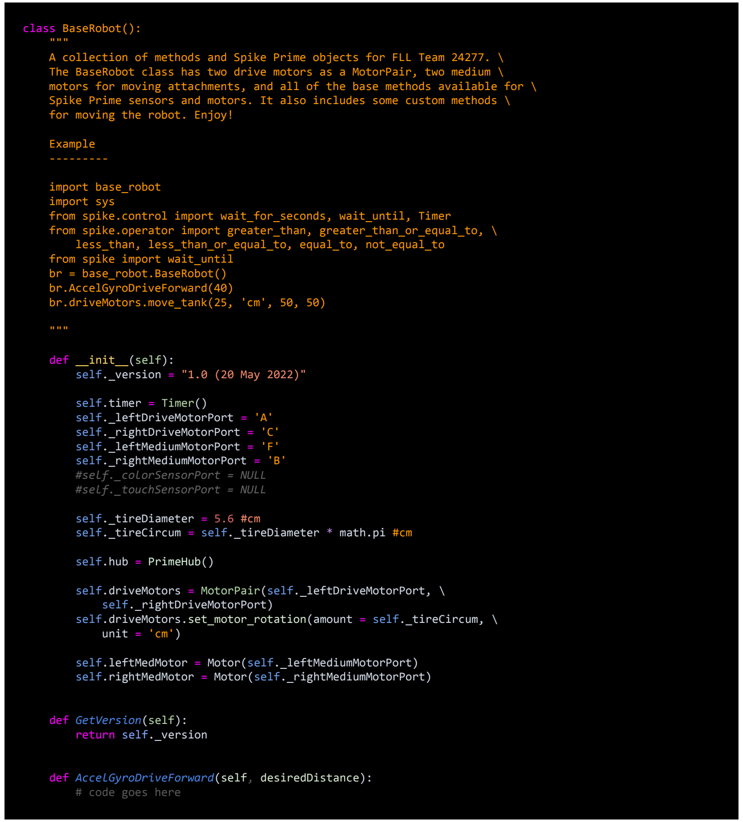
Custom Modules

In Python, a file is a module. A module can contain custom classes, and all of the things that go with normal object-oriented classes such as methods/procedures (sometimes called functions, which are similar to MyBlocks from the EV3) and other objects as needed.

For FLL, a common EV3 MyBlock might be a gyro-drive. This MyBlock simplifies the coding and allows that code to be reused, enabling the robot to drive straighter using feedback from the gyro. This way the gyro-drive code only has to be written once, and it can be shared with other team members.

On the Spike hub, we will do this with a custom python module. It only needs to be written once and then shared with all the other team members.

How We Do It: base_robot.py



```python
class BaseRobot():
    """
    A collection of methods and Spike Prime objects for FLL Team 24277. \
    The BaseRobot class has two drive motors as a MotorPair, two medium \
    motors for moving attachments, and all of the base methods available for \
    Spike Prime sensors and motors. It also includes some custom methods \
    for moving the robot. Enjoy!

    Example
    ---------

    import base_robot
    import sys
    from spike.control import wait_for_seconds, wait_until, Timer
    from spike.operator import greater_than, greater_than_or_equal_to, \
        less_than, less_than_or_equal_to, equal_to, not_equal_to
    from spike import wait_until
    br = base_robot.BaseRobot()
    br.AccelGyroDriveForward(40)
    br.driveMotors.move_tank(25, 'cm', 50, 50)

    """

    def __init__(self):
        self._version = "1.0 (20 May 2022)"

        self.timer = Timer()
        self._leftDriveMotorPort = 'A'
        self._rightDriveMotorPort = 'C'
        self._leftMediumMotorPort = 'F'
        self._rightMediumMotorPort = 'B'
        #self._colorSensorPort = NULL
        #self._touchSensorPort = NULL

        self._tireDiameter = 5.6 #cm
        self._tireCircum = self._tireDiameter * math.pi #cm

        self.hub = PrimeHub()

        self.driveMotors = MotorPair(self._leftDriveMotorPort, \
            self._rightDriveMotorPort)
        self.driveMotors.set_motor_rotation(amount = self._tireCircum, \
            unit = 'cm')

        self.leftMedMotor = Motor(self._leftMediumMotorPort)
        self.rightMedMotor = Motor(self._rightMediumMotorPort)


    def GetVersion(self):
        return self._version


    def AccelGyroDriveForward(self, desiredDistance):
        # code goes here
```

This is a snippet of our base_robot.py module and BaseRobot class. It's not very unusual in that it looks just like any other python class, so if you understand python classes, you should be fine.

The most important thing to understand is that nothing in this files runs directly. This file only works when it is called by another file (that is, the mission file which will be written by the team members).

When you are ready, you will need to upload any custom modules (files) to your hub using ampy.

This file is created by one or more advanced programmers on the team, and then shared with all of the other team members (by uploading it to their robot with ampy).

ampy

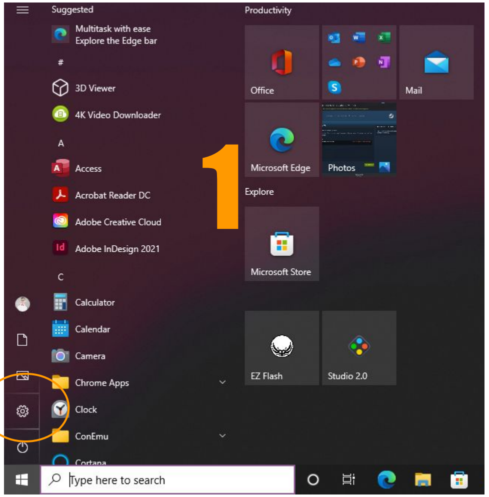
Adafruit created ampy to upload files to micropython controllers. To use it, you will need to know what COM port your computer uses to communicate over bluetooth (or USB) with your hub.

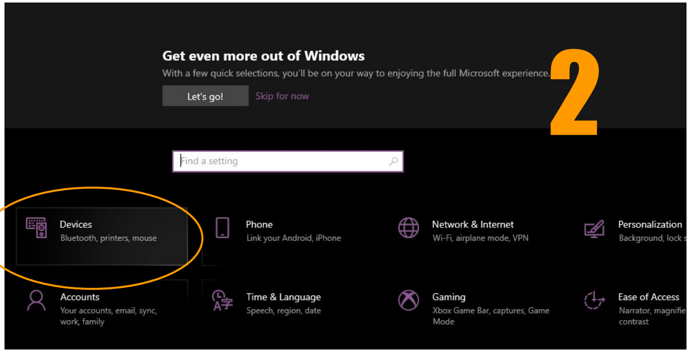
1. Using the Spike app, connect your hub to your computer over Bluetooth. If any updates are ready to install, go ahead and do that. Name your hub something memorable. Then disconnect from the Spike app.
2. Under Windows settings, go to Bluetooth and then click on “More BlueTooth Options”. Click on the COM Ports tab and look for your hub. You will see two listed: one for incoming and one for outgoing. You need the outgoing port.
3. Use ampy like this (open a command window):

   ```
   ampy -p COM5 put c:/users/myuser/Documents/base_robot.py /base_robot.py
   ```

4. Note: if you ever want to rename your hub, you will need to go into the Windows Bluetooth settings and remove the device first. Then go to the Spike app, connect, and rename your hub.
5. If your team has more than one hub, you will need to upload the base_robot.py to each of them.
6. Note: using ampy to upload files is different than uploading a program into a specific slot on the hub. There is no way to interact directly (using the hub) with files uploaded via ampy

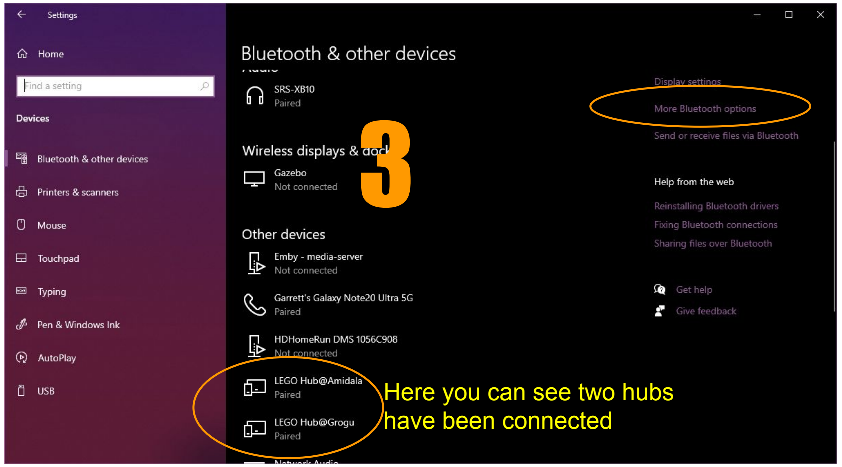
Bluetooth Spike Primes on Windows



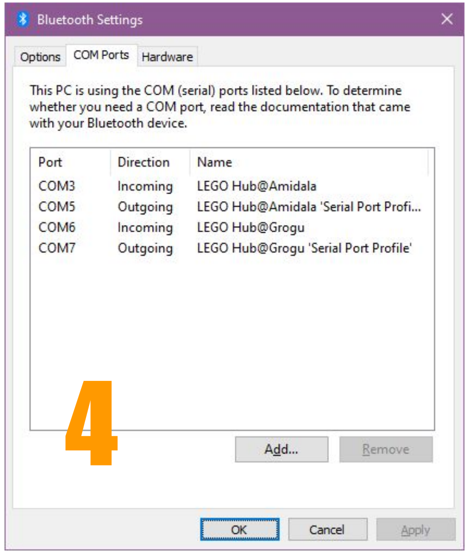

Get even more out of Windows

With a few quick selections, you'll be on your way to enjoying the full Microsoft experience.

Let's go!
Skip for now
Find a setting
Devices
Bluetooth, printers, mouse
Phone
Link your Android, iPhone
Network & Internet
Wi-Fi, airplane mode, VPN
Personalization
Background, lock s
Accounts
Your accounts, email, sync, work, family
Time & Language
Speech, region, date
Gaming
Xbox Game Bar, captures, Game Mode
Ease of Access
Narrator, magnifie contrast
2



Here you can see two hubs have been connected



This PC is using the COM (serial) ports listed below. To determine whether you need a COM port, read the documentation that came with your Bluetooth device.

Port	Direction	Name
COM3	Incoming	LEGO Hub@Amidala
COM5	Outgoing	LEGO Hub@Amidala 'Serial Port Profi...
COM6	Incoming	LEGO Hub@Grogu
COM7	Outgoing	LEGO Hub@Grogu 'Serial Port Profile'

Here you can see two hubs have outgoing COM ports. Amidala is on COM5 and Grogu is on COM7.

Note that if another computer connects to a hub, it may be on a different COM port.

BaseRobot class or not?

You don’t have to use a BaseRobot class. We use one since it is a nice container to hold all of the common programs that we like to share with the team. It also gives the advanced programmers a chance to learn more about Object Oriented Programming.

If you never feel the need to share code, and each team member will be responsible for all of their own code, then do not use a BaseRobot class. In that case you might still want to set up the folders with the python files to help VS code, but you don’t need to include the base_robot.py file.

Setting up a Folder for “Regular Team Member” Files

These are the instructions for individual team members that are writing code to solve a mission.

1. Make an empty folder on your computer. Perhaps something like “FLL_Fall_2022”. Putting this folder inside of your documents folder would be fine. I use a folder under Documents titled “FLL” and inside that I have a folder for each year (I’ve been doing this for eight years now).
2. Get the other .py files from https://github.com/FLL-Team-24277/BaseRobotFall2022 and copy them to your empty folder created in step 1. If you know how git works, you can clone that repository, or you can download the zip file and manually put them in place.
3. You shouldn’t edit any of the other spike files. The other .py files are there as helpers for VS code’s intellisense and code completion. Even the base_robot.py file is there just to help VS code. The file you uploaded to the hub is the one that will be used by the hub. By the way, you *could* just copy my base_robot.py file and use that, but where’s the fun in that? Please have your team create their own file and upload it to the hubs.
4. Your folder should look like this

```
Documents/FLL/FLL_Fall_2022/
├── spike.py *
├── spike/
│   ├── control.py *
│   └── operator.py *
├── base_robot.py *
├── main.py
├── mission1.py
├── AnyOtherProgramsYouWrite.py
```

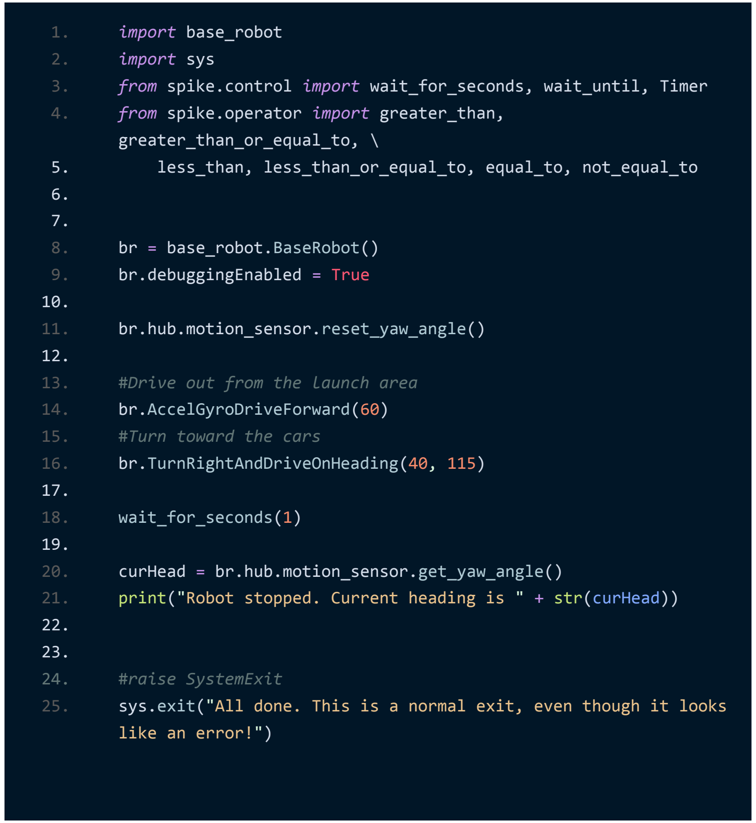
Do not edit or delete any of the files with asterisks!
You may edit (or delete) the files with the green font, and you can add other mission programs as needed. Just be sure to put the import statements at the top of each new file you create.
If your team is not using a BaseRobot class, then you should include the other files with asterisks, but you do not need the base_robot.py file.

Set up folders like this for each member of the team and/or each computer. Even though a team member may only create and edit one mission file all season, they should keep the files with the asterisks in their folder to help VS Code.

Programming Summary

1. Create a BaseRobot class that has all of the methods you want for your base robot. Upload it to all of your hubs. You only have to do this once (or not at all if you don't want to use a BaseRobot class). However, if (when!) you decide you need to make a change to the BaseRobot class, then you will have to re-upload it to all of the hubs.
2. Each team member will create their own folder/file for writing mission programs. They can call the methods in the BaseRobot class if there is one. They should include all of the Spike files and the base_robot.py file in their folder, and they shouldn't edit any of them.

Writing Mission Programs



```
1.      import base_robot
2.      import sys
3.      from spike.control import wait_for_seconds, wait_until, Timer
4.      from spike.operator import greater_than,
        greater_than_or_equal_to, \
5.          less_than, less_than_or_equal_to, equal_to, not_equal_to
6.
7.
8.      br = base_robot.BaseRobot()
9.      br.debuggingEnabled = True
10.
11.     br.hub.motion_sensor.reset_yaw_angle()
12.
13.     #Drive out from the launch area
14.     br.AccelGyroDriveForward(60)
15.     #Turn toward the cars
16.     br.TurnRightAndDriveOnHeading(40, 115)
17.
18.     wait_for_seconds(1)
19.
20.     curHead = br.hub.motion_sensor.get_yaw_angle()
21.     print("Robot stopped. Current heading is " + str(curHead))
22.
23.
24.     #raise SystemExit
25.     sys.exit("All done. This is a normal exit, even though it looks
        like an error!")
```

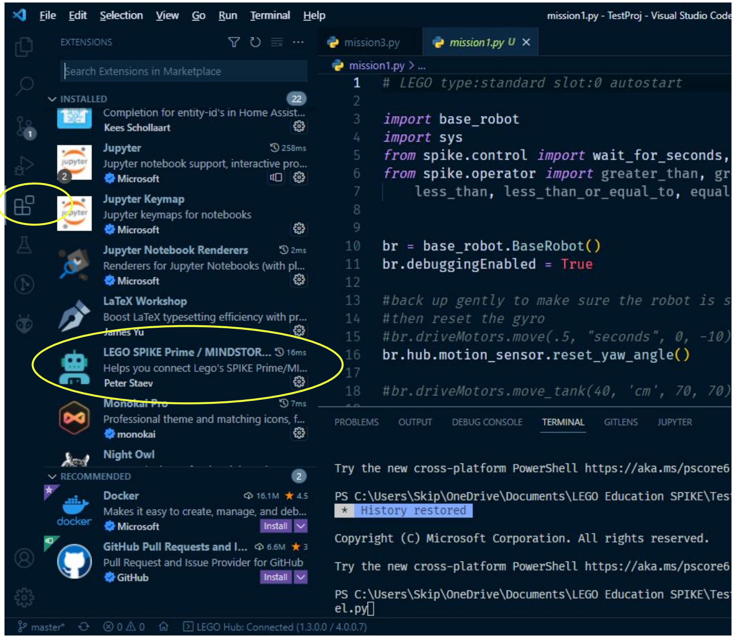
This could be a typical mission program. The robot drives out and does some things. Each team member will need to write a program like this to run their mission.

Lines 1-8 will be the same for all of the mission programs each member writes. Then, every command that you see that starts with "`br.`" is doing something from the class file. Line 9, we have set "`debuggingEnabled`" to "`True`" so we get more output on the screen when the program is running, which is a feature we added to our BaseRobot class. Line 15 we call our custom `AccelGyroDriveForward` program. Line 18 (`wait_for_seconds(1)`) does not start with "`br`" so it isn't using any custom class file code. Same for lines 20 & 21.

Line 25 can be included or not. When python programs run on the hub, when they end, they don't always "completely" end, and the hub still thinks it is running. You can hit the middle hub button to end it, or you can "really" end it automatically this way. It looks messy and prints what looks like an error message, but at least we don't have to remember to hit that button each time. Up to you if you want to do it like this.

All of the lines that start with # are comments and are ignored by the python interpreter. Remember to write good comments!

VS Code Spike Prime Plugin


File Edit Selection View Go Run Terminal Help
mission1.py - TestProj - Visual Studio Code
EXTENSIONS
Search Extensions in Marketplace
INSTALLED 22
Completion for entity-id's in Home Assist... Kees Schollaart
Jupyter 258ms Jupyter notebook support, interactive pro... Microsoft
Jupyter Keymap Jupyter keymaps for notebooks Microsoft
Jupyter Notebook Renderers 2ms Renderers for Jupyter Notebooks (with pl... Microsoft
LaTeX Workshop Boost LaTeX typesetting efficiency with pr... James Yu
LEGO SPIKE Prime / MINDSTOR... 16ms Helps you connect Lego's SPIKE Prime/MI... Peter Staev
Monokai Pro 7ms Professional theme and matching icons, f... monokai
Night Owl
RECOMMENDED 2
Docker 16.1M 4.5 Makes it easy to create, manage, and deb... Microsoft Install
GitHub Pull Requests and I... 6.6M 3 Pull Request and Issue Provider for GitHub GitHub Install
mission3.py
mission1.py U
mission1.py > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL GITLENS JUPYTER
master* 0 0 LEGO Hub: Connected (1.3.0.0 / 4.0.0.7)

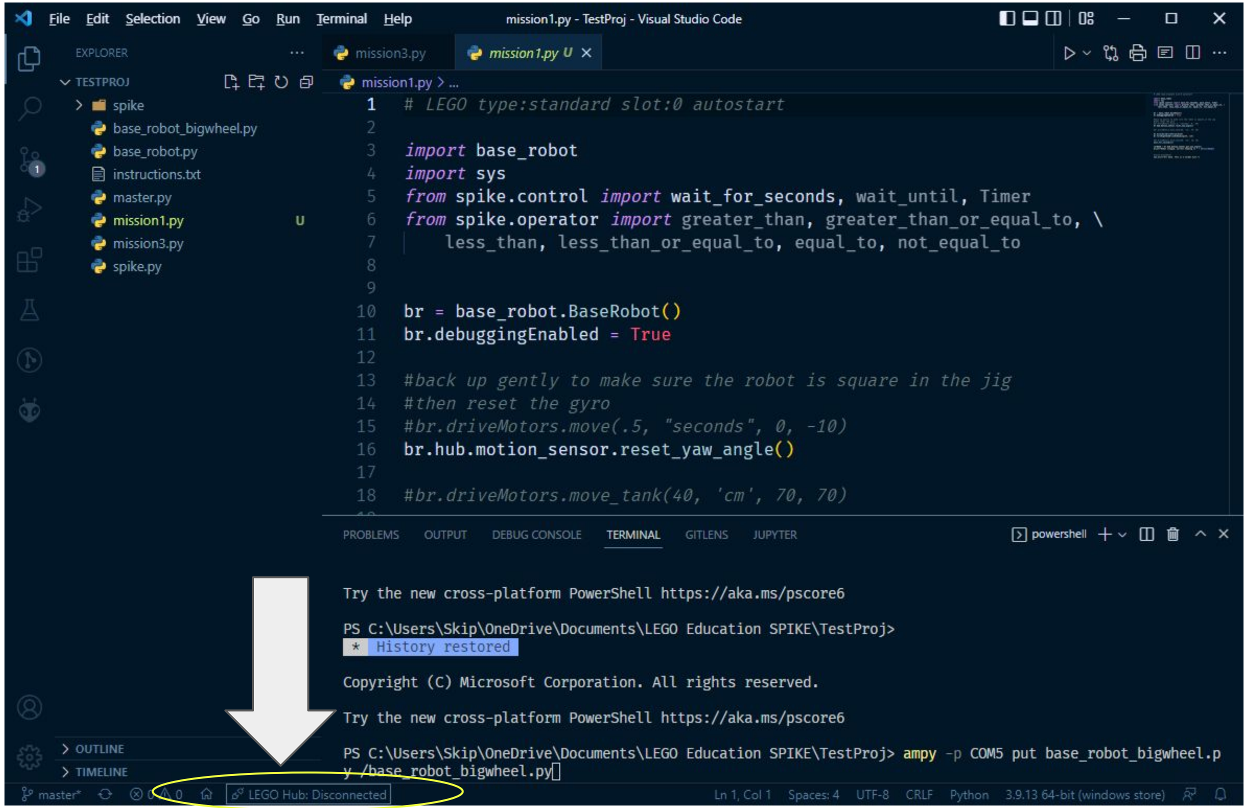
```python
# LEGO type:standard slot:0 autostart

import base_robot
import sys
from spike.control import wait_for_seconds,
from spike.operator import greater_than, gr
    less_than, less_than_or_equal_to, equal


br = base_robot.BaseRobot()
br.debuggingEnabled = True

#back up gently to make sure the robot is s
#then reset the gyro
#br.driveMotors.move(.5, "seconds", 0, -10)
br.hub.motion_sensor.reset_yaw_angle()

#br.driveMotors.move_tank(40, 'cm', 70, 70)
```

```
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Skip\OneDrive\Documents\LEGO Education SPIKE\Tes
 *  History restored

Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Skip\OneDrive\Documents\LEGO Education SPIKE\Tes
el.py
```

In VS Code, click on the “Extensions” icon on the left side, and install the LEGO Spike Prime/Mindstorms plugin (by Peter Staev). This will give you the ability to run your Spike programs right from VS Code.

Sorry, but there isn’t a way to do real debugging on a computer connected to a micropython board.

Connecting the hub to VS Code

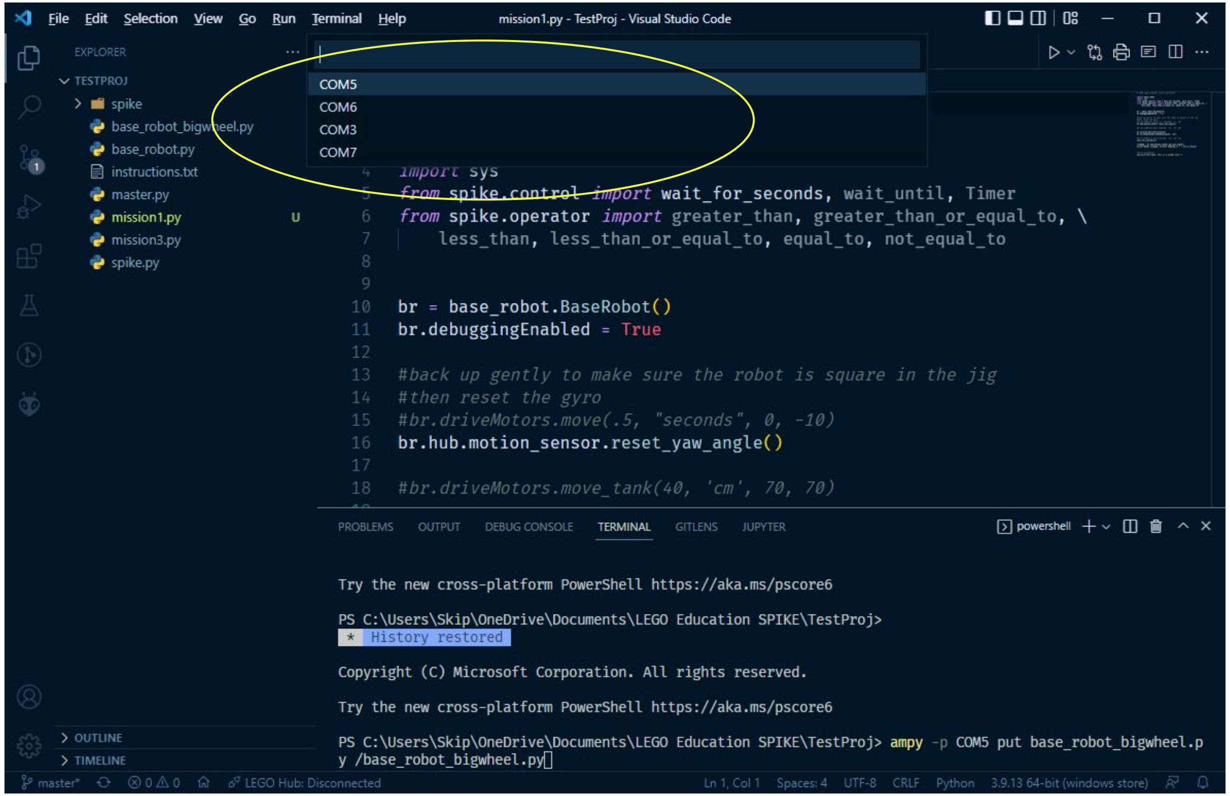


```python
# LEGO type:standard slot:0 autostart

import base_robot
import sys
from spike.control import wait_for_seconds, wait_until, Timer
from spike.operator import greater_than, greater_than_or_equal_to, \
    less_than, less_than_or_equal_to, equal_to, not_equal_to


br = base_robot.BaseRobot()
br.debuggingEnabled = True

#back up gently to make sure the robot is square in the jig
#then reset the gyro
#br.driveMotors.move(.5, "seconds", 0, -10)
br.hub.motion_sensor.reset_yaw_angle()

#br.driveMotors.move_tank(40, 'cm', 70, 70)
```

```
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Skip\OneDrive\Documents\LEGO Education SPIKE\TestProj>
 *  History restored

Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Skip\OneDrive\Documents\LEGO Education SPIKE\TestProj> ampy -p COM5 put base_robot_bigwheel.p
y /base_robot_bigwheel.py
```

Turn your hub on and click on the “LEGO Hub” button to start the connection.

Select COM Port

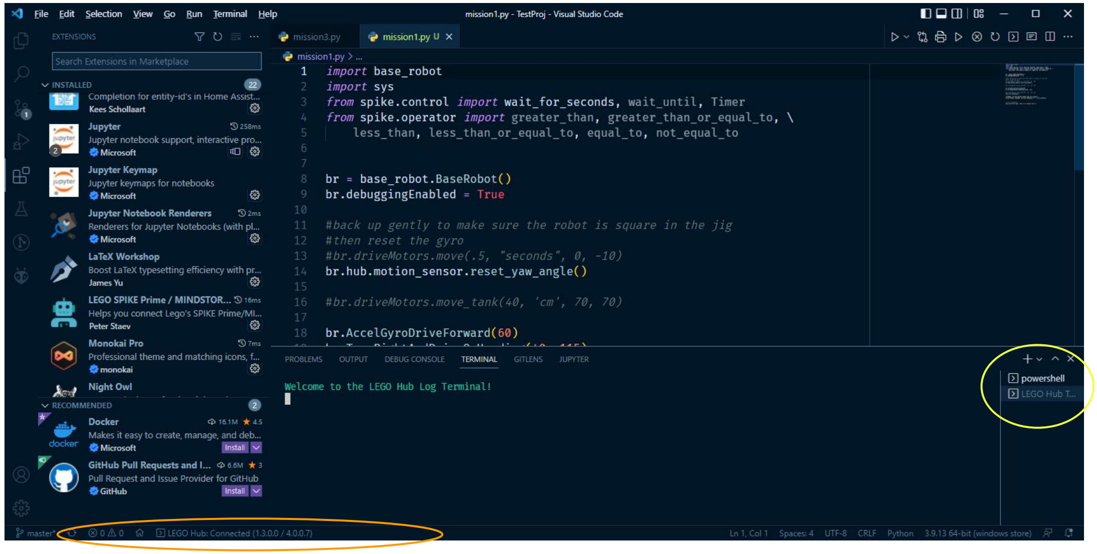

File Edit Selection View Go Run Terminal Help
mission1.py - TestProj - Visual Studio Code
EXPLORER
TESTPROJ
spike
base_robot_bigwheel.py
base_robot.py
instructions.txt
master.py
mission1.py U
mission3.py
spike.py
COM5
COM6
COM3
COM7
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL GITLENS JUPYTER
powershell
OUTLINE
TIMELINE
master* ⊗ 0 ⚠ 0 LEGO Hub: Disconnected
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python 3.9.13 64-bit (windows store)

```python
import sys
from spike.control import wait_for_seconds, wait_until, Timer
from spike.operator import greater_than, greater_than_or_equal_to, \
    less_than, less_than_or_equal_to, equal_to, not_equal_to


br = base_robot.BaseRobot()
br.debuggingEnabled = True

#back up gently to make sure the robot is square in the jig
#then reset the gyro
#br.driveMotors.move(.5, "seconds", 0, -10)
br.hub.motion_sensor.reset_yaw_angle()

#br.driveMotors.move_tank(40, 'cm', 70, 70)
```

```
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Skip\OneDrive\Documents\LEGO Education SPIKE\TestProj>
 *  History restored

Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Skip\OneDrive\Documents\LEGO Education SPIKE\TestProj> ampy -p COM5 put base_robot_bigwheel.p
y /base_robot_bigwheel.py
```

Select the correct COM port and you should be connected. If this doesn’t work, double-check that you have the right COM port and that the robot is turned on and a Bluetooth connection has been made at least once using the Spike app. Also make sure nothing else is connected to the bluetooth connection on the hub.

Connected!



```python
import base_robot
import sys
from spike.control import wait_for_seconds, wait_until, Timer
from spike.operator import greater_than, greater_than_or_equal_to, \
    less_than, less_than_or_equal_to, equal_to, not_equal_to


br = base_robot.BaseRobot()
br.debuggingEnabled = True

#back up gently to make sure the robot is square in the jig
#then reset the gyro
#br.driveMotors.move(.5, "seconds", 0, -10)
br.hub.motion_sensor.reset_yaw_angle()

#br.driveMotors.move_tank(40, 'cm', 70, 70)

br.AccelGyroDriveForward(60)
```

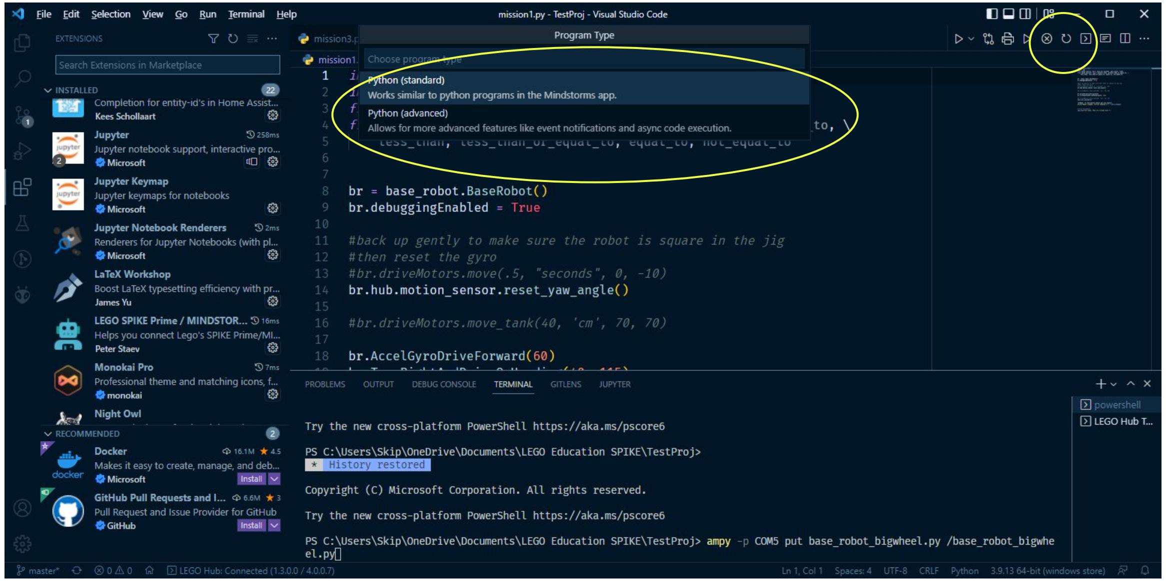
```
Welcome to the LEGO Hub Log Terminal!
```

```
master*    ⊗ 0 ⚠ 0    LEGO Hub: Connected (1.3.0.0 / 4.0.0.7)    Ln 1, Col 1   Spaces: 4   UTF-8   CRLF   Python   3.9.13 64-bit (windows store)
```

mission1.py - TestProj - Visual Studio Code
File Edit Selection View Go Run Terminal Help
mission3.py
mission1.py U
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL GITLENS JUPYTER
powershell
LEGO Hub T...

It now says "Connected" on the bottom and you can see the LEGO Hub Log Terminal. This terminal is where you will see the output from any print() statements. You can also switch to PowerShell which is an easy way to run ampy if you need to.

Upload a Program



mission1.py - TestProj - Visual Studio Code

Program Type

Choose program type

Python (standard)
Works similar to python programs in the Mindstorms app.

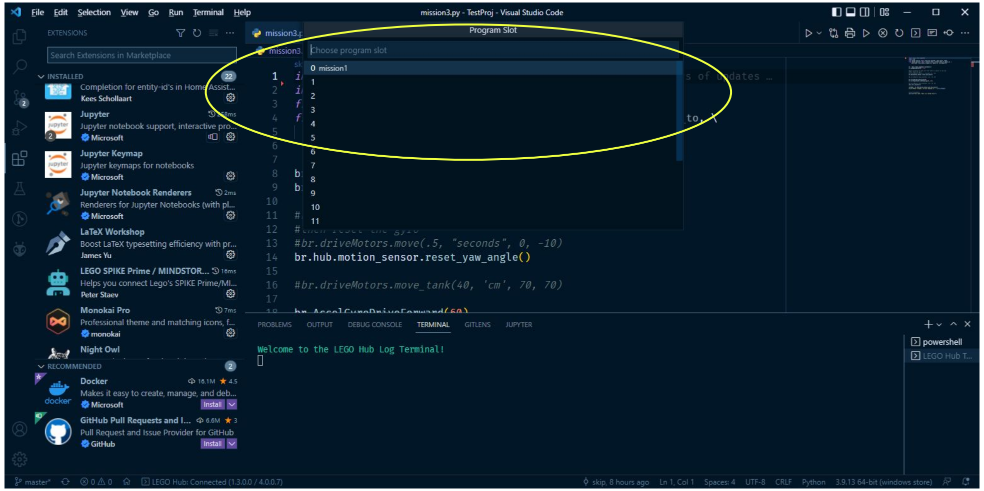
Python (advanced)
Allows for more advanced features like event notifications and async code execution.

```
br = base_robot.BaseRobot()
br.debuggingEnabled = True

#back up gently to make sure the robot is square in the jig
#then reset the gyro
#br.driveMotors.move(.5, "seconds", 0, -10)
br.hub.motion_sensor.reset_yaw_angle()

#br.driveMotors.move_tank(40, 'cm', 70, 70)

br.AccelGyroDriveForward(60)
```

```
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Skip\OneDrive\Documents\LEGO Education SPIKE\TestProj>
 *  History restored

Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Skip\OneDrive\Documents\LEGO Education SPIKE\TestProj> ampy -p COM5 put base_robot_bigwheel.py /base_robot_bigwhe
el.py
```

Click the circle-arrow icon to upload a program. I have only tried running standard python programs, so if you try the advanced option, you are on your own.

Choose a Slot

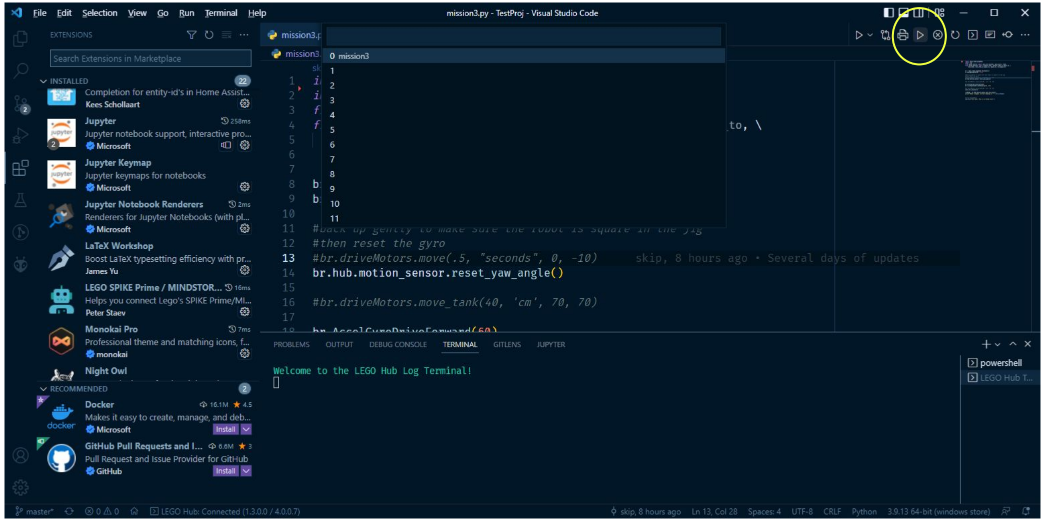


```
#br.driveMotors.move(.5, "seconds", 0, -10)
br.hub.motion_sensor.reset_yaw_angle()

#br.driveMotors.move_tank(40, 'cm', 70, 70)
```

```
Welcome to the LEGO Hub Log Terminal!
```

Choose a slot to store the program. There are twenty possible slots to use.

Run a Program


File Edit Selection View Go Run Terminal Help
mission3.py - TestProj - Visual Studio Code
EXTENSIONS
Search Extensions in Marketplace
INSTALLED 22
RECOMMENDED 2
mission3.py
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL GITLENS JUPYTER
powershell
LEGO Hub T...
Install

Completion for entity-id's in Home Assist... Kees Schollaart

Jupyter — Jupyter notebook support, interactive pro... Microsoft

Jupyter Keymap — Jupyter keymaps for notebooks Microsoft

Jupyter Notebook Renderers — Renderers for Jupyter Notebooks (with pl... Microsoft

LaTeX Workshop — Boost LaTeX typesetting efficiency with pr... James Yu

LEGO SPIKE Prime / MINDSTOR... — Helps you connect Lego's SPIKE Prime/MI... Peter Staev

Monokai Pro — Professional theme and matching icons, f... monokai

Night Owl

Docker — Makes it easy to create, manage, and deb... Microsoft

GitHub Pull Requests and I... — Pull Request and Issue Provider for GitHub GitHub

```
0 mission3
1
2
3
4
5
6
7
8
9
10
11
```

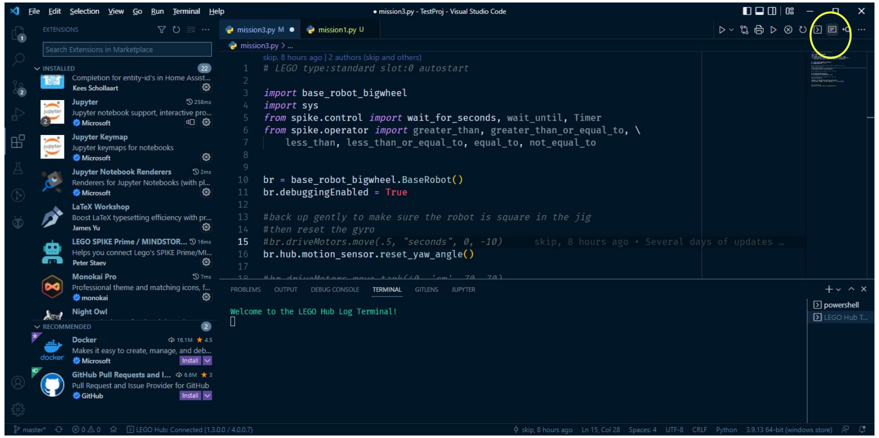
```
#then reset the gyro
#br.driveMotors.move(.5, "seconds", 0, -10)      skip, 8 hours ago • Several days of updates
br.hub.motion_sensor.reset_yaw_angle()

#br.driveMotors.move_tank(40, 'cm', 70, 70)
```

```
Welcome to the LEGO Hub Log Terminal!
```

```
master*   ⊗ 0 ⚠ 0   LEGO Hub: Connected (1.3.0.0 / 4.0.0.7)   skip, 8 hours ago   Ln 13, Col 28   Spaces: 4   UTF-8   CRLF   Python   3.9.13 64-bit (windows store)
```

To run a program, click the “play button and choose the slot. Your program will run on the hub. Or crash if you are like me and you made some silly syntax error.

Upload and Run with ONE Click!



```
# LEGO type:standard slot:0 autostart

import base_robot_bigwheel
import sys
from spike.control import wait_for_seconds, wait_until, Timer
from spike.operator import greater_than, greater_than_or_equal_to, \
    less_than, less_than_or_equal_to, equal_to, not_equal_to


br = base_robot_bigwheel.BaseRobot()
br.debuggingEnabled = True

#back up gently to make sure the robot is square in the jig
#then reset the gyro
#br.driveMotors.move(.5, "seconds", 0, -10)
br.hub.motion_sensor.reset_yaw_angle()
```

```
Welcome to the LEGO Hub Log Terminal!
```

Click the “Add File Header” button and it will ask you some questions and then add a small comment as the first line of your program. In this case, it added:

LEGO type: standard slot:0 autostart

And now when I click the upload button, the program will be uploaded to slot 0 as a standard program, and then it will run.

Git - The Ultimate in Code Sharing and Version Control

This is optional, but if you install git (be sure to pay attention for the installation step that asks if you want to add git to the path. Answer “yes”), then you can upload your code to github. This tutorial will not go into using git and github, but there are many good tutorials on the internet that should be able to help you.