Week 7: An Introduction to the use of IDEs

Learning Objectives

By the end of this module, you should be able to complete the following tasks:

- Use a Command line interface to navigate the folder system of your computer
- Set up a python environment using Conda
- Describe the advantages of using an integrated development environment (IDE)
- Set up a VSCode project
- Describe the features of VSCode

Up to this point, you've been working with our modules using the University of Toronto's JupyterHub web platform, and the simple "Jupyter Classic" user interface for notebooks. While this interface is suitable for beginners, as you perform more advanced computational tasks, you'll begin needing more sophisticated ways of managing multiple files, Python packages (or other programming languages), and executing and debugging your code. Moreover, you'll need a way to do programming on your own computer, rather than just relying on an Internet connection and cloud-based platforms. The first step in this is learning how to directly interact with your operating system.

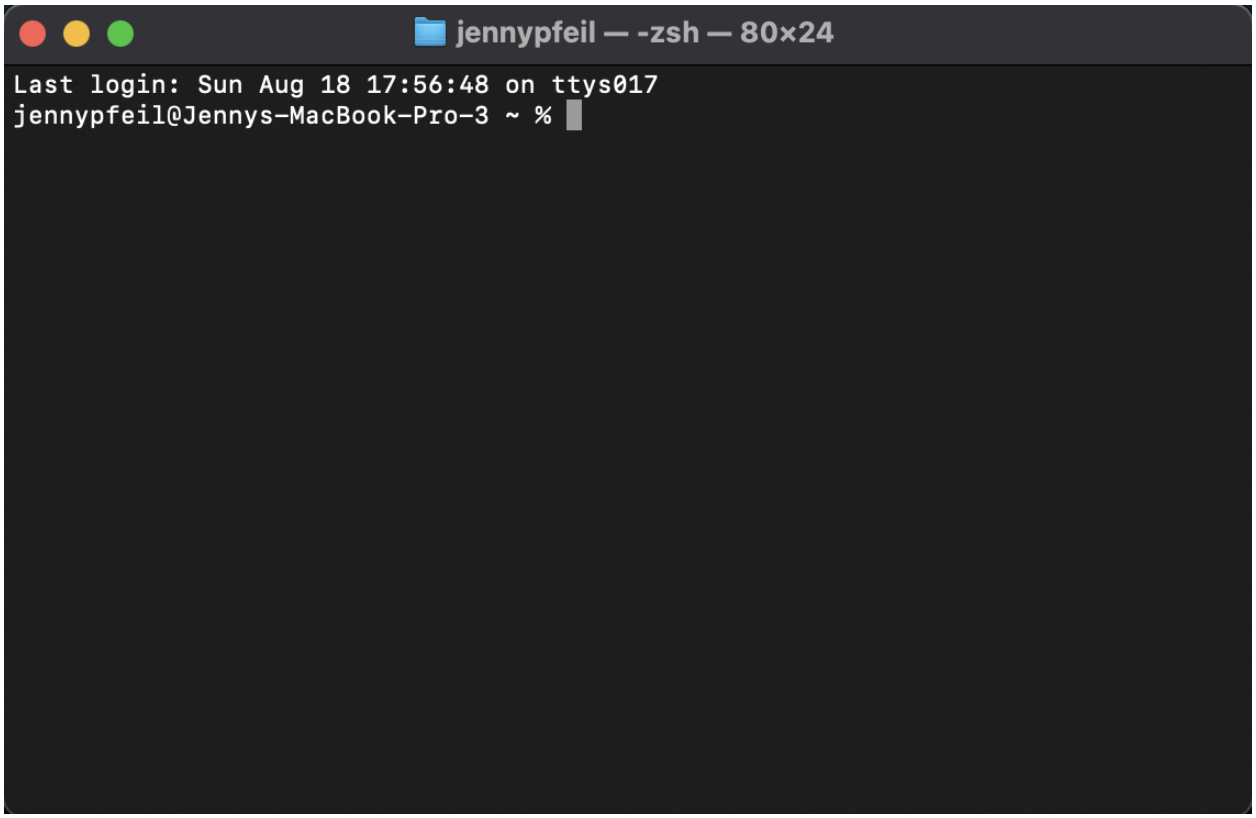
Introduction to CLIs

Command line interfaces (CLIs) are applications on your computer that allow you to communicate directly with the operating system through a text-based user interface. CLIs will differ depending on your operating system; *Terminal* for macOS users and *Command Prompt* for Windows users.

In order to use CLIs, you enter standard commands that the operating system understands. A **command** is a line of text that follows a specific format and instructs a computer to execute a specific function. A single command will often consist of a single string of characters with no spaces, often abbreviating the name for a task or resembling it phonetically. Please note that commands for you to type into the CLI for this module will be denoted in the following format `example_command`. Alternatively you may see a command with the following notation `<command name>` with any text appearing between <> as text that you will need to change before entering the command. Please note that the < and > are also to be removed before entering the command. Commands will often also include multiple options that can be added to modify the command, options are denoted by `-<option name>` To use a command in a CLI, type a command into the user interface when prompted and press enter to execute it.

First let's open the *Terminal* application (icon below) on your computer. This will open a new window, called a "terminal", where you can type in commands for your computer to execute.

The new window should look something like this



You will see your own username @ name of your computer ~ , then a % or $. This will signify that the terminal is ready to take a command. After executing a command and before executing your next command, make sure that you see the % or $ reappear which will mean that the previous command has finished executing and that the CLI is ready to take your next command.

All computer systems are organized into folders, also called **directories**, which can store files or more folders within. Within a computer there often exists multiple users that are represented by separate folders. Each one of these folders will be that user's **home directory**. Within that home directory, each user will have several general standard folders with your home directory like Downloads, Documents, Applications and more. In reference to each other the Downloads directory is a **subdirectory** of the home directory and the home directory is a **parent directory** of the Downloads directory.

First we will be using CLIs to navigate, create, and remove directories on your computer. These tasks can also be done using standard applications that allow you to navigate your folders through a graphical user interface (*Finder* for macOS users and *Folders* for Windows users). However, learning these basic skills with a CLI is the start of performing more complex tasks with a CLI.

Follow the steps below to navigate your computer through the CLI.

1. Type `ls`, then press enter. This command will allow you to list all contents within the current directory. Everytime you open the terminal, you will start in your home directory meaning that you are in the largest parent folder for your user account on your computer and should have several general standard folders with your home directory like Downloads, Documents, Applications and more. You should get something that looks like this.

```
jennypfeil@Jennys-MacBook-Pro-3 ~ % ls
Applications                    Movies
Desktop                         Music
Documents                       Pictures
Downloads                       Projects
Library                         Public
```

2. Type `cd Documents`, then press enter. This command allows you to step into a specific directory. The format of the command to step into a directory is `cd <directory name>`. You should now see that your directory name appears after your username, which means you have successfully entered that directory.

```
jennypfeil@Jennys-MacBook-Pro-3 ~ % cd Documents
jennypfeil@Jennys-MacBook-Pro-3 Documents % ..
```

3. Type `mkdir HMB301_modules`, then press enter. This command allows you to make a directory within the current directory. The format of the command to make a directory is `mkdir <directory name>`.

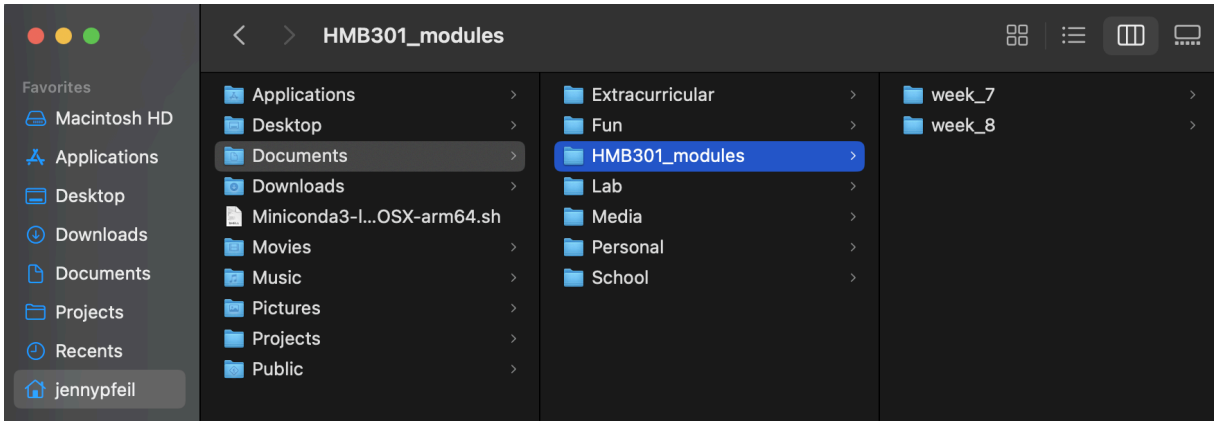
```
jennypfeil@Jennys-MacBook-Pro-3 Documents % mkdir HMB301_modules
```

4. Type `cd HMB301_modules`, then press enter. This will allow you to enter the directory you just created.

```
jennypfeil@Jennys-MacBook-Pro-3 Documents % cd HMB301_modules
jennypfeil@Jennys-MacBook-Pro-3 HMB301_modules %
```

5. Now we will create two more directories within the HMB301_modules directory. Type `mkdir week_7`, then press enter. Then type `mkdir week_8`, then press enter.

You can check these have been correctly added by navigating through the Finder application to find these 2 new directories. Here we can see the two sub directories we created.



6. Type `cd ..` , then press enter. this will allow you to move out of the current directory and into its parent directory. Type `cd ..` , then press enter one more time and you should be back in your home directory. You should now see that nothing appears after your username, which means you have successfully made it back to your home directory.

7. Now we will try to remove some of the directories we just created. Type `rmdir /Users/(username)/Documents/HMB301_modules` , then press enter. This command should not work and you should get an error telling you that you cannot remove directories that are not empty. You can only remove a directory when no content is in it. If you want to remove the whole directory including all its contents, use will need to use an option with that command in the format `rmdir -r <directory name>` where the -r means the computer will remove all files inside the parent directory

```
jennypfeil@Jennys-MacBook-Pro-3 ~ % rmdir /Users/jennypfeil/Documents/HMB301_mod
ules
rmdir: /Users/jennypfeil/Documents/HMB301_modules: Directory not empty
```

8. Now we will try to remove an empty directory. Type `rmdir /Users/(username)/Documents/HMB301_modules/week_7` , then press enter. This should now work. Once again, you can check that the week _7 directory has been correctly added by navigating through the Finders application to find the original HMB_301_modules directory

```
jennypfeil@Jennys-MacBook-Pro-3 ~ % rmdir /Users/jennypfeil/Documents/HMB301_mod
ules/week_7
```

System setup

Now that you have learnt how to interface with your operating system to perform basic tasks like creating directories, we will next use the CLI to set up our computers to be able to use the Python programming language and all the associated Python packages that we have been using in all the previous modules. Although you didn't need to personally set up Python and all the associated Python packages when we were using JupyterHub to write and run our code, this was actually all done for you on the backend. Now it will be your turn to set it all up from scratch.

An **environment** includes the various system configurations that are required to run a specific program, often referred to as the system setup. The setup is a very general concept; it includes the hardware of your computer (i.e which GPU you use), the operating system (i.e. macOS or Windows), the Python version you have, and the various installed packages you have (e.g numpy, pandas). All of the setup can be done on your computer without the use of an environment, which would be used for any Python program you chose to run on your computer. However, different projects may require different setups and the use of environments allows you to keep the setup for these projects separately on your computer at the same time.

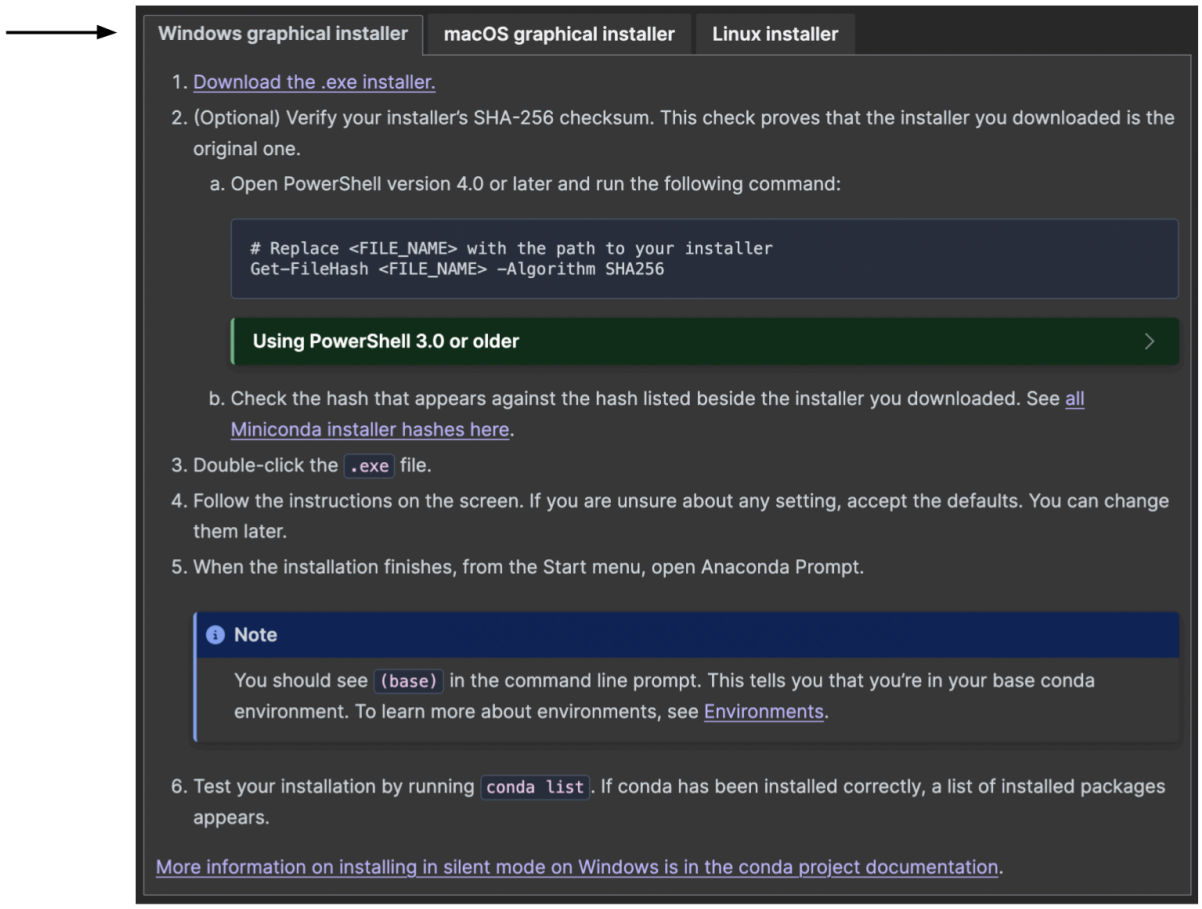
A CLI alone can be used to set up environments on your computer, but it can be complex. Instead, **Conda** is a specific environment and package manager that assists you to create multiple separate environments on your computer. For this exercise, we will be using Conda to create a single environment to install Python and select Python packages.

Follow the steps below on your computer to set up a new python environment using Conda and use the environment.

Step 1: Install Conda

1. Based on your operating system, follow the instructions provided through the link to install Conda on your computer https://docs.anaconda.com/miniconda/miniconda-install/

screenshot of the link and instructions

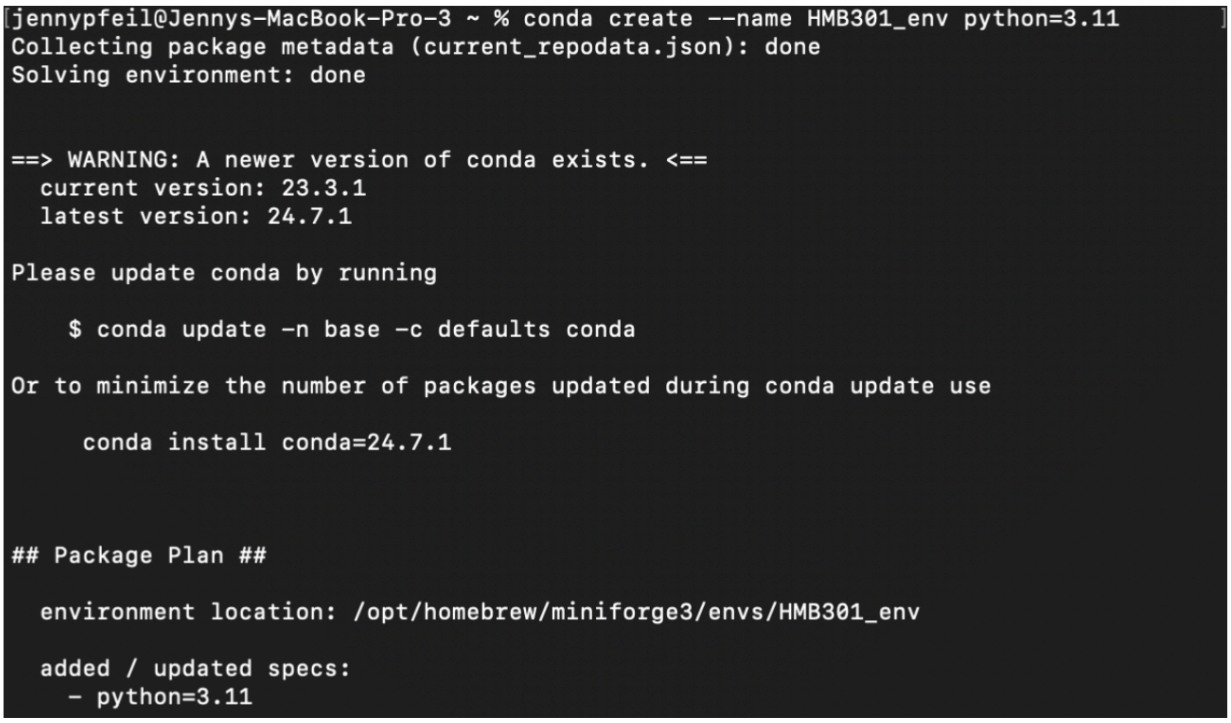


2. Then on your computer, open the *Terminal* (macOS).

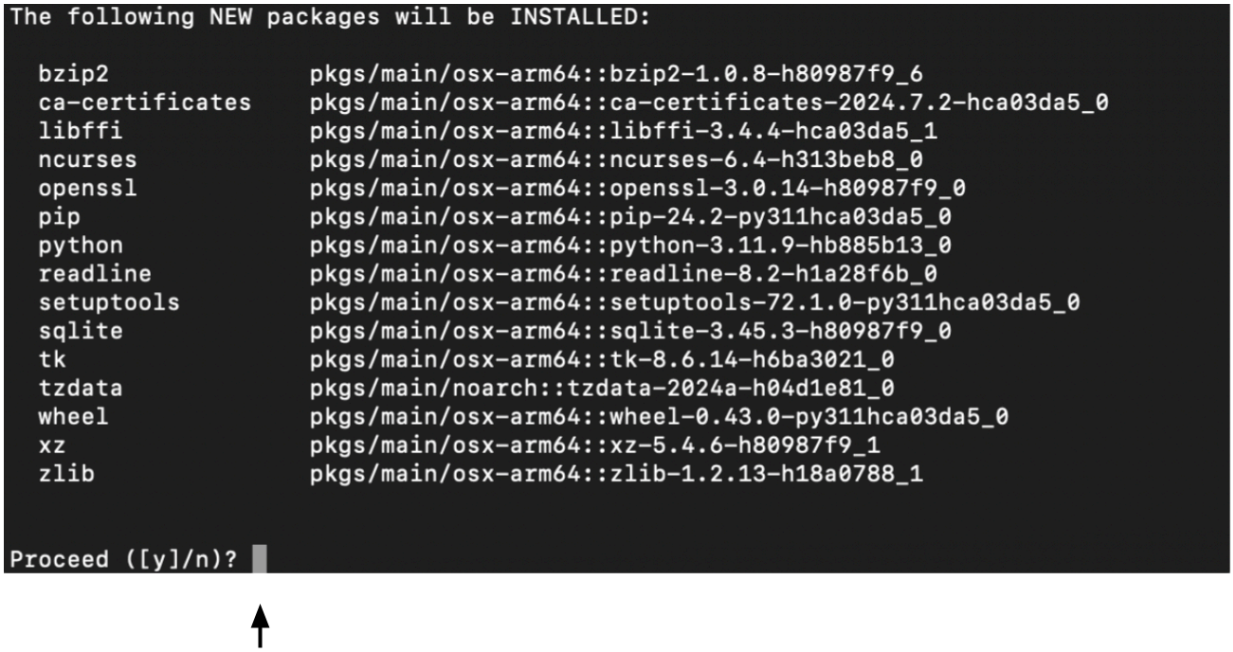
3. Type `conda --help`, then press enter. This is to check if Conda was successfully installed on your computer. You should get something that looks like this if you are a macOS user.

Step 2: Create a Python environment

1. Staying within *Terminal*, type `conda create --name HMB301_env python=3.11`, then press enter. This is a command telling Conda to create a new environment named "HMB301_env" with Python version 3.11 installed. The format of the command for Conda to create a new environment is
`conda create --name <environment name> python=<python version>`
You should get something that looks like this if you are a macOS user.



2. Halfway through the process of creating an environment, you should be prompted on whether you want to install python and its standard packages. Type y, then press enter.



Step 3: Use your new environment

1. To use the environment you just created, you need to activate it. Type `conda activate HMB301_env`, then press enter. The format of the command to activate an environment with Conda is `conda activate <environment name>`. You should now see that your environment name appears in brackets before your computer's username, which means you have successfully activated the environment.

```
jennypfeil@Jennys-MacBook-Pro-3 ~ % conda activate HMB301_env
(HMB301_env) jennypfeil@Jennys-MacBook-Pro-3 ~ %
```

2. To start Python, type `python`, then press enter. You should see the Python prompt (>>>), indicating that your computer is ready to accept Python code.

3. To write code in Python within your environment, type `print('hello word')`, then press enter. You should see the output of the line of code below your original line of code.

You should get something that looks like this if you are a macOS user.

```
(HMB301_env) jennypfeil@Jennys-MacBook-Pro-3 ~ % python
Python 3.11.9 (main, Apr 19 2024, 11:43:47) [Clang 14.0.6 ] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print('hello world')
hello world
>>>
```

4. To exit Python within your environment, type `exit()`, then press enter. You should no longer see the Python prompt (>>>).

5. To leave the environment you just created, you need to deactivate it. Type `conda deactivate`, then press enter. You should no longer see your environment name appearing in brackets before your computer user login.

Step 4: Install packages in your new environment

1. In order to install packages to your environment, you need to first activate your environment. Similar to before, type `conda activate HMB301_env`, then press enter.

2. With the environment activated, you can install packages using Conda, type `conda install numpy`, then press enter. The format of the command to install packages using Conda is `conda install <package name>=<version>`, where you can optionally specify a version of the package if you do not want the latest version.

3. Similar to before, deactivate your environment when you are done installing packages. Type `conda deactivate`, then press enter.

You have now completed the setup of your first environment using Conda.

Now is a good time to skip to the end of the module to complete part 1 of the graded exercise, then come back to this spot to continue the rest of the module to complete part 2 of the graded exercise.

Introduction to IDEs

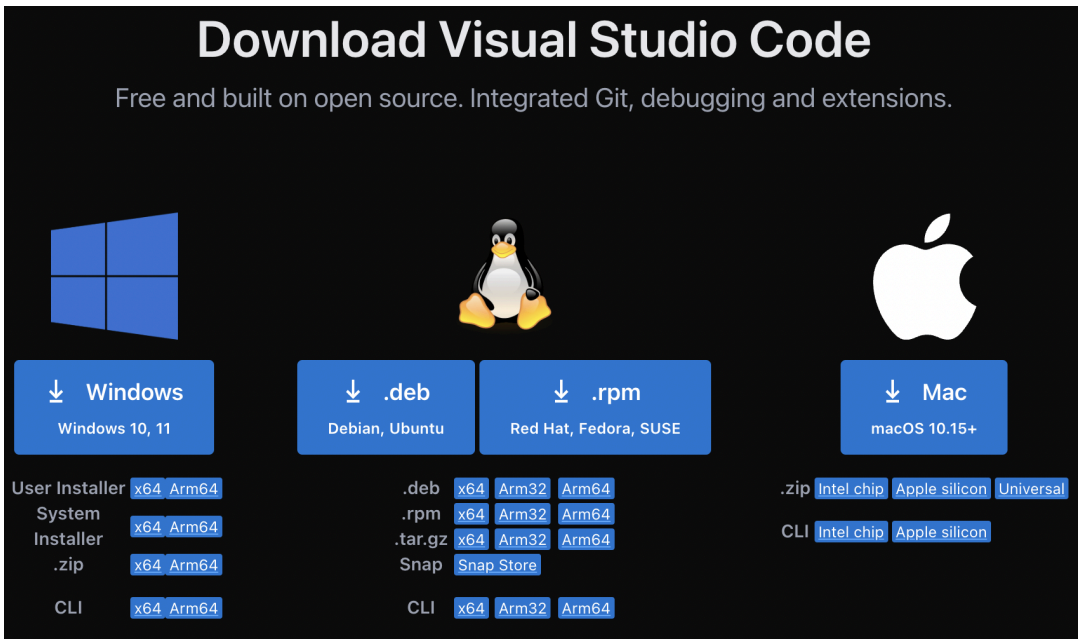
An **integrated development environment (IDE)** is a program you can install on your computer that provides a comprehensive set of tools to help you write more efficiently. IDEs present a unified interface for installing new packages, writing and running code, and provides hints and tools for analysing and debugging your code.

There are multiple IDEs that you can use for programming including Jupyter, PyCharm, and Eclipse, just to list a few. For the context of this module, we will be using **Visual Studio Code (VSCode).** VSCode is an open source software, meaning that anyone can access the original code that is used to create VSCode. This also means that anyone can build onto the VSCode software to add extra features/tools and make coding even more efficient. The choice of which IDE you use will also depend on the ability of that IDE to support the programming language(s) that you plan to use. VScode supports Python which will allow us to write and run code in the Python programming language.

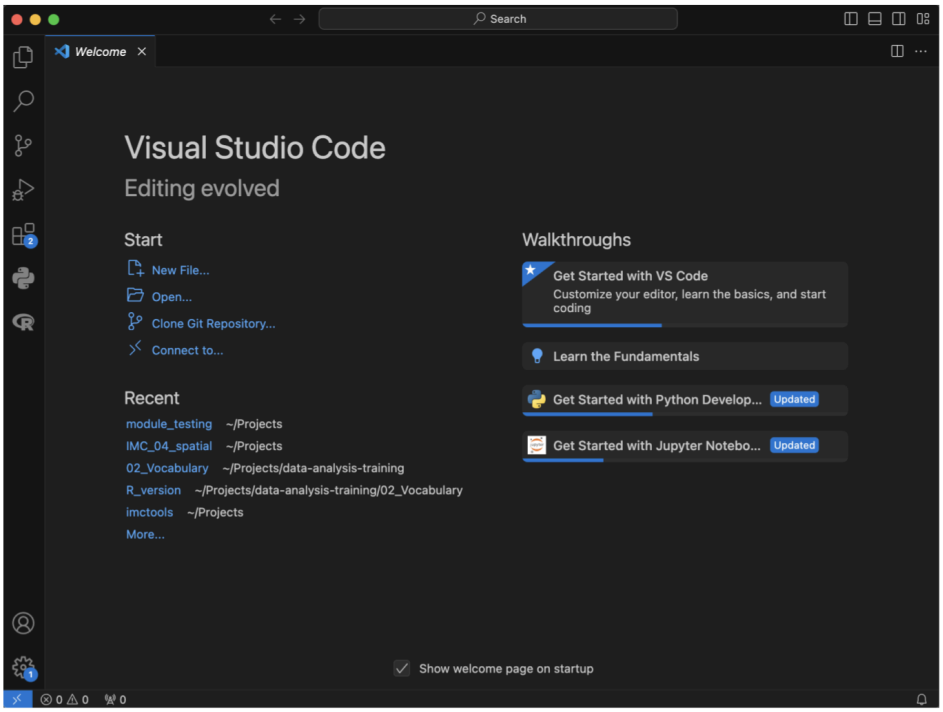
Now that you have your computers set up, you can now begin to install and use VSCode. To do this, follow the steps below.

Step 1: Install and set up VSCode

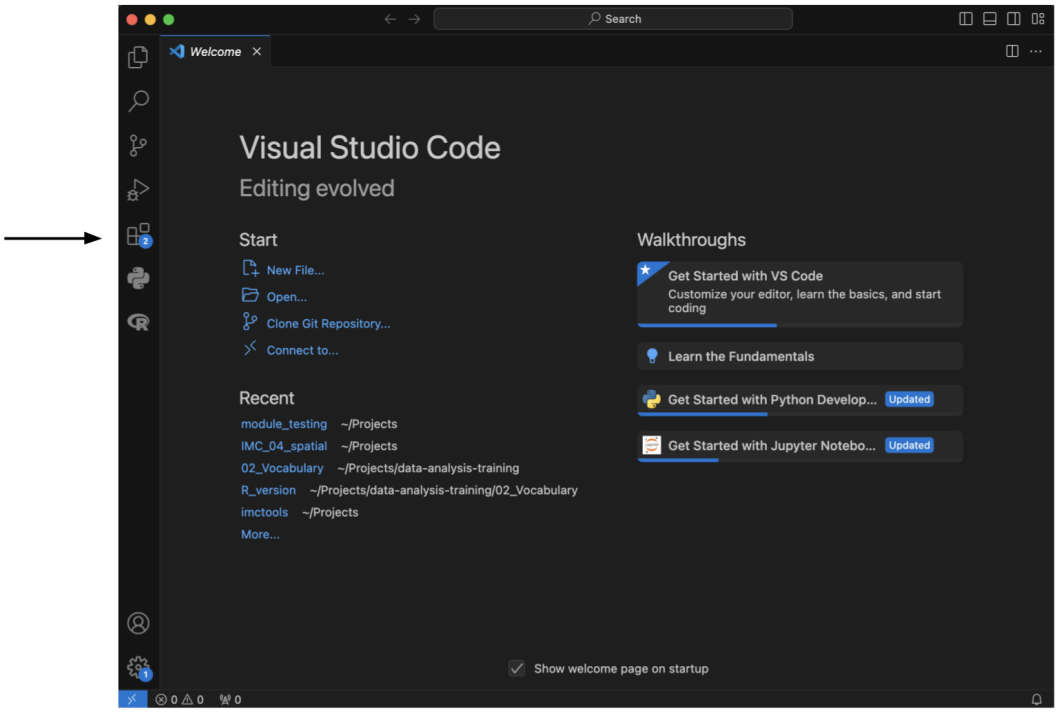
1. Based on your operating system, follow the instructions provided through the link to install VSCode on your computer https://code.visualstudio.com/download. This should follow the normal steps of installing an application on either operating system. A note to Windows users: make sure to add VSCode to the PATH, this should come up as a prompt during installation.



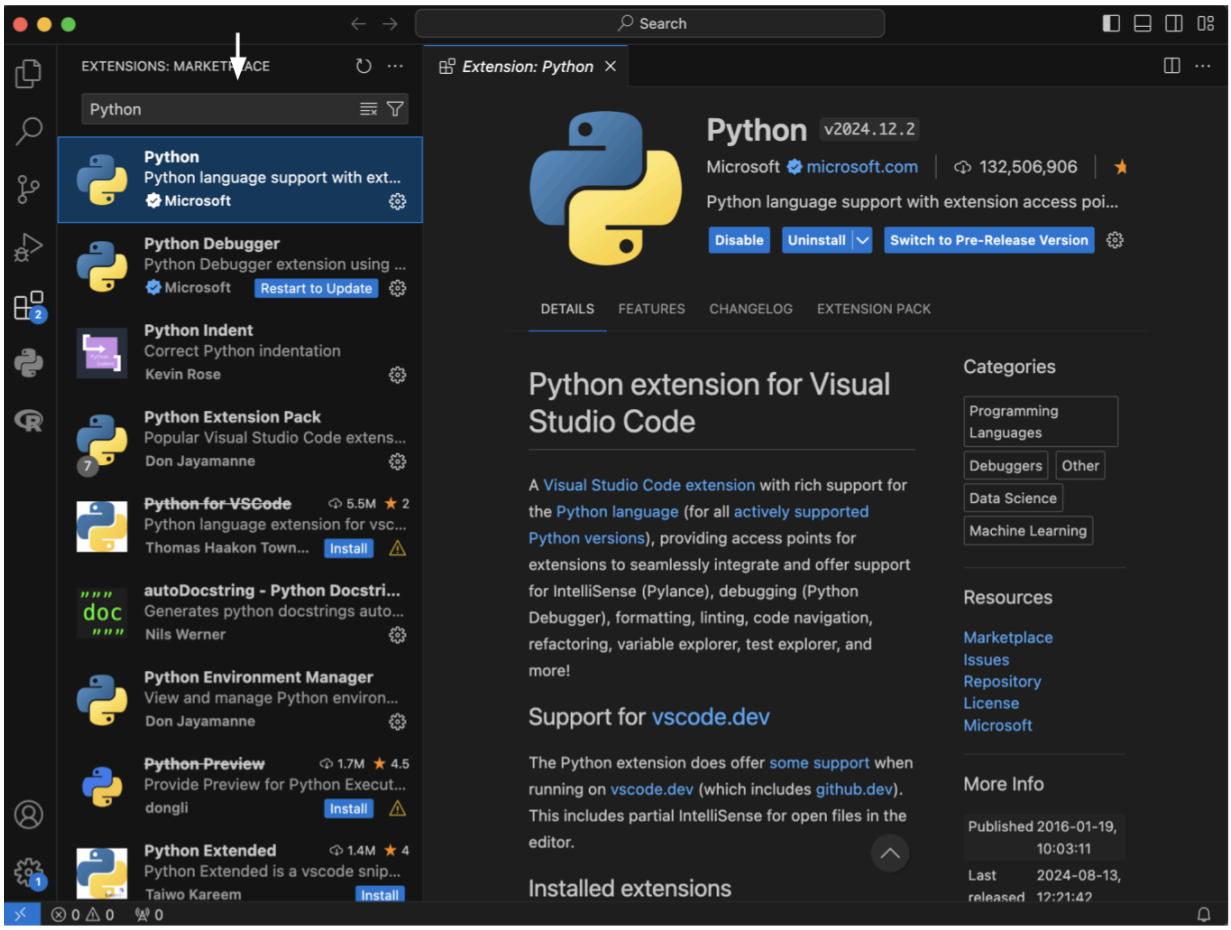
2. Open VSCode. You will be greeted with a Welcome page where you can explore various options and settings.



3. Click on the extensions tab on the left. This will take you to a new tab where you can search all the available extensions in VSCode.



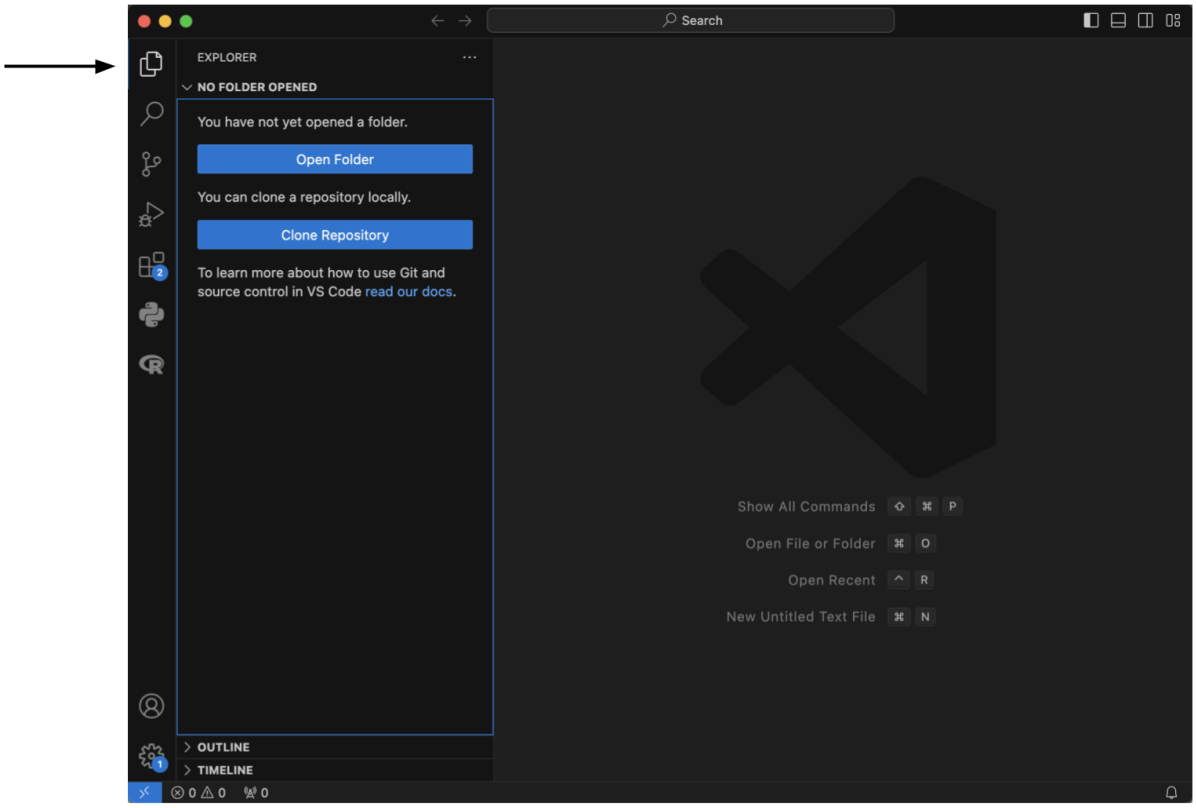
4. Search “Python” in the search bar and the first option should be the Python extension for VS Code, which allows you to seamlessly use Python within VS Code. Install this extension by pressing Install. This extension is used to enhance the development environment by supporting syntax highlighting, auto-completion, quick access to documentation, and more.



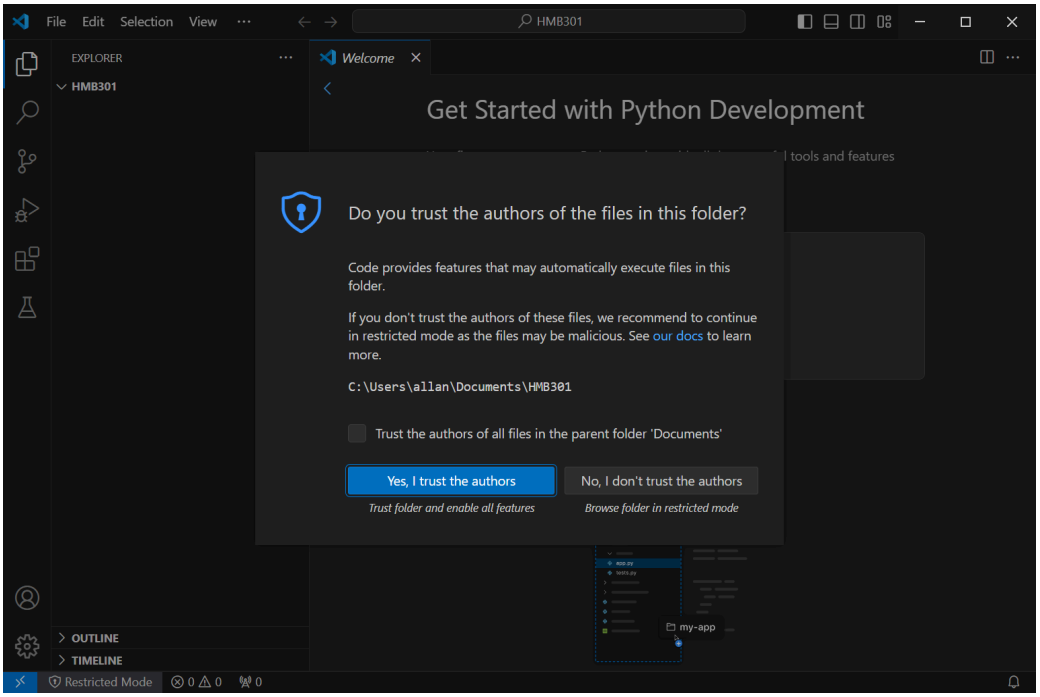
Step 2: Create your first project in VSCode

1. Outside of VSCode, create a new folder on your computer named ‘HMB301’. This will be your project folder. A project folder is simply designed to keep all the related materials for a given project within a single folder, which could include multiple python scripts, data files, output images and more.

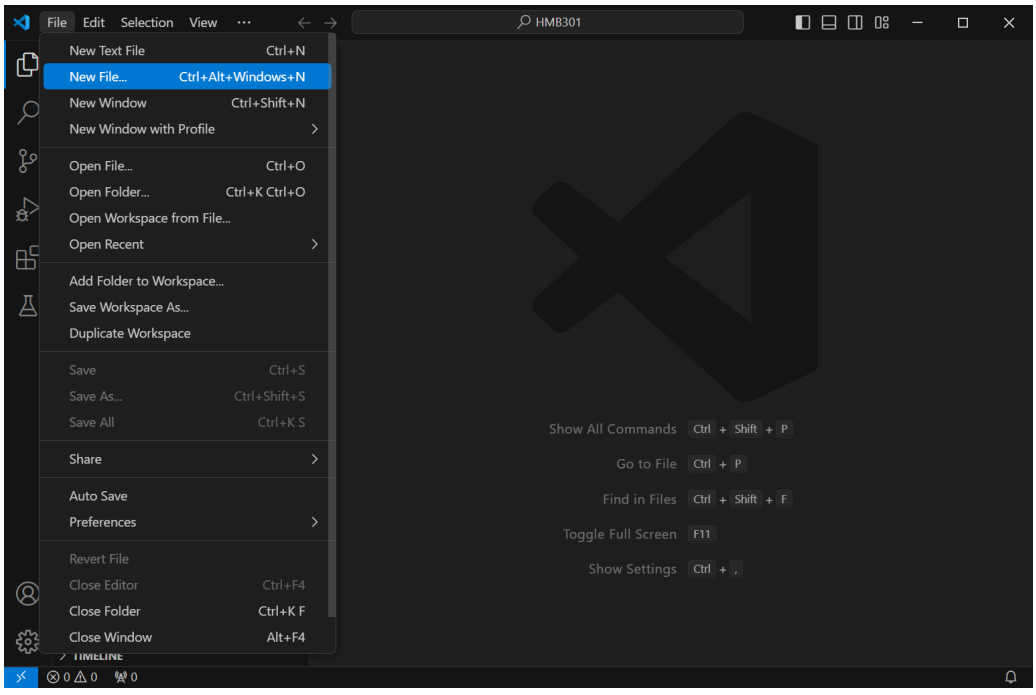
2. Go back to VSCode and click on the explorer tab on the left. This will take you to a new tab where you can search all available folders and files. This should be empty for now because we have not opened any folders or files yet.



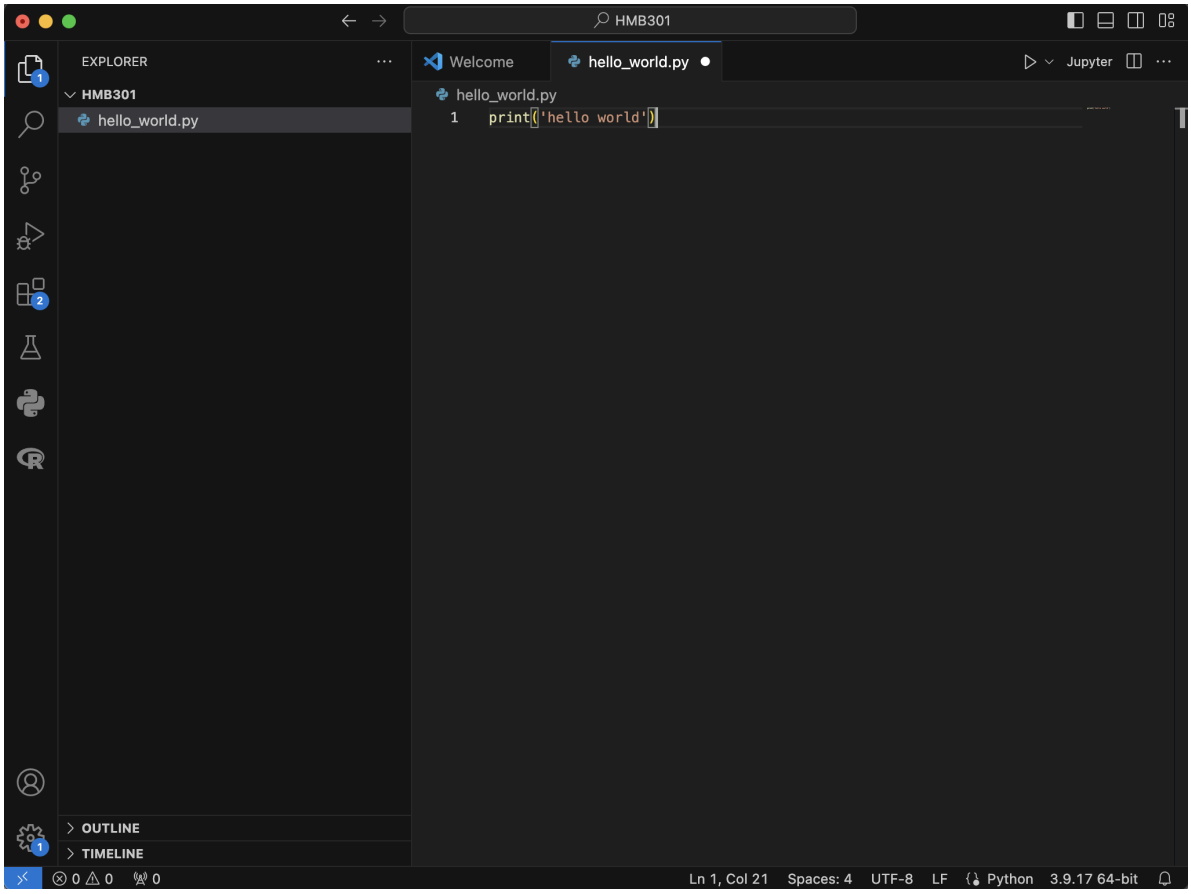
3. Press Open Folder and find the folder named ‘HMB301’ you previously created. You need to first point VSCode to your project folder path where all your files will be stored. If this is your first time opening this folder, you might be asked to trust the files. If so, select “Yes”.



4. Next create a new Python file via the New File... button or the top menu bar as seen below.

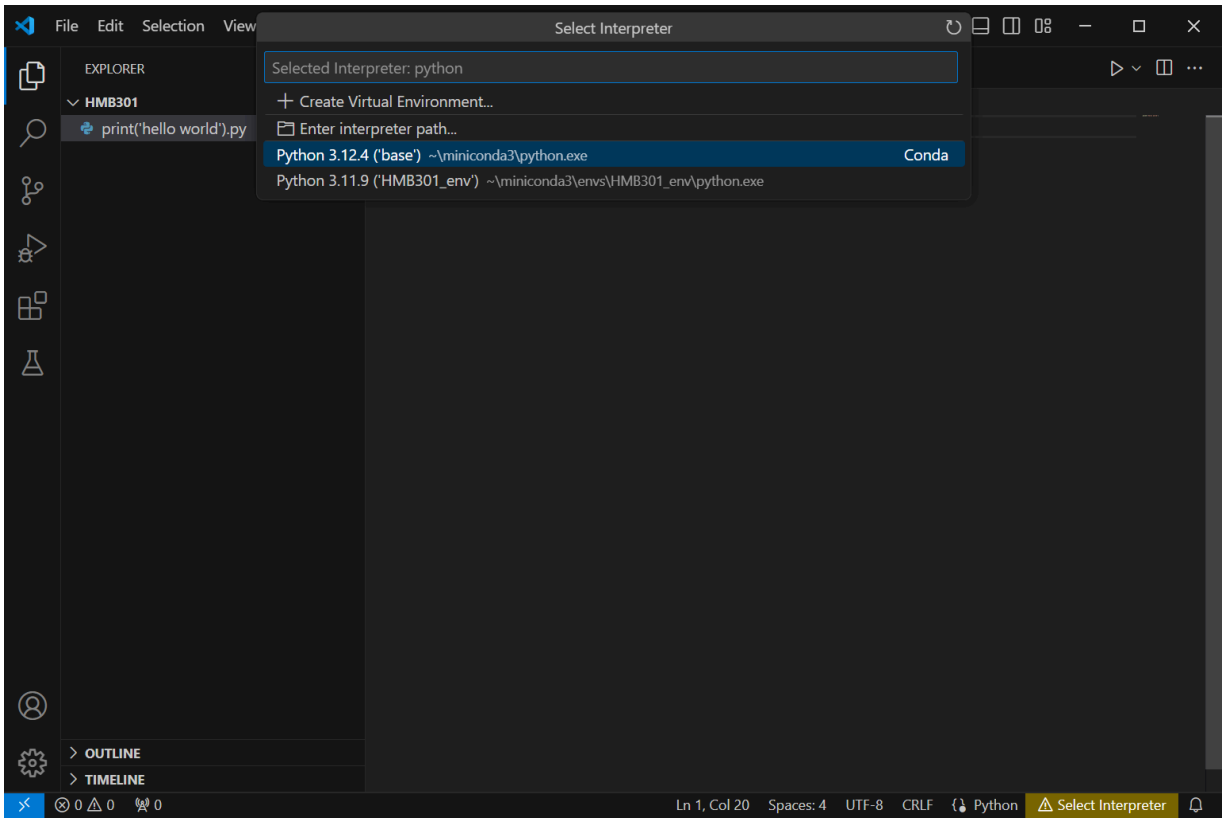


5. Type `print("hello world")` into the new Python file. You should get something that looks like this.



6. Notice that a circle on the right side of the file name  indicates that the change is not saved yet. Save the file like you would with most other applications, either use Ctrl+S on windows, Cmd+S on macOS, or navigate to the File menu to find the Save option.

7. On the bottom left of the window press “Select Interpreter” to select which python VSCode is being used to run the code. This will cause a new window to pop up where you can select the new environment we created named “HMB301_env”.



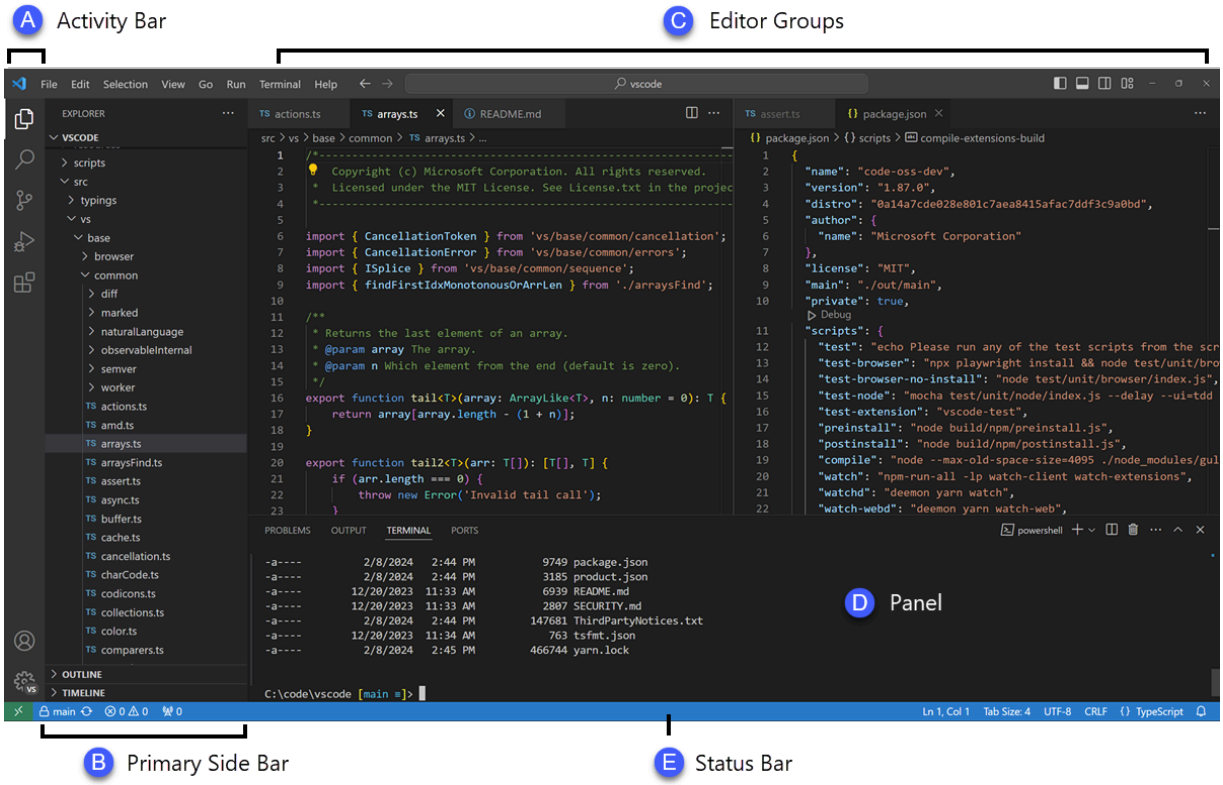
8. Press  to run your Python script. You should see a new panel appear within the VS Code window near the bottom that shows you the output of the code you just ran. You should get something that looks like this.

```
/opt/homebrew/bin/python3.9 /Users/jennypfeil/Projects/HMB301/hello_world.py
jennypfeil@Jennys-MacBook-Pro-3 HMB301 % /opt/homebrew/bin/python3.9 /Users/jennypfeil/
Projects/HMB301/hello_world.py
hello world
jennypfeil@Jennys-MacBook-Pro-3 HMB301 %
```

You have now created your first Python project in VSCode. Here we have only generated a single Python file in the project however once you begin to use VSCode more, there are many other features of VSCode that you can use to make coding more efficient.

Understand the VSCode interface

The main window of VSCode can be broken down into 5 major areas that provide different functions. These major areas are labeled in the image below.



(A) **Activity Bar**: Provides access to the explorer, search, version control, debugging, and extensions tabs. Each of these tabs will allow you to modify different aspects of your project, external to the code itself. The order of the icon will differ per user and can be changed by dragging and moving the icons.

(B) **Primary Side Bar**: Depending on what is selected in the Activity Bar, it can display the contents of your project (explorer), search results (search), source control options (version control), or debug information (debugging), and recommended extensions (extensions).

(C) **Editor Groups**: Displays files that you have open, which can include python scripts, data files, output images and more. You can even open multiple tabs for different files at the same time by displaying them side by side or horizontally. For files containing code, VSCode will automatically highlight keywords and syntax, given the programming language used. VSCode also provides auto-completion and bracket matching for files containing code, which can reduce the likelihood of syntax errors and save your time.

(E) **Status Bar**: Provides information on the files you have opened, such as the format, language, and more. It also provides information about the opened project, such as information on saved versions of your project.

(D) **Panel**: Provide an additional space to view output, errors and warnings, and an integrated terminal. The Panel can also be moved to the left or right for more vertical space.

Conclusion

This wraps up the content for module week 7. We have gone over what an IDE is, how to set up a Python environment, how to set up a VSCode project, and the various features of VSCode. Hopefully with new skills you have learnt from this module, you will be able to explore more coding projects outside of this course and outside of classes.

Graded exercises

Part 1

Install the packages “pandas” and “matplotlib” into your newly created Conda environment named “HMB301_env” through Terminal / Command Prompt.

Part 2

Now that you have learned how to set up your own project in VSCode we want you to create and save a figure within VSCode.

1. Create a new Python file and paste the following code within the file. Note that there are some changes within the code that you need to make and are specified with `#TODO:`. The dataset file should have been downloaded with this pdf and should be saved within the same project folder as the Python file to run in VSCode.

```
import pandas as pd
import matplotlib.pyplot as plt
# TODO: load the dataset as pandas dataframe
# the file name should be "mammal_teeth.csv"
plt.figure(figsize=(5, 10)) # set figure size
plt.scatter(x=df_teeth['Top incisors'],
            y=df_teeth['MAMMAL']) # set figure x, y axis
plt.gca().xaxis.set_visible(False)
# TODO: change the title name to include your name
plt.title("plot for mammal_teeth dataset")
plt.savefig("mammal_teeth_scatterplot.png", dpi=150) # save the figure
```

2. Run the code which should output an image. Upload the output image to quercus for grading.