Task 5: Setting up a virtual environment for our class

When you installed Python on your computer (see setup2_python) you installed it globally on your computer. But, working in a global environment can be tricky because different versions of python packages may not play nicely together. To avoid any conflicts, developers usually create a virtual environment for each project and then install the packages they need in that virtual environment. When you then run a Python program within that environment, you know that it's running against only those specific packages.

Setting up a virtual environment in our be434-spring-2023 folder

We are going to create a virtual environment in VS Code for our class. We will install several python packages that will help us to test, debug and format our code in this virtual environment.

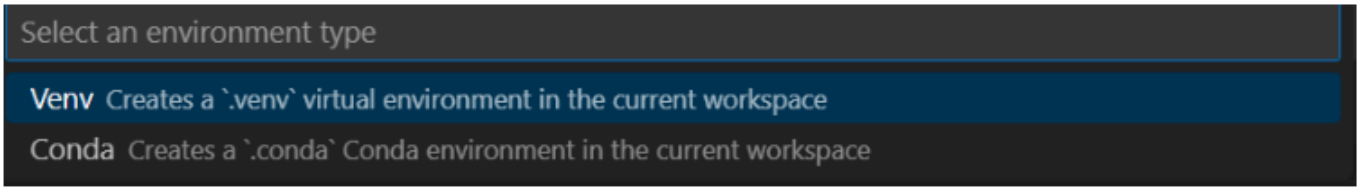
- From the VS Code menu bar select File -> Open Folder -> Then navigate to the be433-spring-2023 folder in GitHub (mine is in Documents/GitHub)

Step 1: Create a virtual environment in the BE434-spring-2023 workspace (directory)

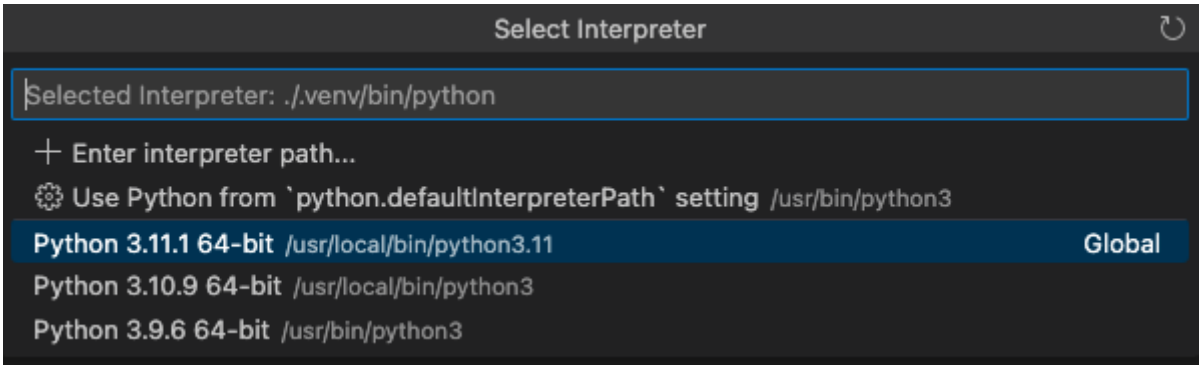
Next, we will create a virtual environment (venv) in VS Code to install all of the Python packages we will need for our class. venv allows you to manage separate package installations for different projects and is installed with Python 3 by default.

- From within VS Code, you can create non-global environments, using virtual environments by opening the Command Palette (⇧⌘P), start typing "Python: Create Environment command" to search, and then select the command.

The command presents a list of environment types: Venv or Conda. Select Venv.



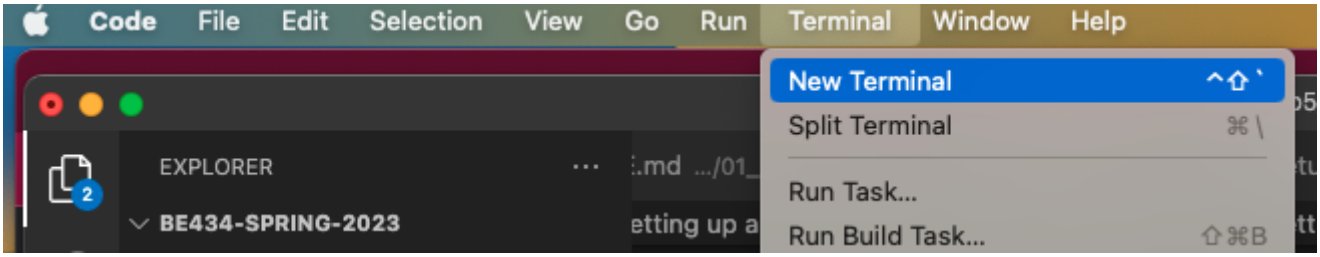
The the command presents a list of interpreters that can be used as a base Python for the new virtual environment. Select Python Version 3.11.1 that we just installed.



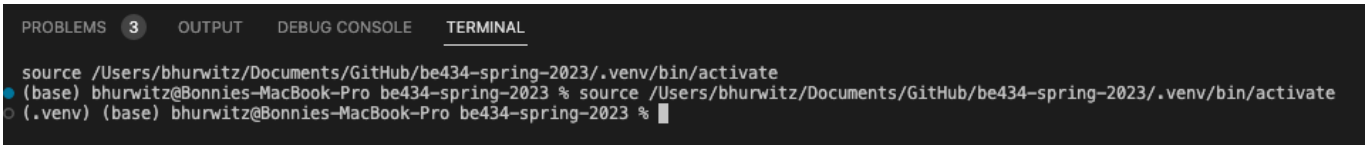
Step 2: Install the Python modules we need for this class.

Once you have created your vitual environment, you can install Python modules in that environment. In Python, modules are how you obtain any number of useful code libraries. For example, in this class we will use several modules that will help us to test, debug and format code. These are all contained in the requirement.txt file in the docs directory in the be434-spring-2023 Github repository.

First we need to open a terminal window in VS Code:



Note, you should already have the be434-spring-2023 directory open in VS Code and it should automatically activate the virtual environment when you open the terminal



If this is not the case, be sure to open the be434-spring-2023 directory in VS Code and then activate the cirtual machine

For a Mac, you will need to enter the following commands in the terminal:

```
source .venv/bin/activate
```

For a PC, you will need to enter the following commands:

```
.venv\scripts\activate
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope Process
```

Now you can install the python modules we need into your virtual environment:

You can do this with a single command (from the be343-spring-2023 folder):

For Mac

```
python3 -m pip install -r ./docs/requirements.txt
```

For PC

```
python -m pip install -r ./docs/requirements.txt
```

Or by installing each one individually...

For a Mac, you will need to enter the following commands in the terminal:

```
python3 -m pip install pytest
python3 -m pip install pylint
python3 -m pip install flake8
python3 -m pip install yapf
python3 -m pip install black
python3 -m pip install mypy
python3 -m pip install pytest-flake8
python3 -m pip install pytest-mypy
python3 -m pip install pytest-pylint
```

For a PC, you will need to enter the following commands:

```
python -m pip install pytest
python -m pip install pylint
python -m pip install flake8
python -m pip install yapf
python -m pip install black
python -m pip install mypy
python -m pip install pytest-flake8
python -m pip install pytest-mypy
python -m pip install pytest-pylint
```

Fixing a small issue with pylint

Now you have installed all of the python modules that we need for testing your code! Congrats! When we start testing our code (in the weeks to come), you might find that "pylint" complains about the variable `rv` (return value) that is in the *test.py* file of each homework. This is a perfectly fine variable name, so to silence this warning, create your own configuration file like so:

```
pylint --generate-rcfile > ~/.pylintrc
```

Then edit that file to add the following line after "MAIN". Note that this should be one continuous line, but I've broken it here for display:

```
disable=too-many-locals,invalid-name,too-many-statements,too-many-
arguments,\
cell-var-from-loop,wrong-import-order
```

```
nano ~/.pylintrc
```

Use ctr-O & return to save the file, and ctr-X to exit