

## **Documentation Set-Up Guide**

This document will walk you through my preferred steps to set up your IDE. In this guide we will be using Vscode and Python. Other tools needed are Poetry and Sphinx. All these tools will be covered, including installation and setup, later in the document.

### **Contents:**

- **Step 1: Install your IDE (Vscode)**
- **Step 2: Install your programming language (Python)**
- **Step 3: Install Necessary Extensions**
- **Step 4: Create a folder to hold your files**
- **Step 5: Open the folder**
- **Step 6: Create a file**
- **Step 7: Running your file**
- **Step 8: Installing other tools (Poetry)**
- **Step 9: Installing other tools (Sphinx)**
- **Step 10: Using Sphinx**

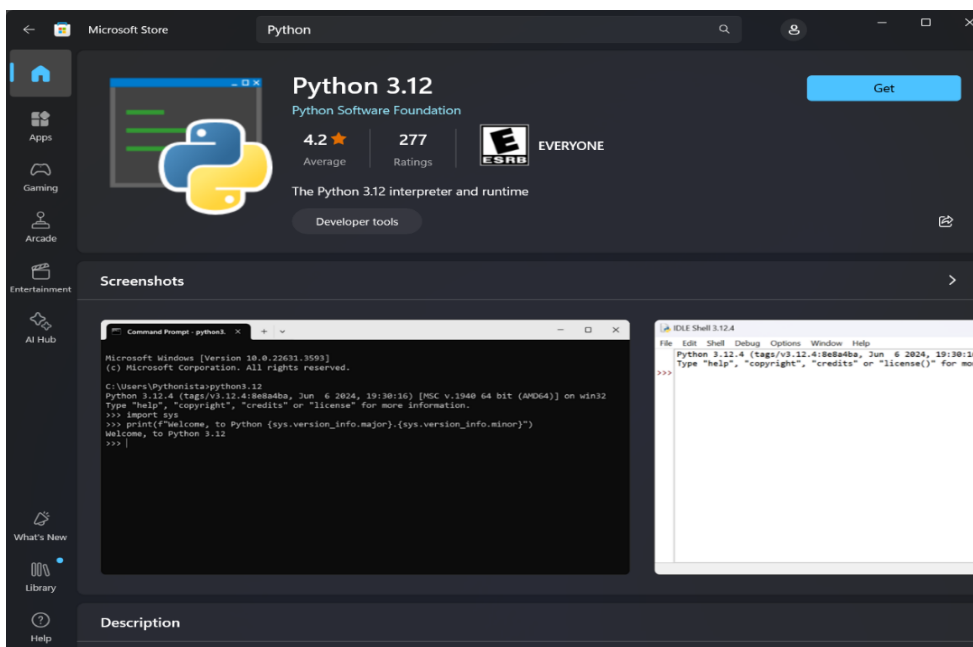
## **Documentation Set-Up Guide**

### **Step 1: Install your IDE (Vscode)**

Navigate to Vscode's download site (<https://code.visualstudio.com/download>) and install the version that matches your operating system. The screenshots that will be included are from a Windows machine. After installing Vscode, the install wizard will guide you through the installation process.

### **Step 2: Install your programming language (Python)**

Your IDE needs a language to use, so you will need to install it. For Python, you can navigate to the Python website and install the latest version (<https://www.python.org/downloads/>) or for Windows users, navigate to the Microsoft Store and install it from there.

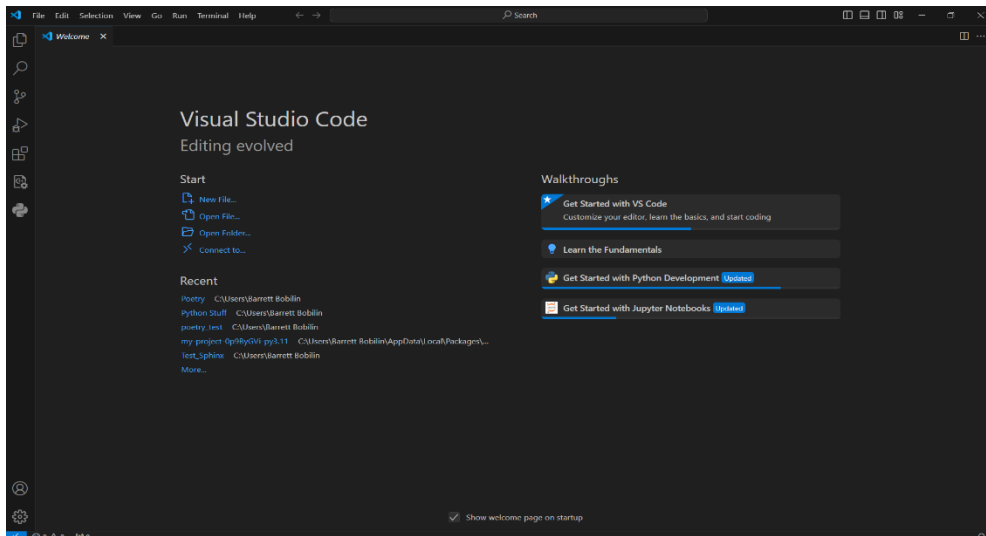


### **Step 3: Install Necessary Extensions**

On start up, your Vscode should look like this:

# Documentation Set-Up Guide

Note: There may be welcome tabs open, feel free to read them, then close them.



On the left side of the screen, you will see the Icon that looks like a building block. This is where you will install your extensions. Click on it, and in the search bar on the top left, type “Python”. A list of options will pop up. Click the top one. It will be by Microsoft.com. Click the install button.



## Documentation Set-Up Guide

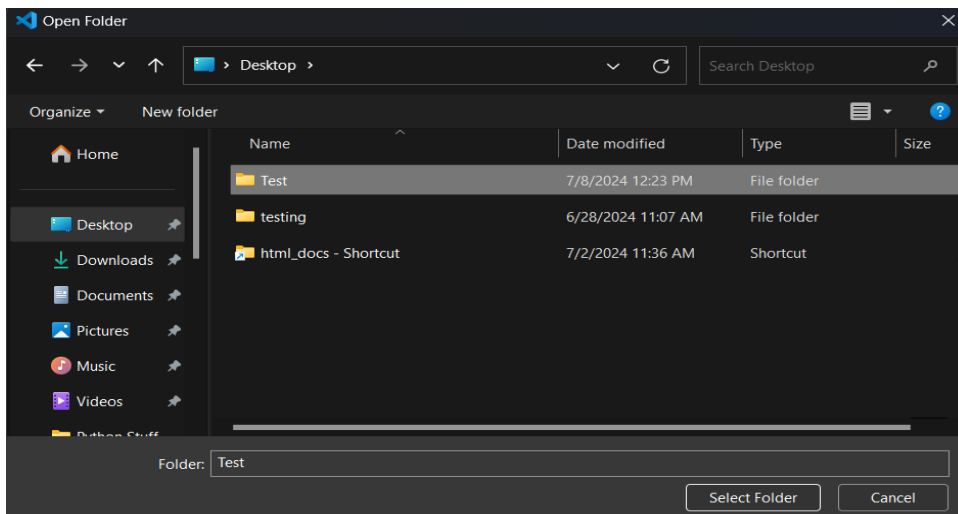
Once Python and its extensions are installed you are ready to create a file. Click the Explorer Icon. It looks like two files on top of each other, located above the Extensions tab. This will bring you back to your file menu.

### Step 4: Create a folder to hold your files

Navigate to your desktop screen, right click, click New, then click Folder. A folder will appear, name it whatever you wish. I named it “test”.

### Step 5: Open the folder

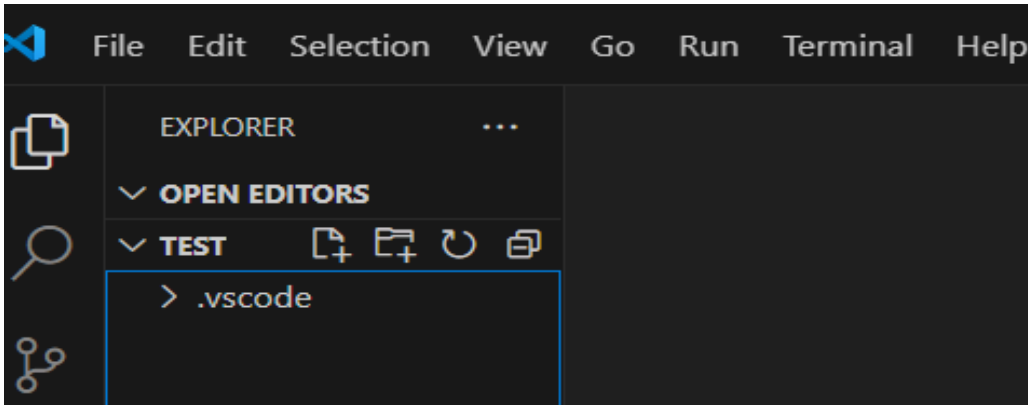
In Vscode, click “File” on the top left, then select “Open Folder”. A folder selection screen will pop up, on the left side of that screen, scroll until you find “Desktop”, click it. After that, you should see your “Test” folder.



### Step 6: Create a file

Once your folder is created, selected, and opened, you’ll need to create a file. When your mouse goes into the Explorer tab, some icons will appear next to your “Test” folder. Click the icon that looks like a file with a plus (+) sign.

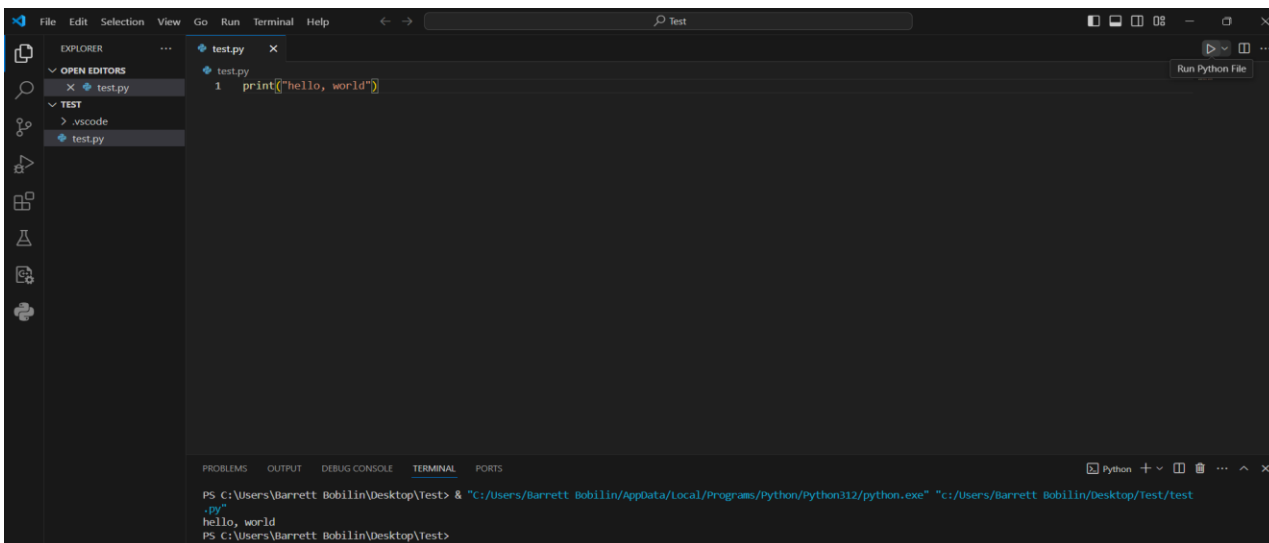
## Documentation Set-Up Guide



A box will pop up prompting you for a file name. I will name mine “test.py”. The “.py” signifies that it is a Python file. Other file names like .txt or .html will have different uses but won’t allow us to run Python code. In my file I will write some basic code: `print(“hello, world”)`.

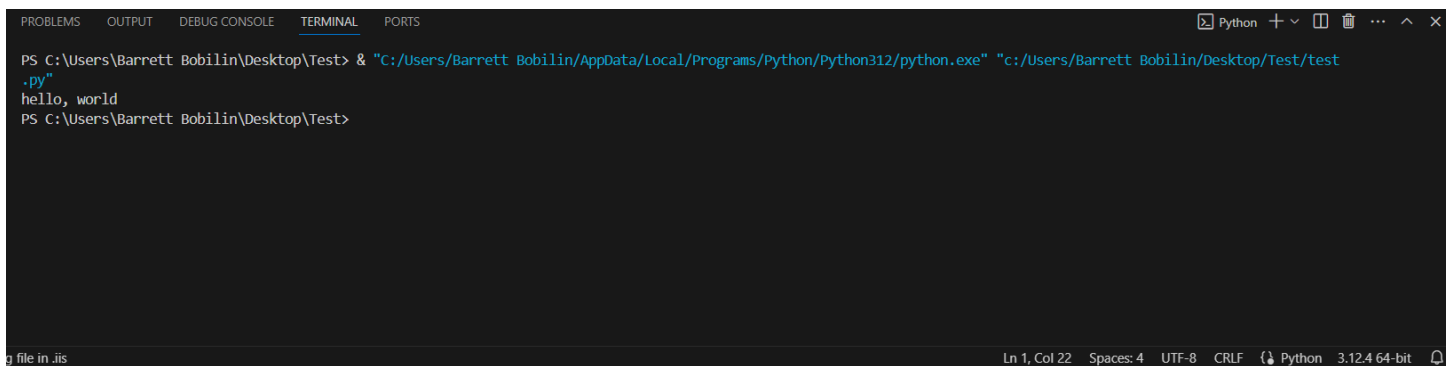
### Step 7: Running your file

After you’ve written your code and corrected your errors, you can run your code. Click the Play button on the top right of your screen, this will run your code and generate the response in terminal the bottom of your screen.



## Documentation Set-Up Guide

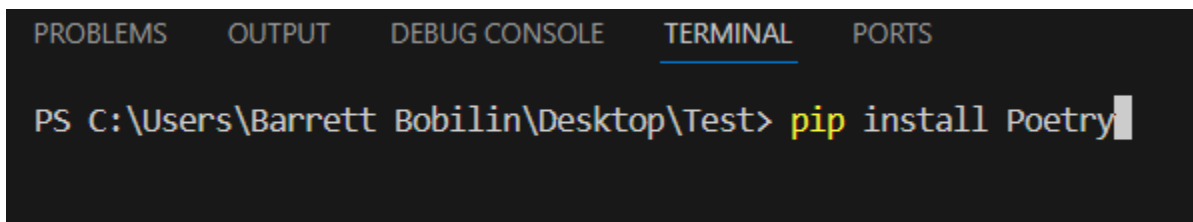
Sometime Vscode will not auto-select the language you wish to use, despite making it a “.py” file. To fix this, there is a section at the bottom right of the screen that says {}Python. If you don’t see this, click the area and type Python in the drop-down box that appears at the top of your screen.



A screenshot of a VS Code terminal window. The terminal title bar shows 'Python' with a dropdown arrow, a maximize icon, a close icon, and a refresh icon. The terminal content shows a PowerShell prompt 'PS C:\Users\Barrett Bobilin\Desktop\Test>' followed by a command to run a Python script: '& "C:/Users/Barrett Bobilin/AppData/Local/Programs/Python/Python312/python.exe" "c:/Users/Barrett Bobilin/Desktop/Test/test.py"'. The output of the script is 'hello, world'. The terminal status bar at the bottom shows 'Ln 1, Col 22', 'Spaces: 4', 'UTF-8', 'CRLF', and 'Python 3.12.4 64-bit'.

### Step 8: Installing other tools (Poetry)

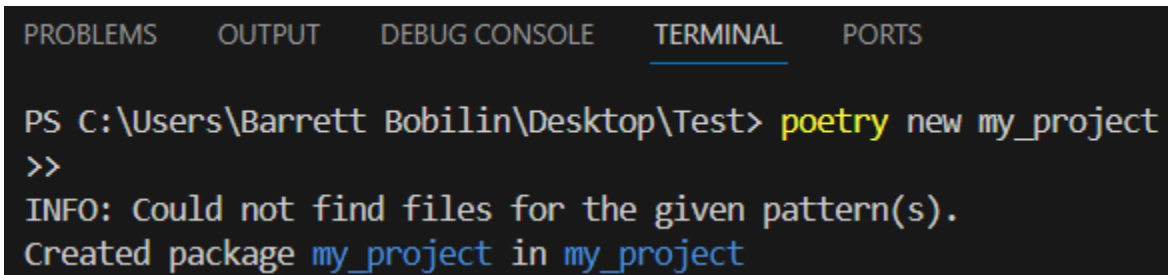
As stated at the beginning of this file, there are other tools and libraries we will need to install. In the terminal at the bottom of your screen type “pip install Poetry”.



A screenshot of a VS Code terminal window. The terminal title bar shows 'Python' with a dropdown arrow, a maximize icon, a close icon, and a refresh icon. The terminal content shows a PowerShell prompt 'PS C:\Users\Barrett Bobilin\Desktop\Test>' followed by the command 'pip install Poetry'. The terminal status bar at the bottom shows 'Ln 1, Col 22', 'Spaces: 4', 'UTF-8', 'CRLF', and 'Python 3.12.4 64-bit'.

If you don’t have a terminal open, click “Terminal” on the top left of your screen and click “New Terminal” and run the code. After this code is run, you will have installed Poetry. Poetry allows us to keep track of the tools we use and separate them from other libraires we have installed. In your terminal type “poetry new my\_project” , this will create a new file controlled by Poetry.

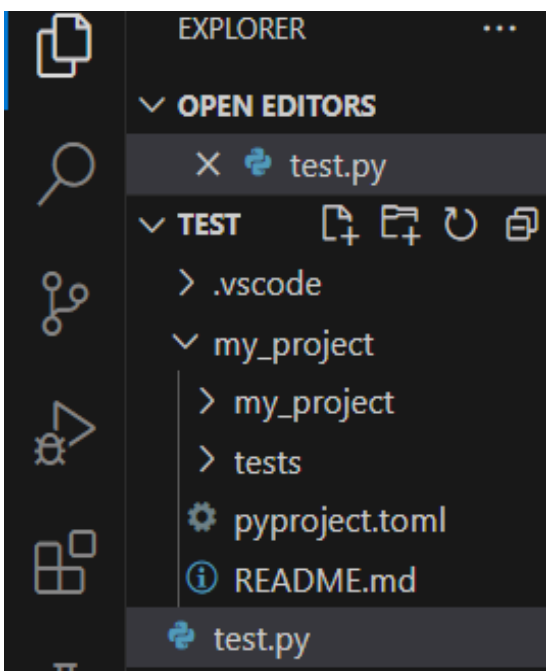
## Documentation Set-Up Guide



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

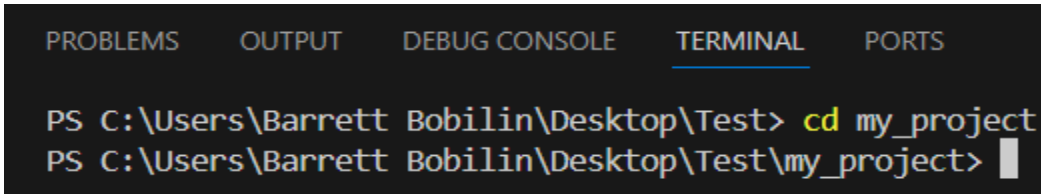
PS C:\Users\Barrett Bobilin\Desktop\Test> poetry new my_project
>>
INFO: Could not find files for the given pattern(s).
Created package my_project in my_project
```

You'll notice on the left side of your screen; the new folder has been added to the parent "Test" folder.



Navigate to this folder by typing "cd my\_project" in the terminal. This should update your file path in the terminal.

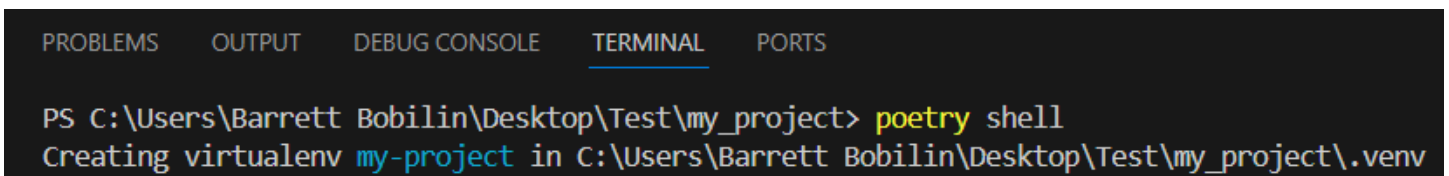
## Documentation Set-Up Guide



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  
PS C:\Users\Barrett Bobilin\Desktop\Test> cd my_project  
PS C:\Users\Barrett Bobilin\Desktop\Test\my_project> |
```

Notice how “my\_project” has been added to our terminal. That means we are now operating in the “my\_project” folder. If you wish to leave the folder, type “cd ..” (‘cd’ with two periods).

Now we need to activate Poetry. We do this by typing “poetry shell”.

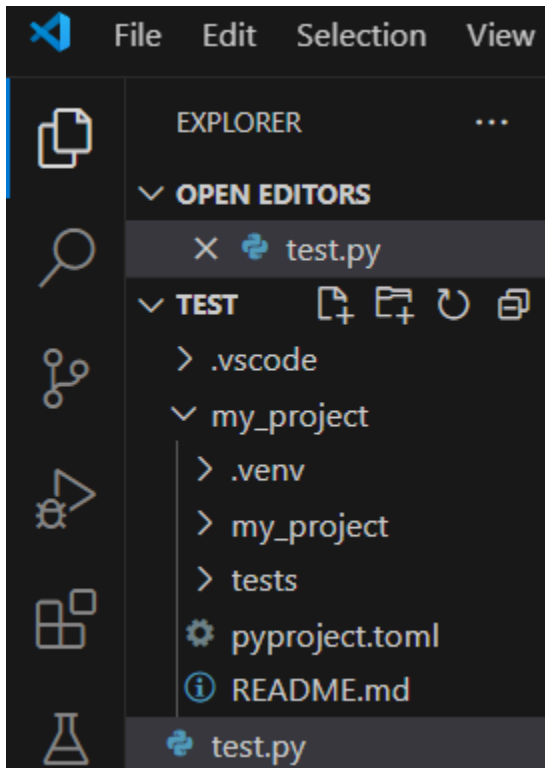


```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  
PS C:\Users\Barrett Bobilin\Desktop\Test\my_project> poetry shell  
Creating virtualenv my-project in C:\Users\Barrett Bobilin\Desktop\Test\my_project\.venv
```

After this you will see another folder called “.venv”, this stands for Virtual Environment.

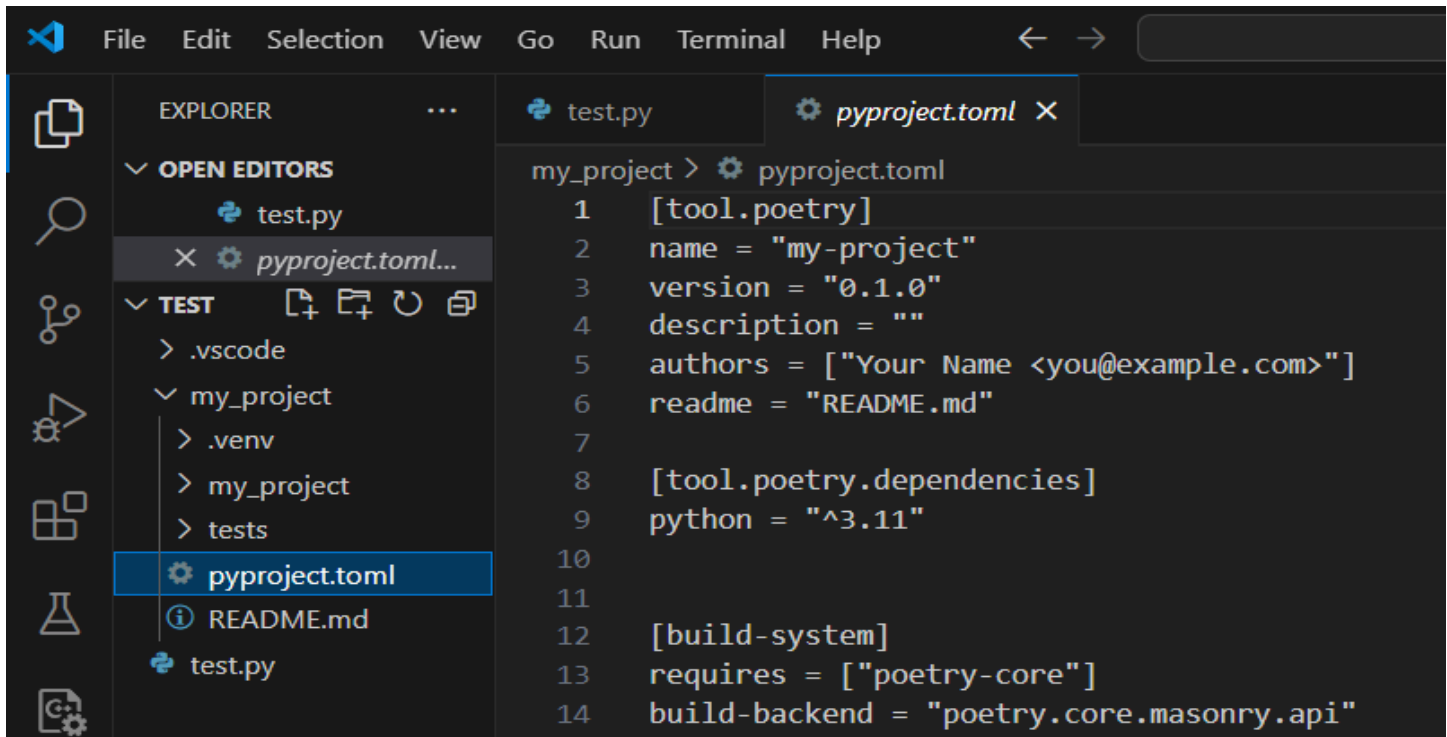


## Documentation Set-Up Guide



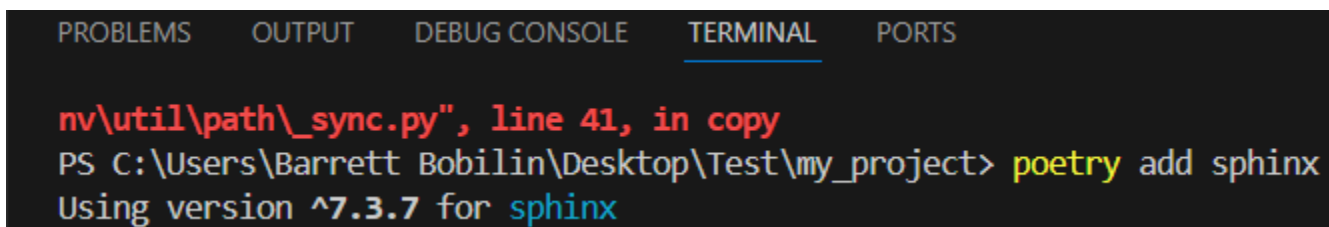
Poetry is now set up and ready for us to use, but first let's look at some of the items poetry gives us. Under the `pyproject.toml`, you will see what poetry has installed. For now it should be pretty empty, but you will see it has python installed: `[tool.poetry.dependencies] python = "^3.11"`. This means python is installed in the environment.

## Documentation Set-Up Guide



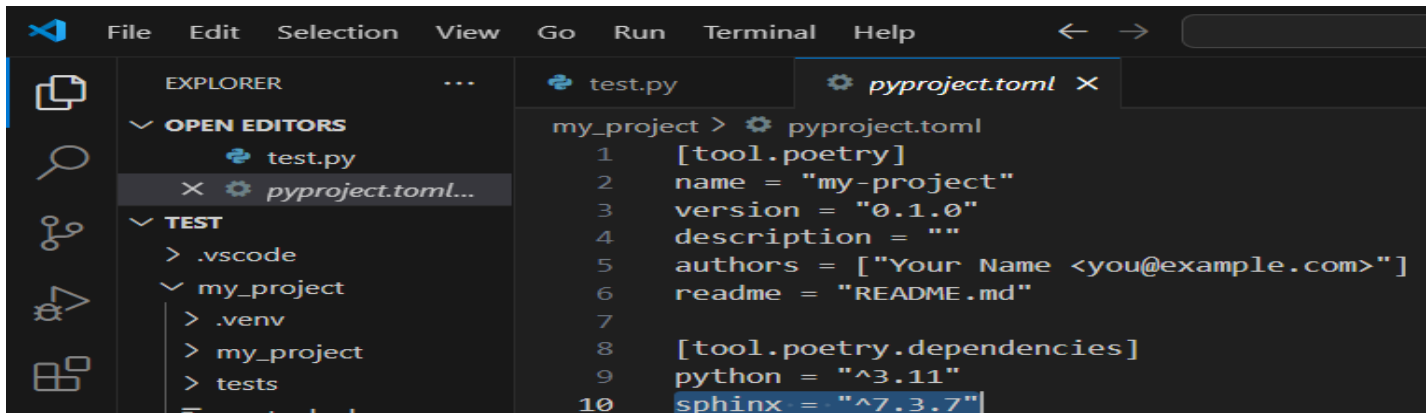
### Step 9: Installing other tools (Sphinx)

To write documentation, we will be using a tool called Sphinx. To add this to your Poetry project, run the command “poetry add sphinx” in your terminal.



If you return to your pyproject.toml file, you will notice that poetry has updated the required dependencies file automatically. This is a big reason why Poetry is so popular.

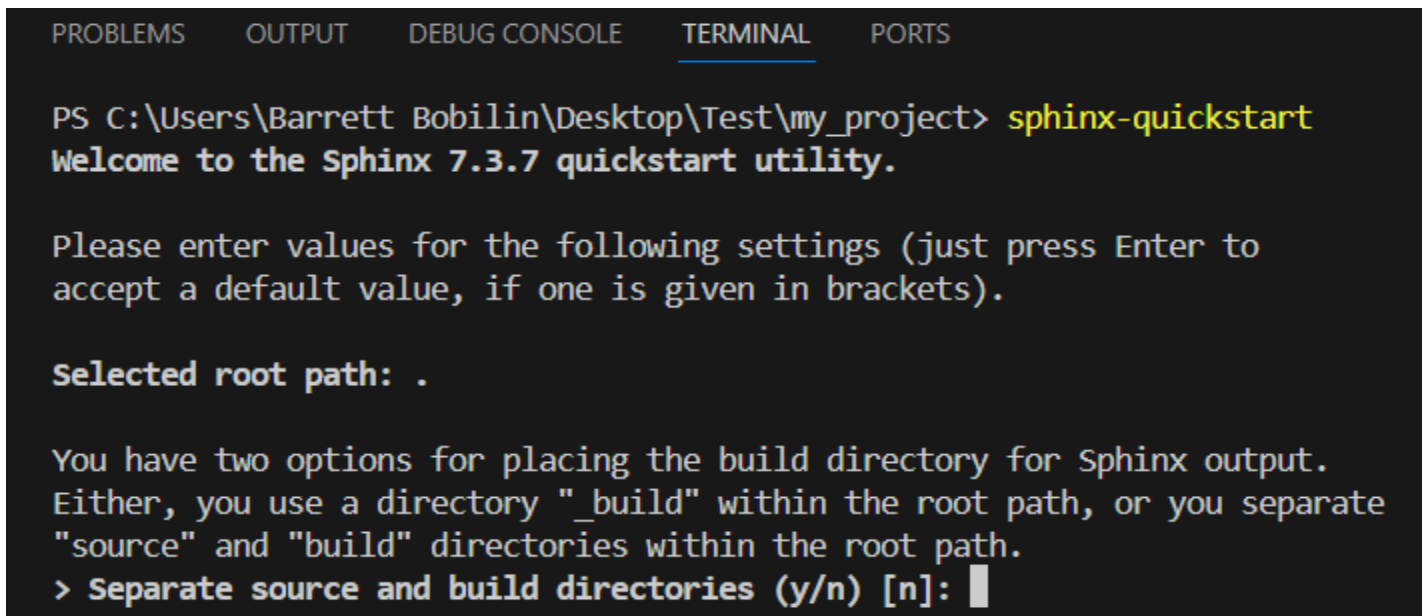
## Documentation Set-Up Guide



### Step 10: Using Sphinx

Sphinx is a powerful documentation builder. To start it, use the command “sphinx-quickstart” in your terminal.

Sphinx will then prompt you for set up instructions, we will be using the defaults.



## Documentation Set-Up Guide

Separating the source and build directories means it takes the files you wish to create from the “source” folder, builds them (typically into an html file) and then puts these built files into a folder called “build”. Just hit enter for this. It will prompt you for the project name, author name, and version. I entered “Sphinx\_test”, “Barrett”, “0.1” for each of these fields, respectively. It will then ask which language you wish to use. English is the default, so if you wish to use English, just hit Enter. This will complete the Sphinx setup.

```
The project name will occur in several places in the built documentation.
> Project name: Sphinx_test
> Author name(s): Barrett
> Project release []: 0.1

If the documents are to be written in a language other than English,
you can select a language here by its language code. Sphinx will then
translate text that it generates into that language.

For a list of supported codes, see
https://www.sphinx-doc.org/en/master/usage/configuration.html#confval-language.
> Project language [en]:

Creating file C:\Users\Barrett Bobilin\Desktop\Test\my_project\conf.py.
Creating file C:\Users\Barrett Bobilin\Desktop\Test\my_project\index.rst.
Creating file C:\Users\Barrett Bobilin\Desktop\Test\my_project\Makefile.
Creating file C:\Users\Barrett Bobilin\Desktop\Test\my_project\make.bat.

Finished: An initial directory structure has been created.

You should now populate your master file C:\Users\Barrett Bobilin\Desktop\Test\my_project\index.rst and create other documentation
source files. Use the Makefile to build the docs, like so:
    make builder
where "builder" is one of the supported builders, e.g. html, latex or linkcheck.

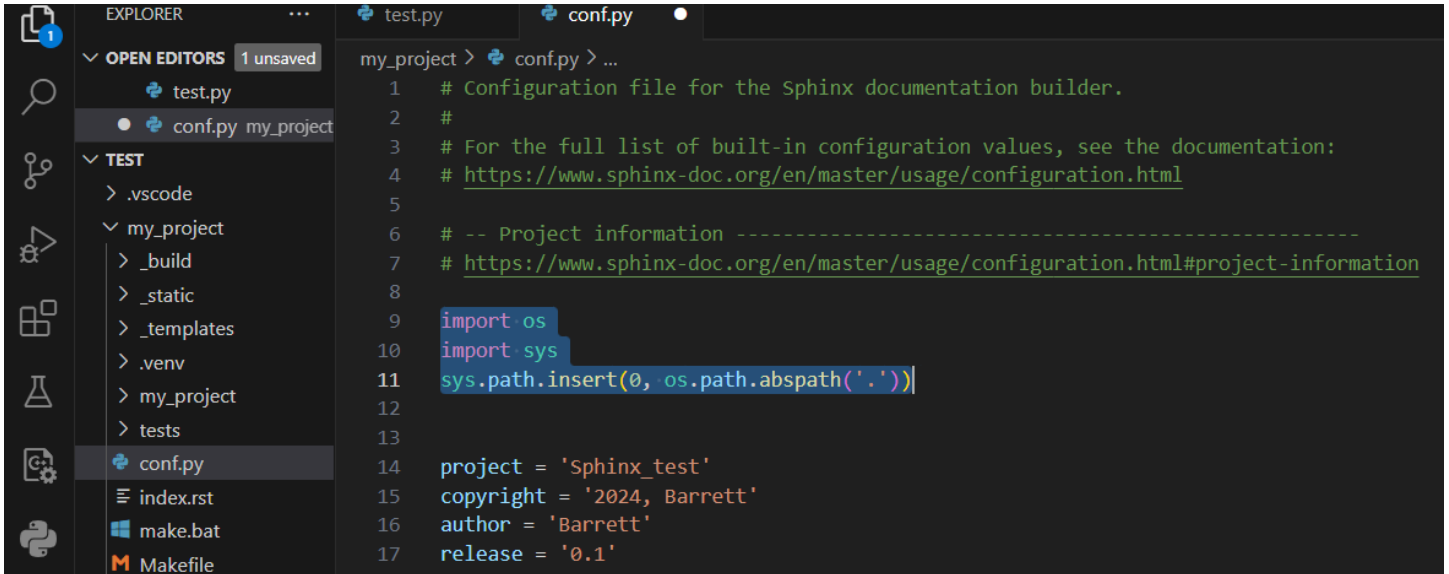
PS C:\Users\Barrett Bobilin\Desktop\Test\my_project>
```

After completing this, you will see new files and folders have been added to your project such as `_build`, `_static`, `conf.py`, `index.rst`, and others. `_build` is where our built files will be placed after we run Sphinx, more on that later. The `conf.py` prompt tells sphinx how to run, and `index.rst` is the master file that sphinx will look at to build your documentation.

Let's edit our `conf.py` file with some basic code that will allow sphinx to automatically read our files and generate our documentation. Above the project and author name we will add this code:

```
import os
import sys
sys.path.insert(0, os.path.abspath('.'))
```

## Documentation Set-Up Guide



```
my_project > conf.py > ...
1  # Configuration file for the Sphinx documentation builder.
2  #
3  # For the full list of built-in configuration values, see the documentation:
4  # https://www.sphinx-doc.org/en/master/usage/configuration.html
5
6  # -- Project information -----
7  # https://www.sphinx-doc.org/en/master/usage/configuration.html#project-information
8
9  import os
10 import sys
11 sys.path.insert(0, os.path.abspath('.'))
12
13
14 project = 'Sphinx_test'
15 copyright = '2024, Barrett'
16 author = 'Barrett'
17 release = '0.1'
```

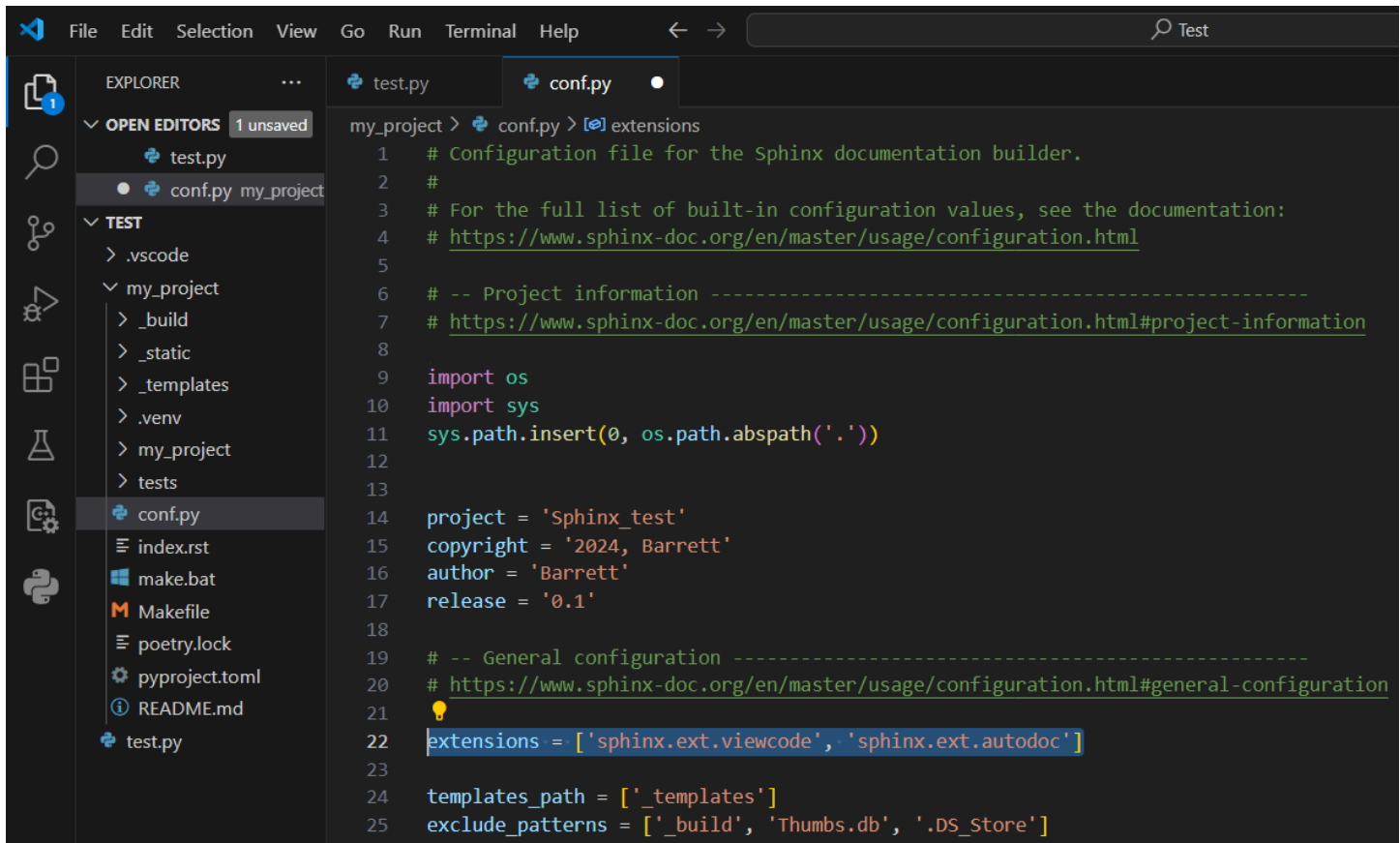
This helps sphinx recognize what folder it needs to look in. The ‘.’ tells sphinx to look in the current directory. If sphinx isn’t finding your files, it can help to switch the single dot ‘.’ to two dots ‘..’ to tell sphinx to look in the parent directory.

Next, we need to add some extensions to the conf.py file to give sphinx the ability to parse, or read, our files.

We will add this code:

```
extensions = ['sphinx.ext.viewcode', 'sphinx.ext.autodoc']
```

## Documentation Set-Up Guide



The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left displays the project structure for 'my\_project', including folders like '\_build', '\_static', and '\_templates', and files like 'index.rst', 'make.bat', 'Makefile', 'poetry.lock', 'pyproject.toml', 'README.md', and 'test.py'. The 'conf.py' file is selected and open in the main editor. The code in 'conf.py' is a Sphinx configuration file. It includes comments for project information and general configuration, and defines the 'extensions' list with 'sphinx.ext.viewcode' and 'sphinx.ext.autodoc'. The 'templates\_path' is set to ['\_templates'] and 'exclude\_patterns' is set to ['\_build', 'Thumbs.db', '.DS\_Store'].

```

my_project > conf.py > extensions
1  # Configuration file for the Sphinx documentation builder.
2  #
3  # For the full list of built-in configuration values, see the documentation:
4  # https://www.sphinx-doc.org/en/master/usage/configuration.html
5
6  # -- Project information -----
7  # https://www.sphinx-doc.org/en/master/usage/configuration.html#project-information
8
9  import os
10 import sys
11 sys.path.insert(0, os.path.abspath('.'))
12
13
14 project = 'sphinx_test'
15 copyright = '2024, Barrett'
16 author = 'Barrett'
17 release = '0.1'
18
19 # -- General configuration -----
20 # https://www.sphinx-doc.org/en/master/usage/configuration.html#general-configuration
21
22 extensions = ['sphinx.ext.viewcode', 'sphinx.ext.autodoc']
23
24 templates_path = ['_templates']
25 exclude_patterns = ['_build', 'Thumbs.db', '.DS_Store']

```

After doing this click Ctrl + S to save the file.

There is more to do but let's see what happens if we try to run sphinx as it is. We will run the code:

```
sphinx-build -b html my_project _build
```

- “sphinx-build” is the command to run sphinx
- “-b html” tells sphinx to build the files as html files
- “my\_project” is the source folder sphinx looks in
- “\_build” is the folder sphinx puts the finished files in

After sphinx runs, it will give a success message:

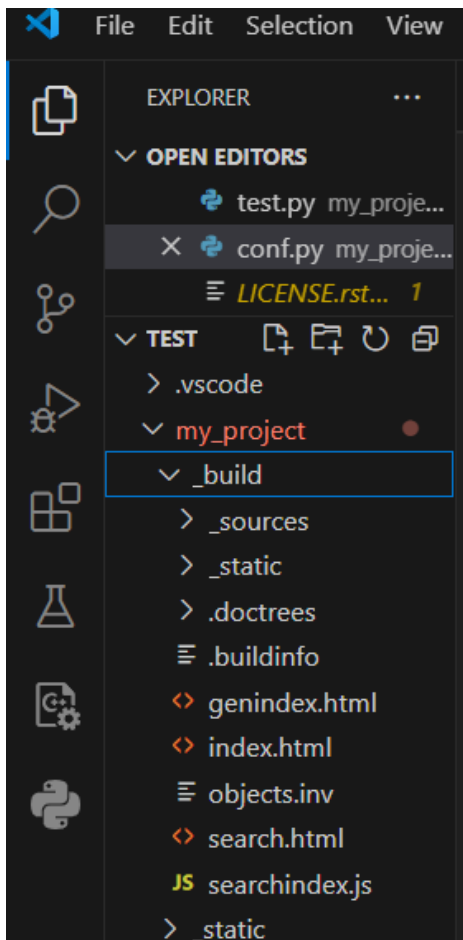
## Documentation Set-Up Guide

```
PROBLEMS 17 OUTPUT DEBUG CONSOLE TERMINAL PORTS

checking consistency... done
preparing documents... done
copying assets... copying static files... done
copying extra files... done
done
writing output... [100%] index
generating indices... genindex done
writing additional pages... search done
dumping search index in English (code: en)... done
dumping object inventory... done
build succeeded, 1 warning.

The HTML pages are in _build.
PS C:\Users\Barrett Bobilin\Desktop\Test\my_project>
```

Here you can see all the files Sphinx created in the `_build` folder:



## Documentation Set-Up Guide

Note: if you get an error when running sphinx along the lines of: Configuration error: config directory doesn't contain a conf.py file (C:\Users\Barrett Bobilin\Desktop\Test\my\_project\my\_project) . This can mean that sphinx cannot find your file. To fix this you can create another folder, I named mine “file\_holder”. Put the files index.rst , conf.py , and the file you want to create documentation from, in my case test.py, into the “file\_holder” folder. Now after running `sphinx-build -b html file_holder _build` , my files are successfully added to \_build

```
PS C:\Users\Barrett Bobilin\Desktop\Test\my_project> sphinx-build -b html file_holder _build
Running Sphinx v7.3.7
WARNING: html_static_path entry '_static' does not exist
building [mo]: targets for 0 po files that are out of date
writing output...
building [html]: targets for 1 source files that are out of date
updating environment: [new config] 1 added, 0 changed, 0 removed
reading sources... [100%] index
looking for now-outdated files... none found
pickling environment... done
checking consistency... done
preparing documents... done
copying assets... copying static files... done
copying extra files... done
done
writing output... [100%] index
generating indices... genindex done
highlighting module code...
writing additional pages... search done
dumping search index in English (code: en)... done
dumping object inventory... done
build succeeded, 1 warning.

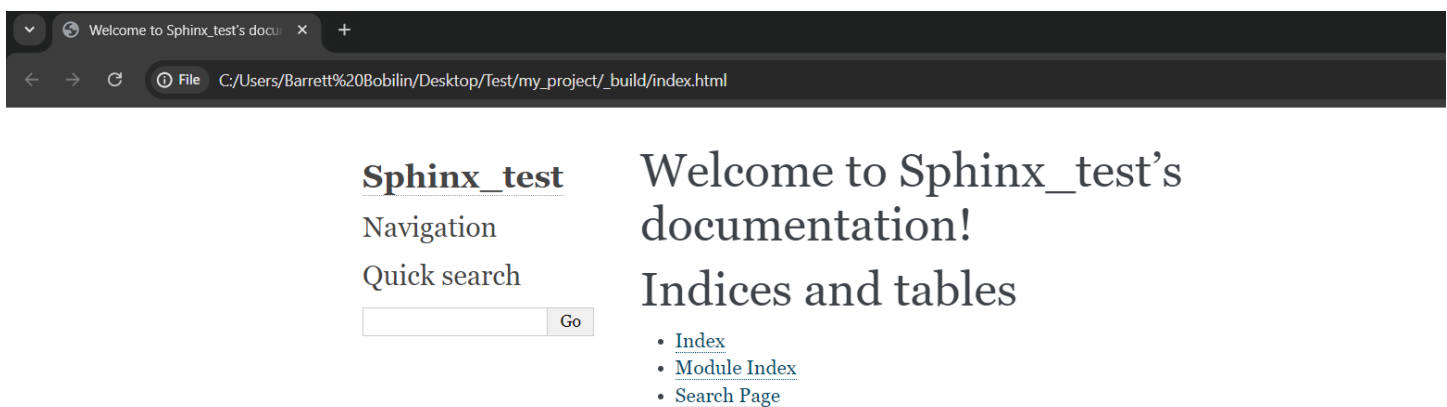
The HTML pages are in _build.
PS C:\Users\Barrett Bobilin\Desktop\Test\my_project> |
```

To open these files, go back to your desktop, open the “test” folder, then the “my\_project” folder then navigate to the “\_build” folder. In the build folder there will be a list of files:



## Documentation Set-Up Guide

Notice some have the Google Chrome icon. You may have a different icon, but under the “Type” column you will see some files are named “html”. Click any one of these html files to open your documentation project. I used “index”.

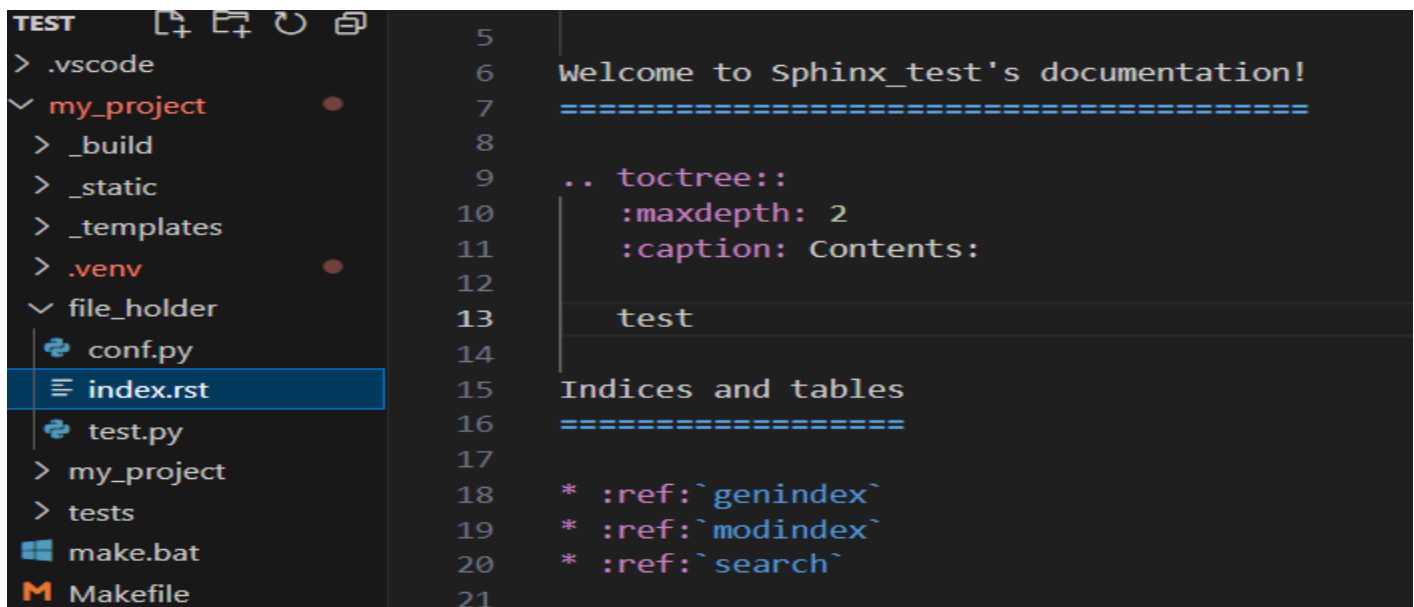


## Documentation Set-Up Guide

This is the bare bones of your documentation. Notice how it has your name in the bottom left and file pathway in the URL bar.

Now let's add some code and let sphinx document it for us. Navigate back to Vscode and open your index.rst file. We need to add the name of the file we will be documenting. Since I am documenting the file "test.py", I will add "test" to my index file. The file type ".py" isn't included in this part. Index.rst might not let you edit it, if this happens, right click the file in the explorer tab and click "Rename" then change the name to "index.txt".

This should let you edit the file. Make sure it follows this indentation:



```

5
6 Welcome to Sphinx_test's documentation!
7 =====
8
9 .. toctree::
10     :maxdepth: 2
11     :caption: Contents:
12
13     test
14
15 Indices and tables
16 =====
17
18 * :ref:`genindex`
19 * :ref:`modindex`
20 * :ref:`search`
21

```

Now we need to create an rst file for our code file. Think of this as a "go-between" of our code file (test.py) and our master file (index.rst). Make sure you name this .rst file the same as your code file. Because my code file is called "test.py" I will name the .rst file "test.rst". Copy this code into the file:

```

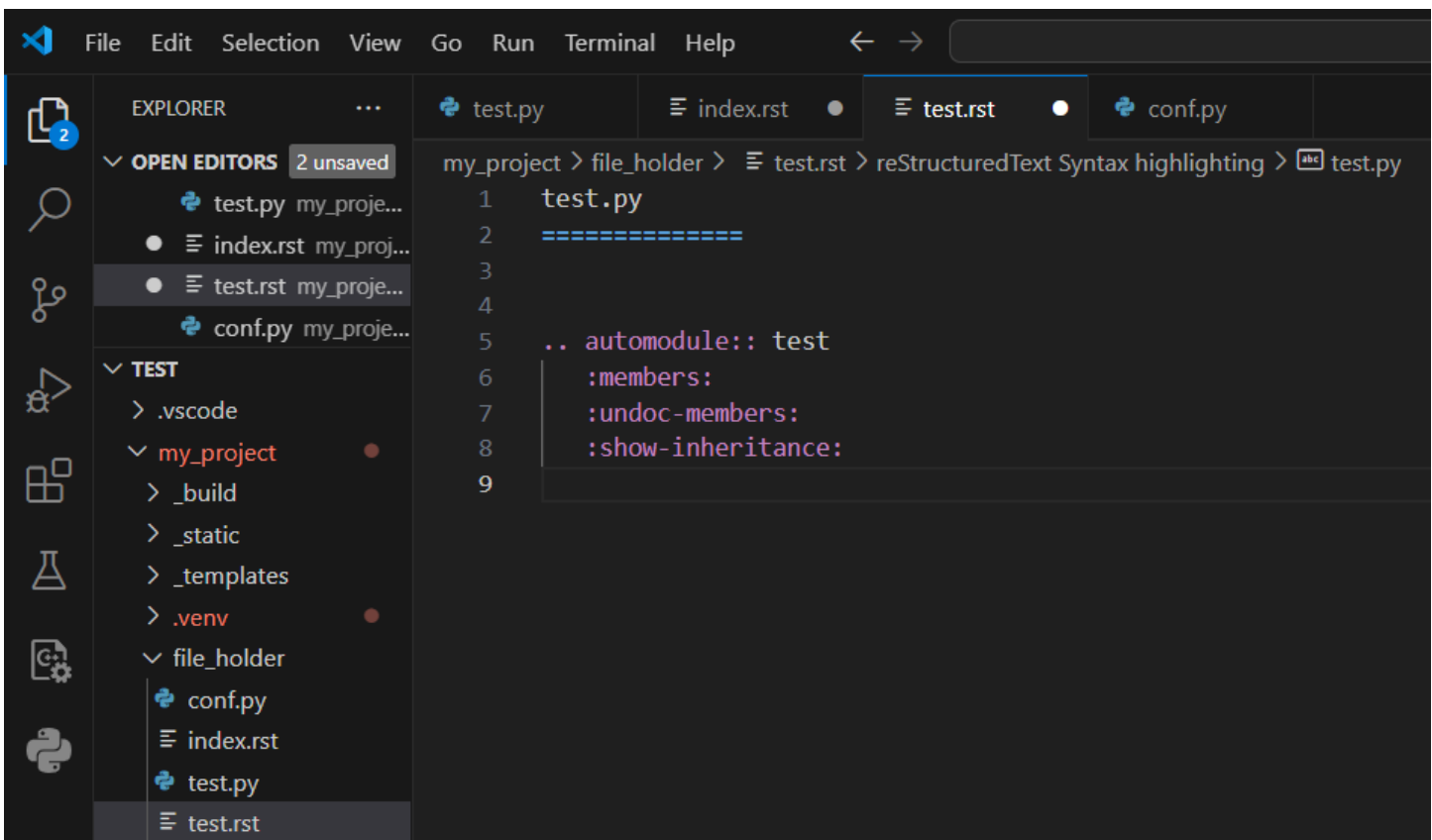
test.py
=====

```

## Documentation Set-Up Guide

```
.. automodule:: test
    :members:
    :undoc-members:
    :show-inheritance:
```

At the top of the file, put the exact name of your code file, in our case, “test.py”. Make sure it has enough “=” underneath it to fully cover the file name. “.. automodule::” will let sphinx analyze our file and create documentation. Next to this line, input your name, with out the “.py”. Again, make it a .txt if it doesn’t allow you to edit it. Make sure you put this .rst file with your other files so **Sphinx** can detect it.



## Documentation Set-Up Guide

Back in our test.py file, I will edit it to add some docstrings. Docstrings are comments enclosed in three double quotes (“”) or three single quotes (‘), I use double quotes. Back in your test.py file I have added some documentation and added our print() into a function called “def test():”

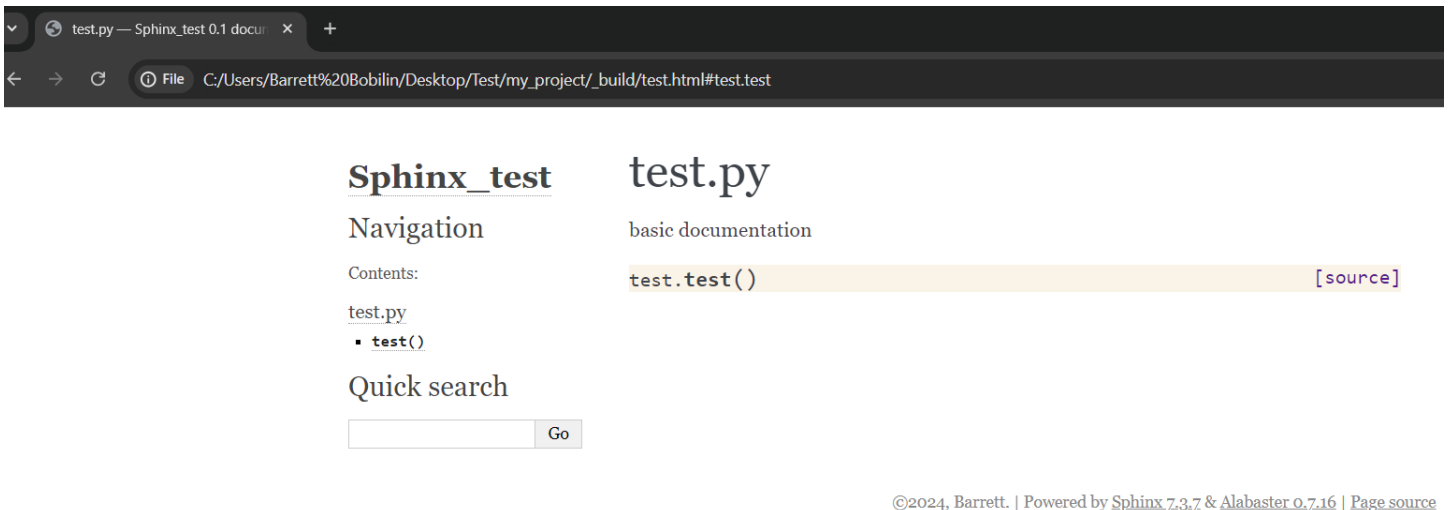
```
"""basic documentation"""
def test():
    print("hello, world")
```

Back on our desktop I have deleted the files contained in \_build. This isn’t good practice, but while learning Sphinx it helps to identify which files you just created. After saving all our files (conf.py, index.rst, test.py, test.rst) with Ctrl + S, you can now run the Sphinx build command again. Accessing our \_build folder will show us some new files:

Name	Date modified	Type	Size
.doctrees	7/9/2024 10:27 AM	File folder	
_modules	7/9/2024 10:27 AM	File folder	
_sources	7/9/2024 10:27 AM	File folder	
_static	7/9/2024 10:27 AM	File folder	
.buildinfo	7/9/2024 10:27 AM	BUILDINFO File	1 KB
genindex	7/9/2024 10:27 AM	Chrome HTML Do...	4 KB
index	7/9/2024 10:27 AM	Chrome HTML Do...	4 KB
objects.inv	7/9/2024 10:27 AM	INV File	1 KB
py-modindex	7/9/2024 10:27 AM	Chrome HTML Do...	4 KB
search	7/9/2024 10:27 AM	Chrome HTML Do...	3 KB
searchindex.js	7/9/2024 10:27 AM	JSFile	2 KB
test	7/9/2024 10:27 AM	Chrome HTML Do...	4 KB

## Documentation Set-Up Guide

Click on your “test” html file at the bottom to access your documentation. It should bring up your code with the documentation.



Notice how our docstrings “““ basic documentation ””” has been added as plain text right below the file name (test.py). Clicking the [source] button allows us to see our source code:



You have now completed your documentation!