Lab 1: Debugging a Raspberry Pi Internet of Things Flask Application

Author	**Dave Glover, Microsoft Cloud Developer Advocate**
Platforms	Linux, macOS, Windows, Raspbian Buster
Tools	Visual Studio Code
Hardware	Raspberry Pi 4. 4GB model required for 20 Users. Raspberry Pi Sense HAT, Optional: Raspberry Pi case, active cooling fan
USB3 SSD Drive	To support up to 20 users per Raspberry Pi you need a **fast** USB3 SSD Drive to run Raspbian Buster Linux on. A 120 USB3 SSD drive will be more than sufficient. These are readily available from online stores.
Language	Python
Date	As of September, 2019

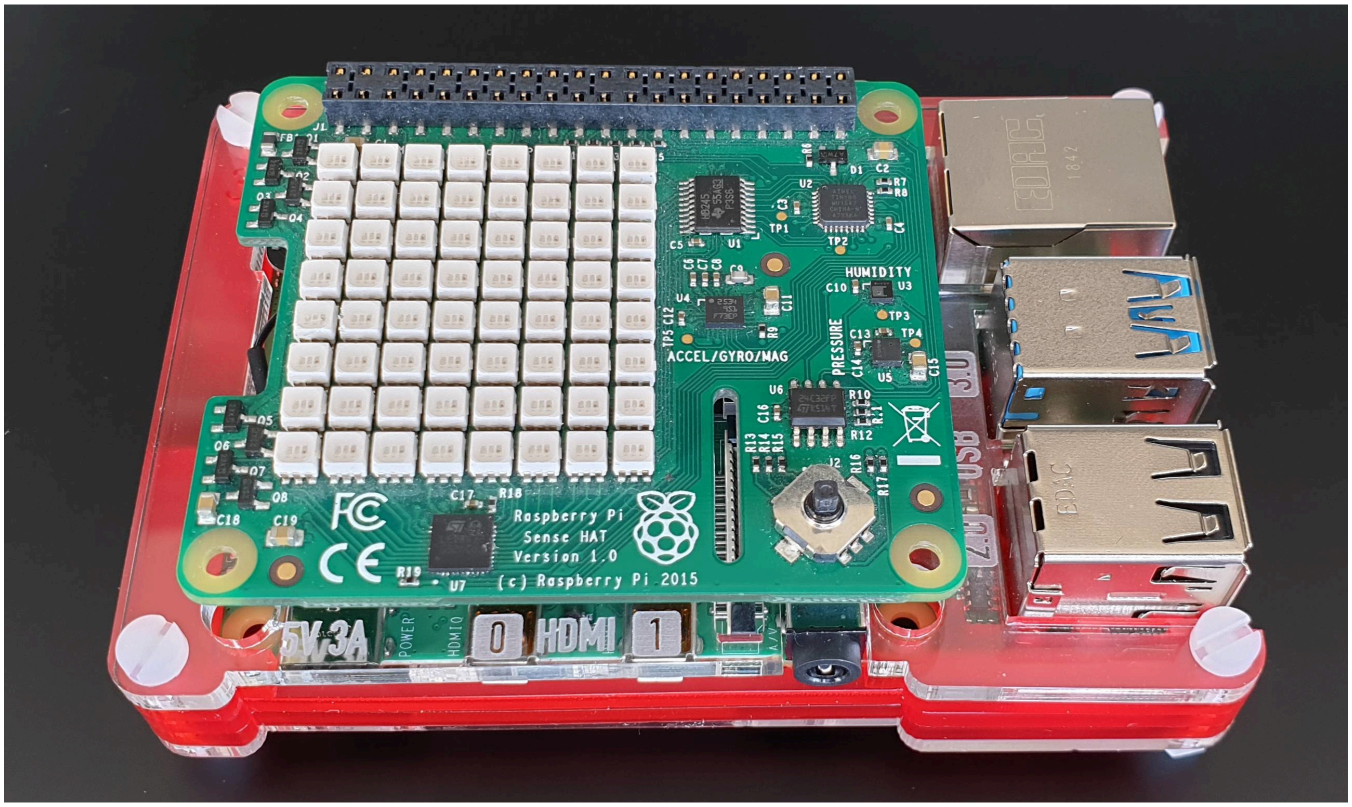
Follow me on Twitter @dglover

PDF Lab Guide

You may find it easier to download and follow the PDF version of the Debugging Raspberry Pi Internet of Things Flask App hands-on lab guide.

Introduction

In this hands-on lab, you will learn how to create and debug a Python web application on a Raspberry Pi with Visual Studio Code and the Remote SSH extension. The web app will read the temperature, humidity, and air pressure telemetry from a sensor connected to the Raspberry Pi.



Software Installation

This hands-on lab uses Visual Studio Code. Visual Studio Code is a code editor and is one of the most popular **Open Source** projects on GitHub. It runs on Linux, macOS, and Windows.

Install Visual Studio Code

1. Install Visual Studio Code

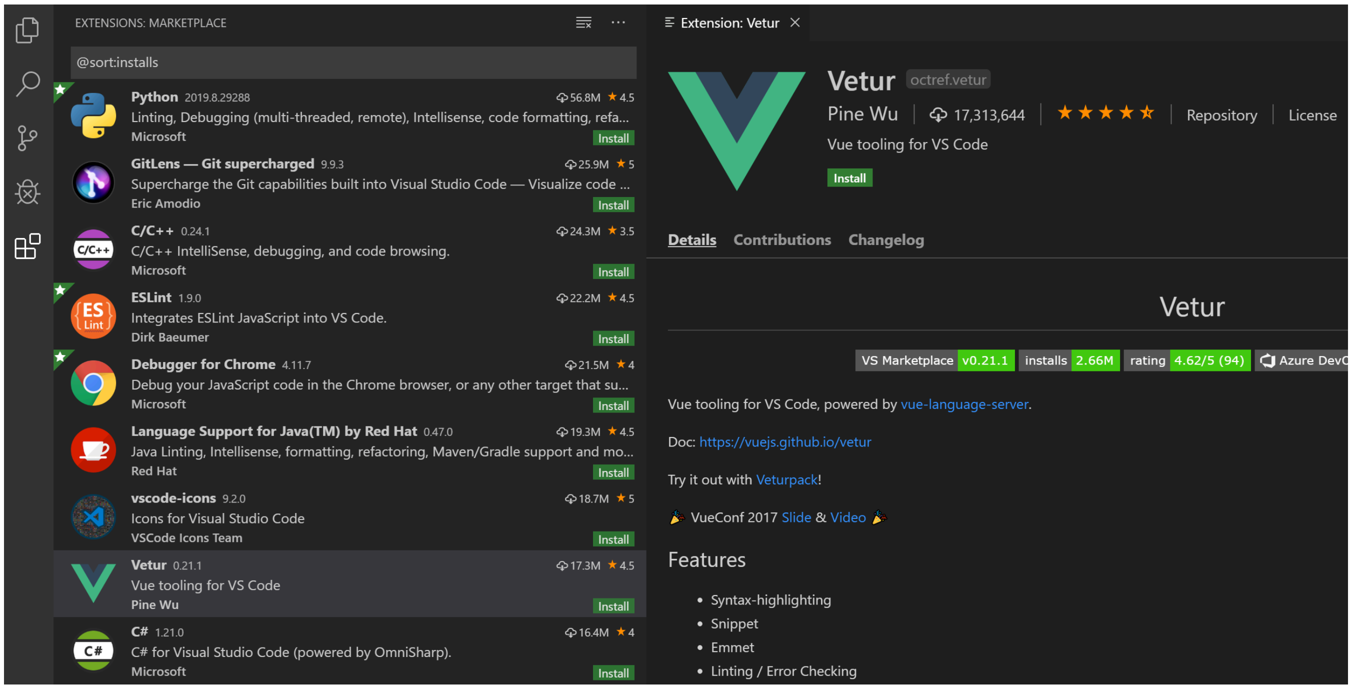
Visual Studio Code Extensions

The features that Visual Studio Code includes out-of-the-box are just the start. VS Code extensions let you add languages, debuggers, and tools to your installation to support your development workflow. VS Code's rich extensibility model lets extension authors plug directly into the VS Code UI and contribute functionality through the same APIs used by VS Code.

Browse for extensions

You can browse and install extensions from within VS Code. Bring up the Extensions view by clicking on the Extensions icon in the **Activity Bar** on the side of VS Code or from the Visual Studio Code menu, select **View** >**Extensions**.

This will show you a list of the most popular VS Code extensions on the VS Code Marketplace.


EXTENSIONS: MARKETPLACE

@sort:installs

Python 2019.8.29288 56.8M ★4.5
Linting, Debugging (multi-threaded, remote), Intellisense, code formatting, refa...
Microsoft Install

GitLens — Git supercharged 9.9.3 25.9M ★5
Supercharge the Git capabilities built into Visual Studio Code — Visualize code ...
Eric Amodio Install

C/C++ 0.24.1 24.3M ★3.5
C/C++ IntelliSense, debugging, and code browsing.
Microsoft Install

ESLint 1.9.0 22.2M ★4.5
Integrates ESLint JavaScript into VS Code.
Dirk Baeumer Install

Debugger for Chrome 4.11.7 21.5M ★4
Debug your JavaScript code in the Chrome browser, or any other target that su...
Microsoft Install

Language Support for Java(TM) by Red Hat 0.47.0 19.3M ★4.5
Java Linting, Intellisense, formatting, refactoring, Maven/Gradle support and mo...
Red Hat Install

vscode-icons 9.2.0 18.7M ★5
Icons for Visual Studio Code
VSCode Icons Team Install

Vetur 0.21.1 17.3M ★4.5
Vue tooling for VS Code
Pine Wu Install

C# 1.21.0 16.4M ★4
C# for Visual Studio Code (powered by OmniSharp).
Microsoft Install

Extension: Vetur

Vetur octref.vetur
Pine Wu | 17,313,644 | ★★★★★ | Repository | License
Vue tooling for VS Code
Install

Details Contributions Changelog

Vetur

VS Marketplace v0.21.1 | installs 2.66M | rating 4.62/5 (94) | Azure DevO

Vue tooling for VS Code, powered by vue-language-server.

Doc: https://vuejs.github.io/vetur

Try it out with Veturpack!

VueConf 2017 Slide & Video

Features

- Syntax-highlighting
- Snippet
- Emmet
- Linting / Error Checking

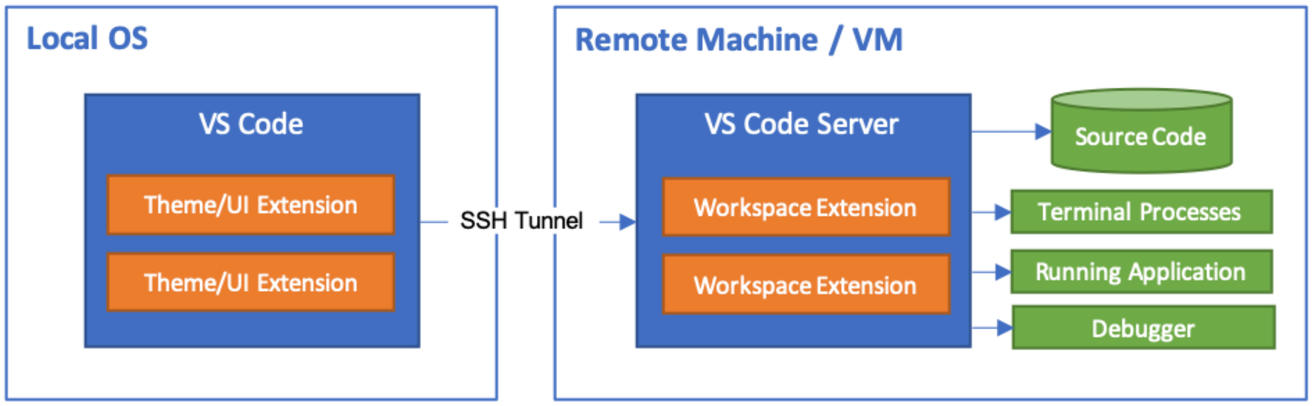
Install the Python and Remote SSH Extensions

Search and install the following two extensions published by Microsoft.

1. Remote - SSH
2. Python

Remote SSH Development

The Visual Studio Code Remote - SSH extension allows you to open a remote folder on any remote machine, virtual machine, or container with a running SSH server and take full advantage of Visual Studio Code.



Raspberry Pi Hardware

If you are attending a workshop, then you can use a shared network-connected Raspberry Pi. You can also use your own network-connected Raspberry Pi for this hands-on lab.

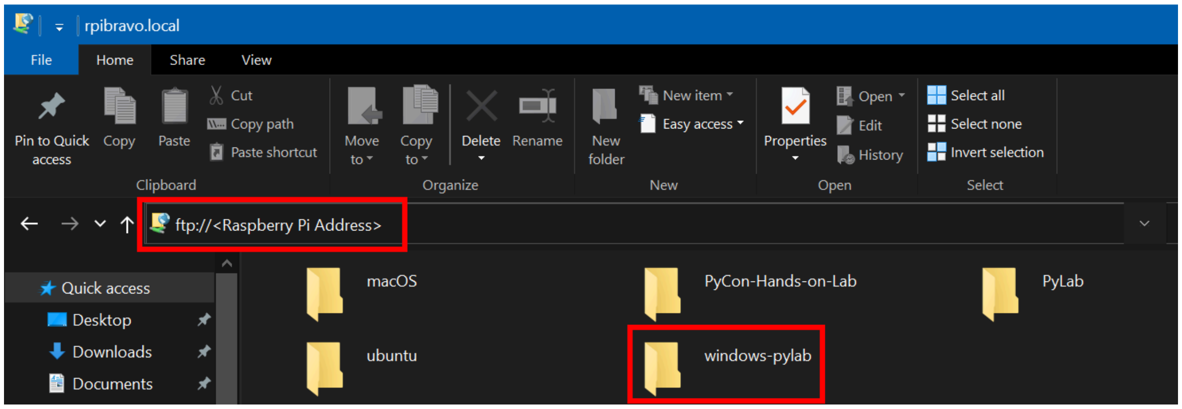
You will need the following information from the lab instructor.

1. The **Network IP Address** of the Raspberry Pi
2. Your assigned **login name** and **password**.

SSH Authentication with private/public keys

Setting up a public/private key pair for SSH authentication is a secure and fast way to authenticate from your computer to the Raspberry Pi. This is recommended for this hands-on lab.

SSH for Windows Users

1. From Windows File Explorer, open **ftp://<Raspberry Pi Address>**
   

2. Copy the **windows-pylab** directory to your **desktop**
3. Open the **windows-pylab** folder you copied to your **desktop**
4. Double click **windows-ssh.cmd**
   You will be guided through the process of setting up an SSH key and copying the SSH public key to the Raspberry Pi.

SSH for Linux and macOS Users

From a Linux or macOS **Terminal Console** or from **git bash** in windows run the following commands:

1. Create your key. This is typically a one-time operation. **Take the default options**.
   ```
   ssh-keygen -t rsa -b 4096 -f ~/.ssh/id_rsa_python_lab
   ```
2. Copy the public key to the Raspberry Pi.
   ```
   ssh-copy-id -i ~/.ssh/id_rsa_python_lab <login@Raspberry IP Address>
   ```
   For example:
   ```
   ssh-copy-id -i ~/.ssh/id_rsa_python_lab dev99@192.168.1.200
   ```
3. Test the SSH Authentication Key

```
ssh -i ~/.ssh/id_rsa_python_lab <login@Raspberry IP Address>
```

A new SSH session will start. You should now be connected to the Raspberry Pi **without** being prompted for the password.

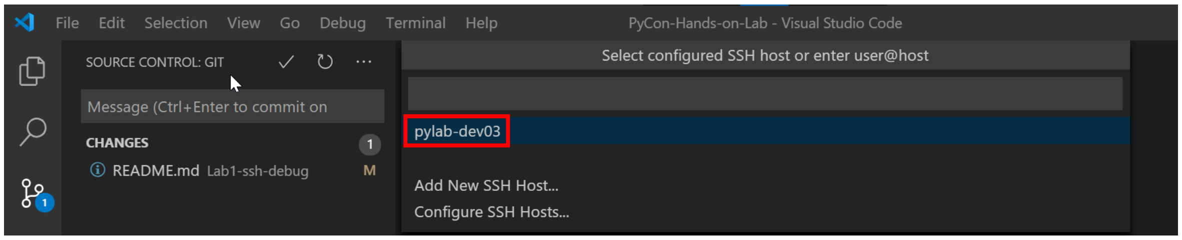
4. Close the SSH session. In the SSH terminal, type exit, followed by ENTER.

Trouble Shooting SSH Client Installation

- Remote Development using SSH
- Installing a supported SSH client

Start a Remote SSH Connection

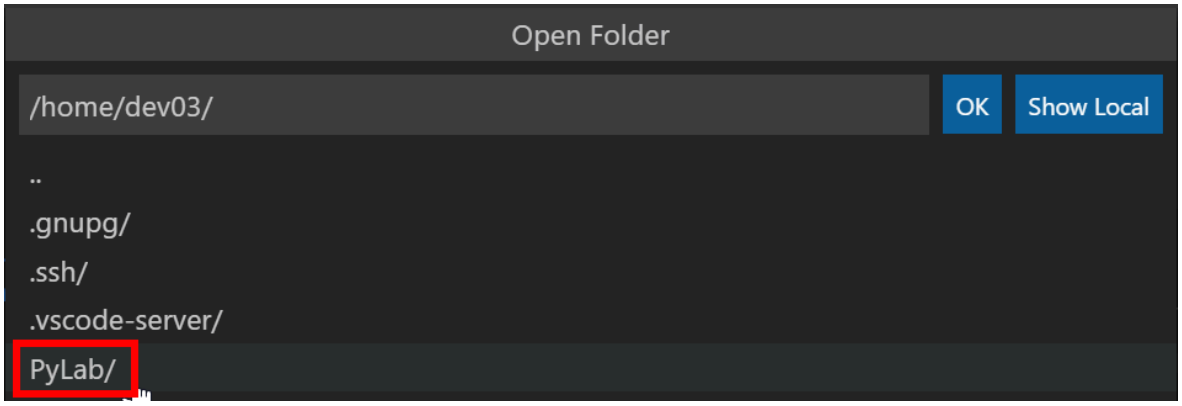
1. Start Visual Studio Code
2. Press **F1** to open the Command Palette, type **ssh connect** and select **Remote-SSH: Connect to Host**
3. Select the **pylab-devnn** configuration


File Edit Selection View Go Debug Terminal Help
PyCon-Hands-on-Lab - Visual Studio Code
SOURCE CONTROL: GIT
Message (Ctrl+Enter to commit on
CHANGES
README.md Lab1-ssh-debug
M
Select configured SSH host or enter user@host
pylab-dev03
Add New SSH Host...
Configure SSH Hosts...

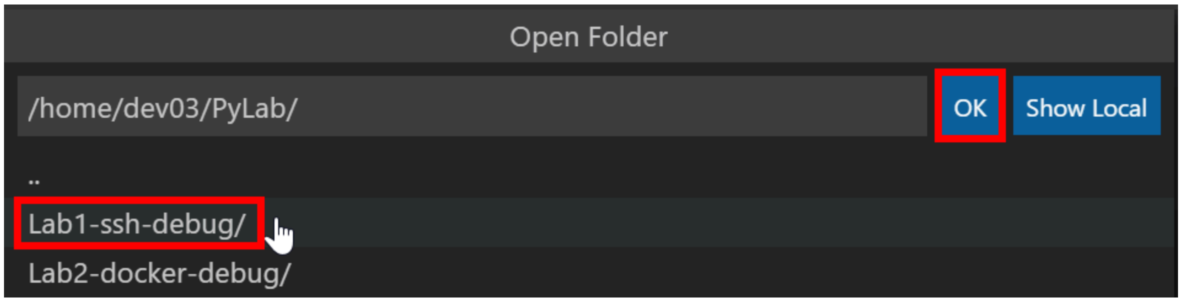
It will take a moment to connect to the Raspberry Pi.

Open the Lab 1 SSH Debug Project

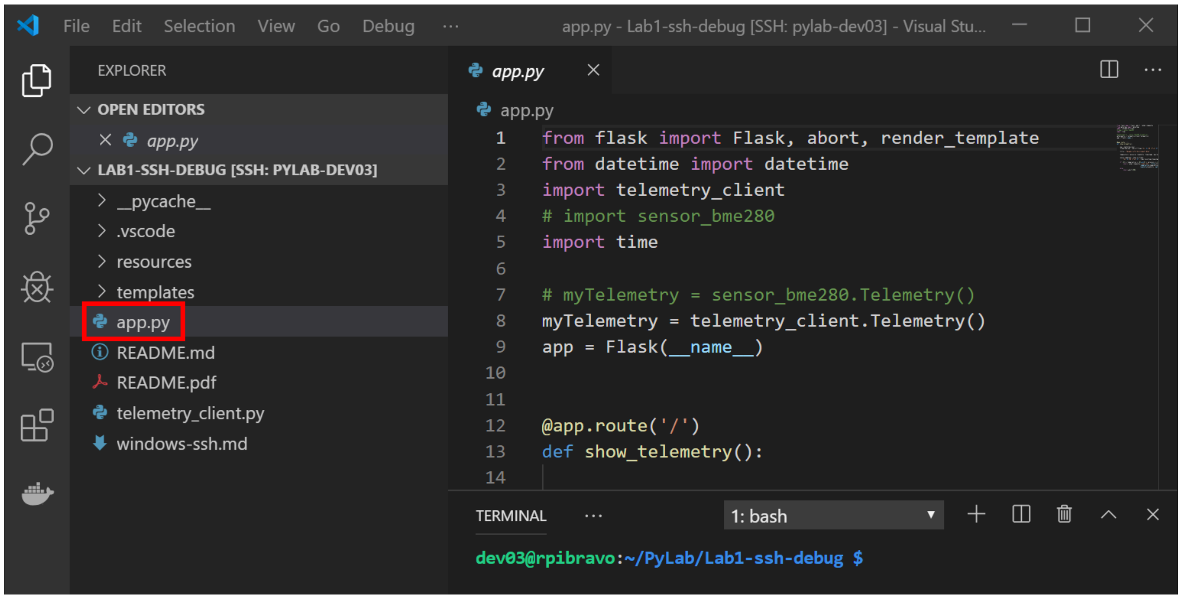
1. From Visual Studio Code main menu: **File** >**Open Folder**
2. Select the **PyLab** directory



3. Next select the **Lab1-ssh-debug** directory



4. Click **OK** to Open the directory
5. From the **Explorer** bar, open the **app.py** file and review the contents

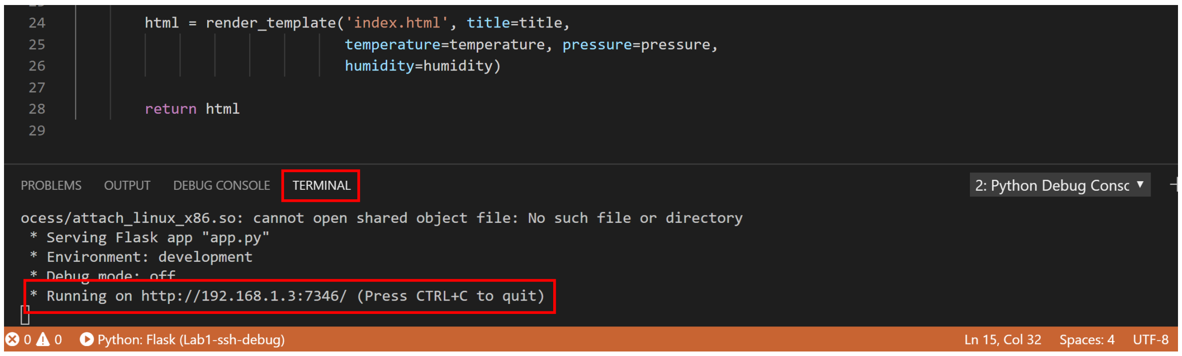


```python
from flask import Flask, abort, render_template
from datetime import datetime
import telemetry_client
# import sensor_bme280
import time

# myTelemetry = sensor_bme280.Telemetry()
myTelemetry = telemetry_client.Telemetry()
app = Flask(__name__)


@app.route('/')
def show_telemetry():
```

```
dev03@rpibravo:~/PyLab/Lab1-ssh-debug $
```

Run the Python Flask App

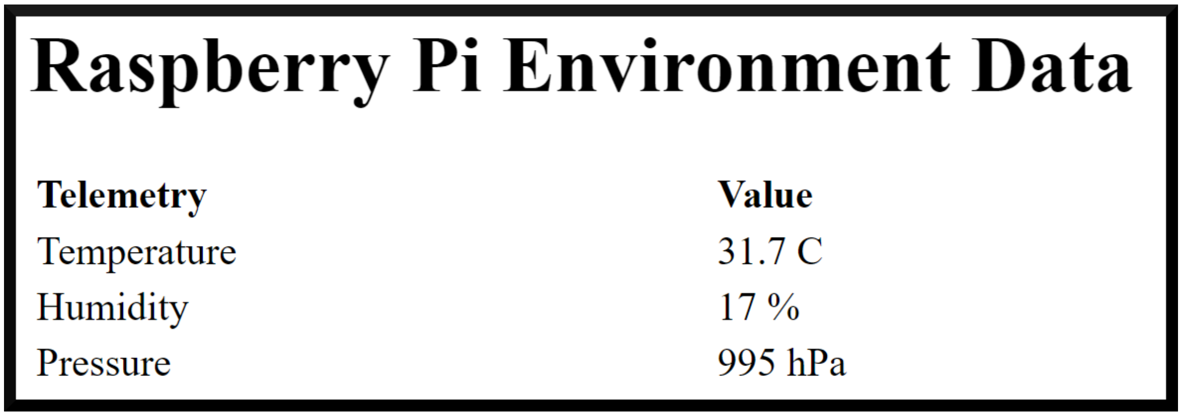
1. Press **F5** to start the Python Flask app.
2. From the Visual Studio Code **Terminal Window**, click the **running on http://...** web link.



```
        html = render_template('index.html', title=title,
                               temperature=temperature, pressure=pressure,
                               humidity=humidity)

        return html
```

```
ocess/attach_linux_x86.so: cannot open shared object file: No such file or directory
 * Serving Flask app "app.py"
 * Environment: development
 * Debug mode: off
 * Running on http://192.168.1.3:7346/ (Press CTRL+C to quit)
```

3. This will launch your desktop Web Browser.
    - The Flask app will read the temperature, air pressure, humidity from the **sensor** attached the Raspberry Pi and display the results in your web browser.



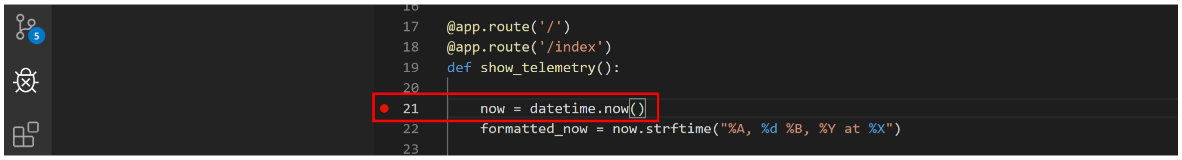
Raspberry Pi Environment Data

Telemetry	Value
Temperature	31.7 C
Humidity	17 %
Pressure	995 hPa

Set a Visual Studio Code Breakpoint

1. Switch back to Visual Studio Code and ensure the **app.py** file is open.
2. Put the cursor on the line that reads **now = datetime.now()**
3. Use one of the following methods to set a **breakpoint**.
    - Press **F9**
    - From the main menu, select **Debug** >**Toggle Breakpoint**
    - **Click directly** in the **margin to the left** of the line number (a faded red dot appears when hovering there)

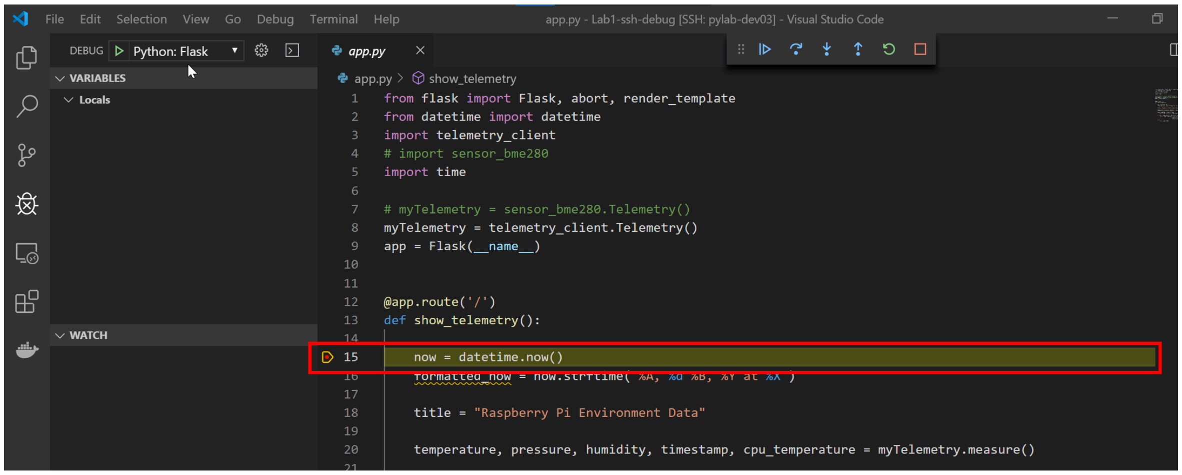
The breakpoint appears as a red dot in the left margin



```
@app.route('/')
@app.route('/index')
def show_telemetry():

    now = datetime.now()
    formatted_now = now.strftime("%A, %d %B, %Y at %X")
```

4. Switch back to the **Web Browser** and click **Refresh**
5. Switch back to **Visual Studio Code**. You will see that the code has stopped

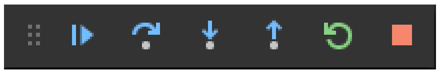
running at the **breakpoint**.



```python
from flask import Flask, abort, render_template
from datetime import datetime
import telemetry_client
# import sensor_bme280
import time

# myTelemetry = sensor_bme280.Telemetry()
myTelemetry = telemetry_client.Telemetry()
app = Flask(__name__)


@app.route('/')
def show_telemetry():

    now = datetime.now()
    formatted_now = now.strftime("%A, %d %B, %Y at %X")

    title = "Raspberry Pi Environment Data"

    temperature, pressure, humidity, timestamp, cpu_temperature = myTelemetry.measure()
```

Stepping through code with the Debugger

When a debug session starts, the **Debug toolbar** will appear at the top of the editor window.

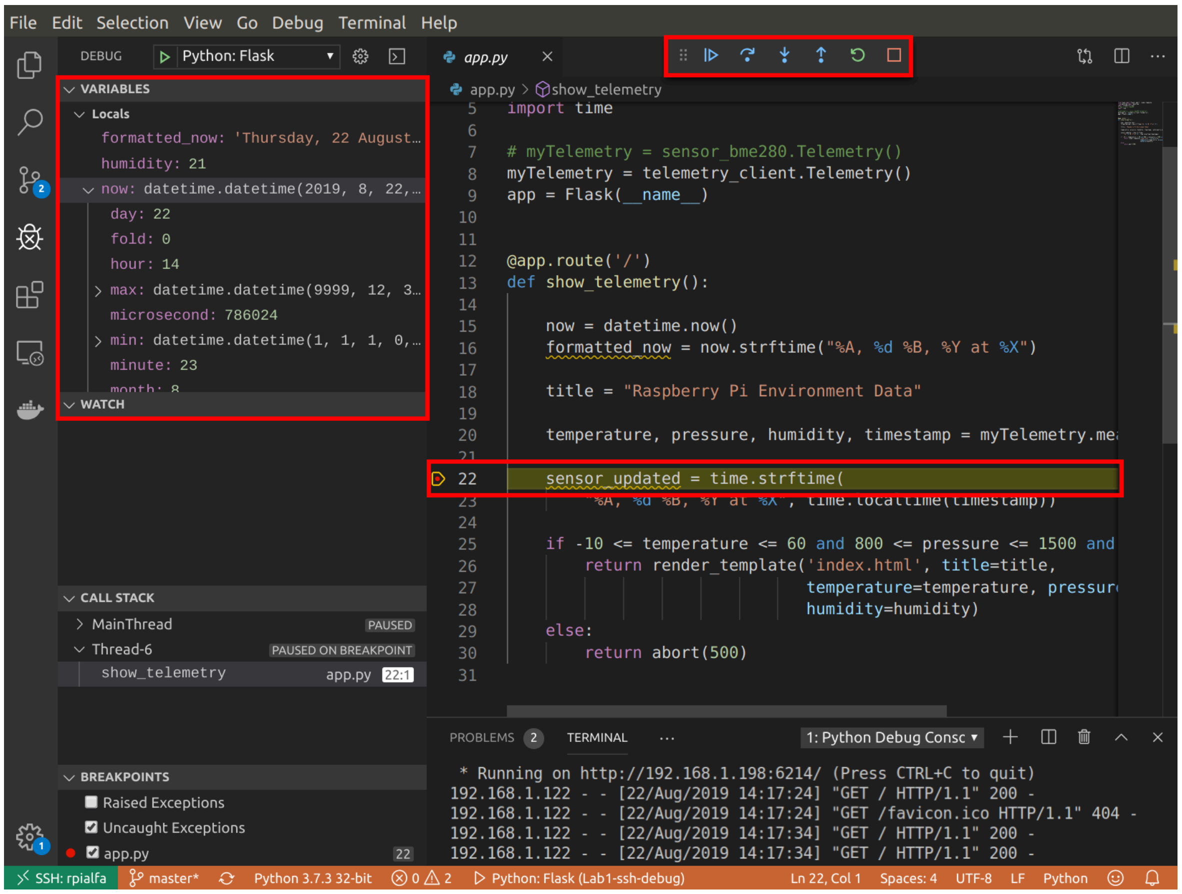


The debugging toolbar (shown above) will appear in Visual Studio Code. It has the following options:

1. Pause (or Continue, F5),
2. Step Over (F10)
3. Step Into (F11),
4. Step Out (Shift+F11),
5. Restart (Ctrl+Shift+F5),
6. and Stop (Shift+F5).

Stepping through Code with the Debugger

1. Step through the code by pressing (**F10**) or clicking **Step Over** on the debugging toolbar.
2. **Repeat** pressing **F10** until you reach the line that reads **if -40 <= temperature <= 60 and 0 <= pressure <= 1500 and 0 <= humidity <= 100:**
3. You will notice that Python variables are displayed in the **Variables Window**.



```
formatted_now: 'Thursday, 22 August…
humidity: 21
now: datetime.datetime(2019, 8, 22,…
  day: 22
  fold: 0
  hour: 14
  max: datetime.datetime(9999, 12, 3…
  microsecond: 786024
  min: datetime.datetime(1, 1, 1, 0,…
  minute: 23
  month: 8
```

```
MainThread                 PAUSED
Thread-6                   PAUSED ON BREAKPOINT
  show_telemetry           app.py 22:1
```

```
Raised Exceptions
Uncaught Exceptions
app.py                     22
```

```
5   import time
6
7   # myTelemetry = sensor_bme280.Telemetry()
8   myTelemetry = telemetry_client.Telemetry()
9   app = Flask(__name__)
10
11
12  @app.route('/')
13  def show_telemetry():
14
15      now = datetime.now()
16      formatted_now = now.strftime("%A, %d %B, %Y at %X")
17
18      title = "Raspberry Pi Environment Data"
19
20      temperature, pressure, humidity, timestamp = myTelemetry.mea
21
22      sensor_updated = time.strftime(
23          "%A, %d %B, %Y at %X", time.localtime(timestamp))
24
25      if -10 <= temperature <= 60 and 800 <= pressure <= 1500 and
26          return render_template('index.html', title=title,
27                                 temperature=temperature, pressur
28                                 humidity=humidity)
29      else:
30          return abort(500)
31
```

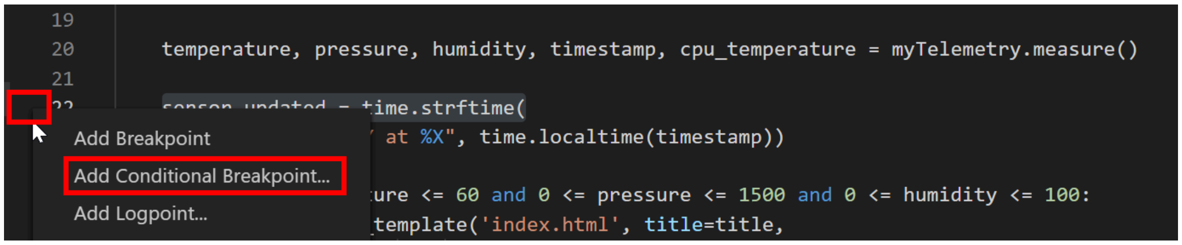
```
 * Running on http://192.168.1.198:6214/ (Press CTRL+C to quit)
192.168.1.122 - - [22/Aug/2019 14:17:24] "GET / HTTP/1.1" 200 -
192.168.1.122 - - [22/Aug/2019 14:17:24] "GET /favicon.ico HTTP/1.1" 404 -
192.168.1.122 - - [22/Aug/2019 14:17:34] "GET / HTTP/1.1" 200 -
192.168.1.122 - - [22/Aug/2019 14:17:34] "GET / HTTP/1.1" 200 -
```

4. **Right mouse click** a variable, and you will discover you can change, copy, or watch variables.
5. Try to change **temperature** variable to **50**. Hint, **right mouse** click on the temperature variable and select **Set Value**, or double click on a **temperature** variable.
6. Press **F5** to resume the Flask App, then **switch back to your web browser** and you will see the temperature, humidity, and pressure Sensor data displayed on the web page.

Set a Conditional Breakpoint

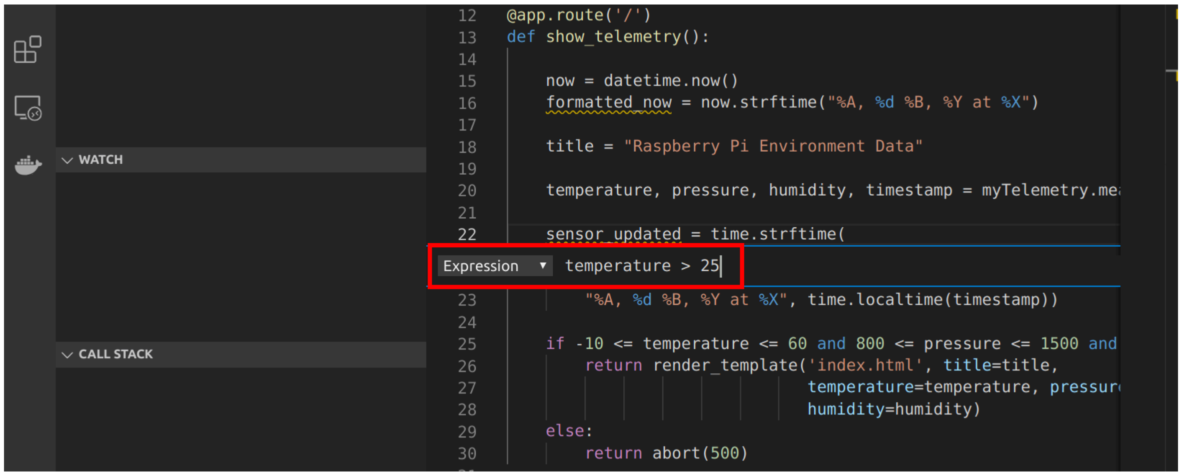
Try setting a **conditional** breakpoint

1. Clear the existing breakpoints. From the main menu select **Debug** > **Remove all breakpoints**.
2. Ensure the **app.py** file open.
3. **Right mouse click** directly in the margin to the **left** of the line number **22**.



```
    temperature, pressure, humidity, timestamp, cpu_temperature = myTelemetry.measure()

    sensor_updated = time.strftime(
                  ' at %X", time.localtime(timestamp))

                 ure <= 60 and 0 <= pressure <= 1500 and 0 <= humidity <= 100:
                 template('index.html', title=title,
```

4. Select **Add Conditional Breakpoint...**
5. Set the condition to **temperature > 25**



```
@app.route('/')
def show_telemetry():

    now = datetime.now()
    formatted_now = now.strftime("%A, %d %B, %Y at %X")

    title = "Raspberry Pi Environment Data"

    temperature, pressure, humidity, timestamp = myTelemetry.mea

    sensor_updated = time.strftime(
```

Expression: temperature > 25

```
        "%A, %d %B, %Y at %X", time.localtime(timestamp))

    if -10 <= temperature <= 60 and 800 <= pressure <= 1500 and
        return render_template('index.html', title=title,
                               temperature=temperature, pressur
                               humidity=humidity)
    else:
        return abort(500)
```

The breakpoint appears as a red dot with an equals sign in the middle:

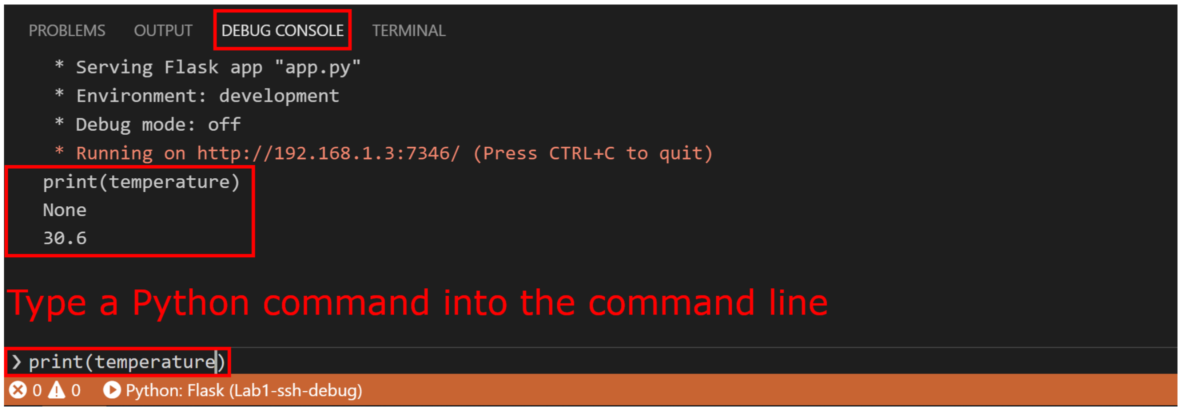
6. **Switch** back to your **web browsers** and **click refresh**
7. **Switch** back to **Visual Studio Code** and you will see the debugger has stopped at the **conditional breakpoint**.
8. Press **F5** to continue running the code
9. **Switch** back to your **web browser** to view the page.

Try the Debug Console

Try the Visual Studio Code **Debug Console**. This will give you access to the Python REPL (Read, Evaluate, Print Loop), try printing or setting variables, importing libraries, etc.

1. **Switch** back to your **web browser** and click refresh
2. **Switch** back to **Visual Studio Code**
3. The code should have stopped at the conditional breakpoint.
4. Select the Visual Studio **Debug Console** window.
5. Try the following Python code at the input prompt **>**

```
print(temperature)
```


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL

```
 * Serving Flask app "app.py"
 * Environment: development
 * Debug mode: off
 * Running on http://192.168.1.3:7346/ (Press CTRL+C to quit)
print(temperature)
None
30.6
```

Type a Python command into the command line

```
> print(temperature)
```

0 0 Python: Flask (Lab1-ssh-debug)

6. Press **Enter**
7. Try running the following Python code snippets from the input prompt.

```
temperature = 24
import random
random.randrange(100, 1000)
```

8. Press **F5** to continue the execution of the Python code.
9. Switch back to you web browser to see the updated page.

Lab Challenges

Lab Challenge 1: Update the Flask Template

1. Update the Flask **index.html** template found in the **templates** folder to display the current date and time.
2. Rerun the Flask app.

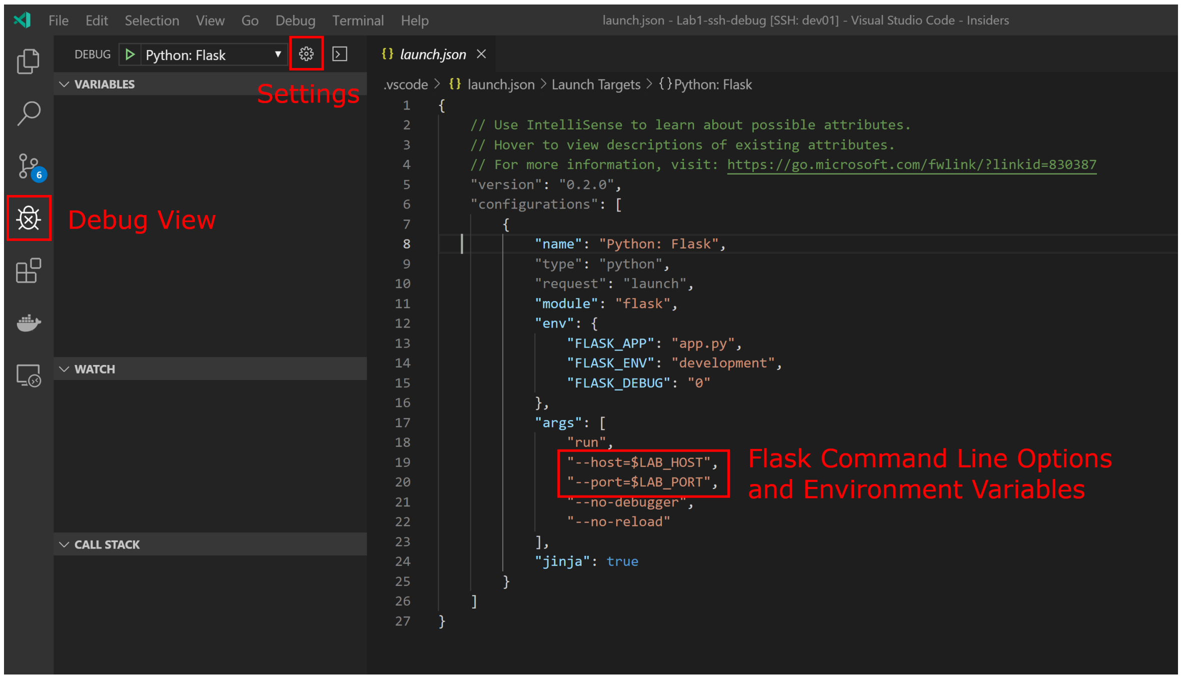
Lab Challenge 2: Experiment with Debugger Options

Things to try:

1. Review the Visual Studio Code Python Tutorial
2. Review the Python Flask tutorial
3. Review the Visual Studio Code Debugging Tutorial

Review the debug options

1. Switch to Debug view in Visual Studio Code (using the left-side activity bar).


File Edit Selection View Go Debug Terminal Help
launch.json - Lab1-ssh-debug [SSH: dev01] - Visual Studio Code - Insiders
DEBUG
Python: Flask
Settings
launch.json
.vscode > launch.json > Launch Targets > Python: Flask
VARIABLES
Debug View
WATCH
CALL STACK
Flask Command Line Options
and Environment Variables

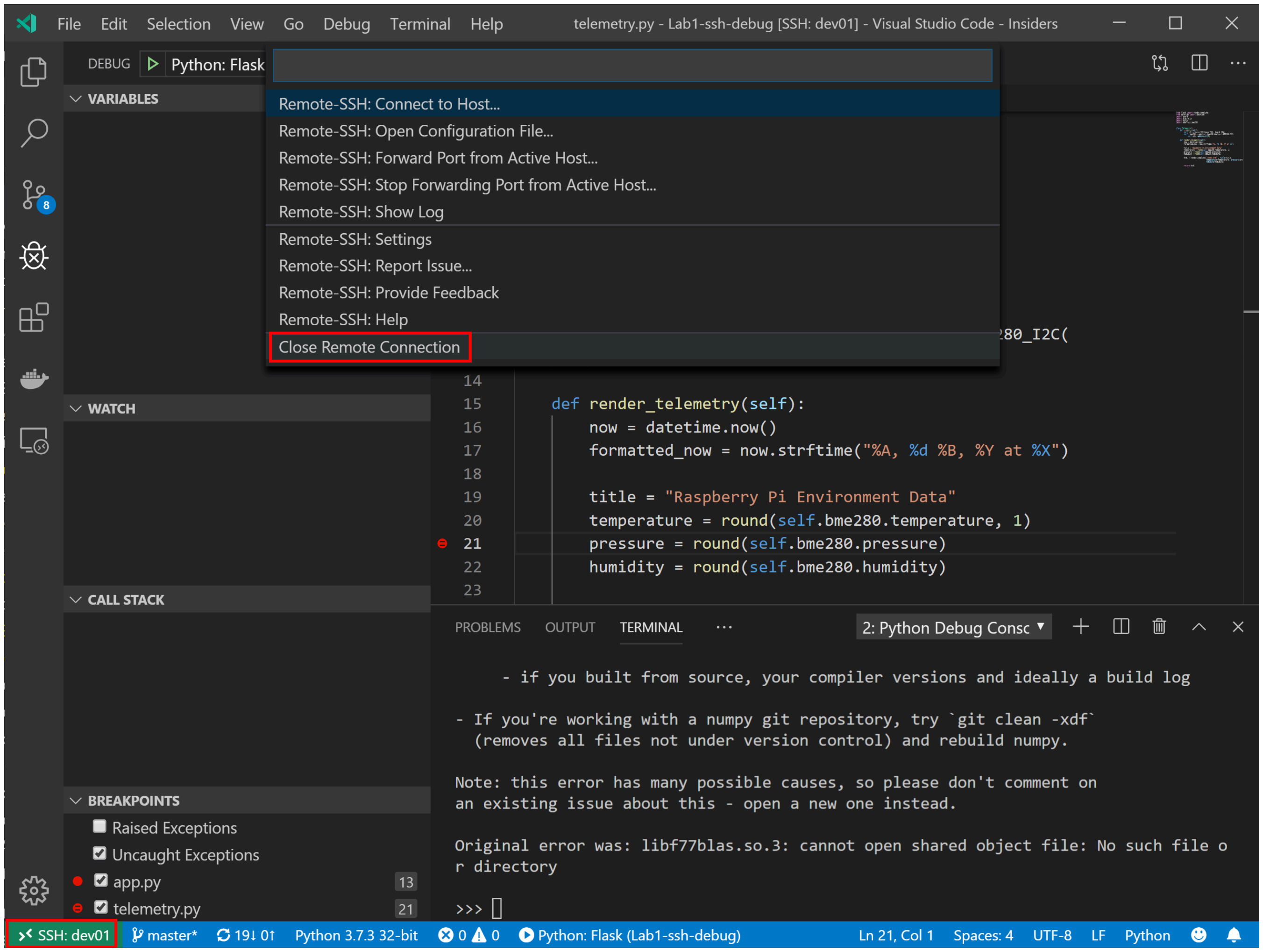
```
{
    // Use IntelliSense to learn about possible attributes.
    // Hover to view descriptions of existing attributes.
    // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
    "version": "0.2.0",
    "configurations": [
        {
            "name": "Python: Flask",
            "type": "python",
            "request": "launch",
            "module": "flask",
            "env": {
                "FLASK_APP": "app.py",
                "FLASK_ENV": "development",
                "FLASK_DEBUG": "0"
            },
            "args": [
                "run",
                "--host=$LAB_HOST",
                "--port=$LAB_PORT",
                "--no-debugger",
                "--no-reload"
            ],
            "jinja": true
        }
    ]
}
```

2. Click the **Settings** button which will open the **launch.json** file.
3. The **launch.json** file defines how the Flask app will start, and what Flask Command Line parameters to pass at startup.
   There are two environment variables used in the launch.json file. These are **LAB_HOST** (which is the IP Address of the Raspberry Pi), and **LAB_PORT** (a random TCP/IP Port number between 5000 and 8000). These environment variables are set by the .bashrc script which runs when you connect to the Raspberry Pi with Visual Studio Remote SSH.

Closing the Remote SSH Session

From Visual Studio Code, **Close Remote Connection**.

1. Click the **Remote SSH** button in the **bottom left-hand corner** and select **Close Remote Connection** from the dropdown list.


File Edit Selection View Go Debug Terminal Help
telemetry.py - Lab1-ssh-debug [SSH: dev01] - Visual Studio Code - Insiders
DEBUG
Python: Flask
Remote-SSH: Connect to Host...
Remote-SSH: Open Configuration File...
Remote-SSH: Forward Port from Active Host...
Remote-SSH: Stop Forwarding Port from Active Host...
Remote-SSH: Show Log
Remote-SSH: Settings
Remote-SSH: Report Issue...
Remote-SSH: Provide Feedback
Remote-SSH: Help
Close Remote Connection
VARIABLES
WATCH
CALL STACK
BREAKPOINTS
Raised Exceptions
Uncaught Exceptions
app.py 13
telemetry.py 21
PROBLEMS OUTPUT TERMINAL
2: Python Debug Consc
SSH: dev01
master*
19↓ 0↑
Python 3.7.3 32-bit
0 0
Python: Flask (Lab1-ssh-debug)
Ln 21, Col 1 Spaces: 4 UTF-8 LF Python

```python
    def render_telemetry(self):
        now = datetime.now()
        formatted_now = now.strftime("%A, %d %B, %Y at %X")

        title = "Raspberry Pi Environment Data"
        temperature = round(self.bme280.temperature, 1)
        pressure = round(self.bme280.pressure)
        humidity = round(self.bme280.humidity)
```

```
    - if you built from source, your compiler versions and ideally a build log

- If you're working with a numpy git repository, try `git clean -xdf`
  (removes all files not under version control) and rebuild numpy.

Note: this error has many possible causes, so please don't comment on
an existing issue about this - open a new one instead.

Original error was: libf77blas.so.3: cannot open shared object file: No such file o
r directory

>>> 
```

Finished

finished

References

- Visual Studio Code
- Python
- Raspberry Pi
- Flask