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

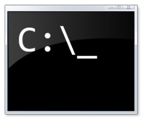
Up to this point, you've been working with our modules using the University of Toronto's JupyterHub web platform, and the simple "Jupyter Classic" user interface for notebooks. While this interface is suitable for beginners, as you perform more advanced computational tasks, you'll begin needing more 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.

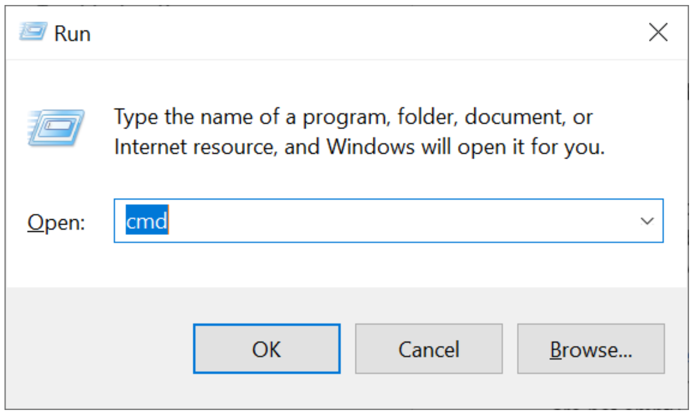
Introduction to CLIs

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.

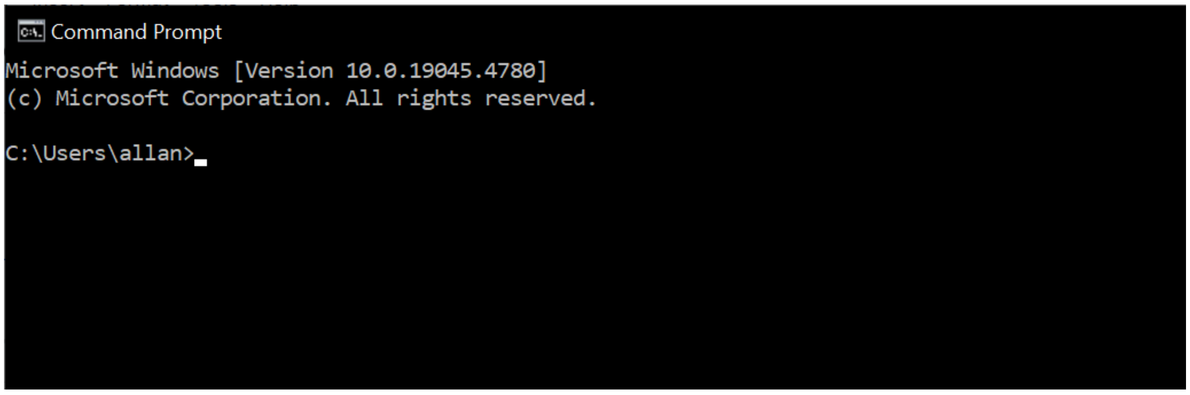
In order to use CLIs, you enter standard commands that the operating system understands. A **command** is a line of text that follows a specific format and instructs a computer to execute a specific function. A single command will often consist of a single string of characters with no spaces, often abbreviating 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 *Command Prompt* on your computer. On your computer, press the Windows key + R on your keyboard, then type CMD and press "enter" to open the Command Prompt application. This will open a new window, where you can type in commands for your computer to execute.




Run
Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.
Open:
cmd
OK
Cancel
Browse...

The new window should look something like this



```
Microsoft Windows [Version 10.0.19045.4780]
(c) Microsoft Corporation. All rights reserved.

C:\Users\allan>
```

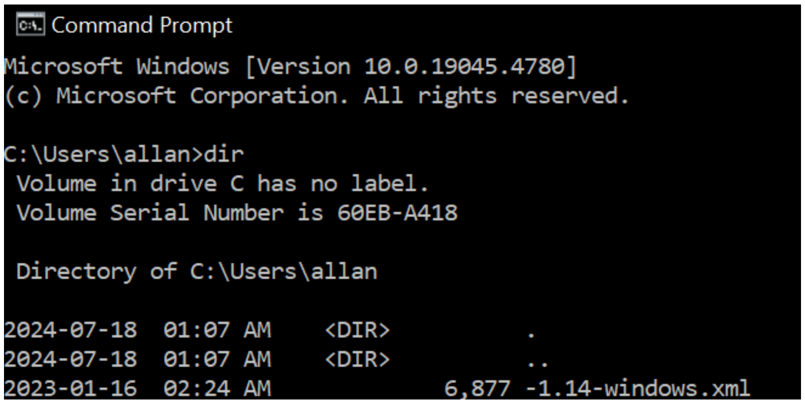
You will see **C:\Users\**, followed by your username, then another \. This will signify that the terminal is ready to take a command. After executing a command and before executing your next command, make sure that you see the \ which will mean that the previous command has finished executing and that 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 the CLI is the start of performing more complex tasks with the CLI.

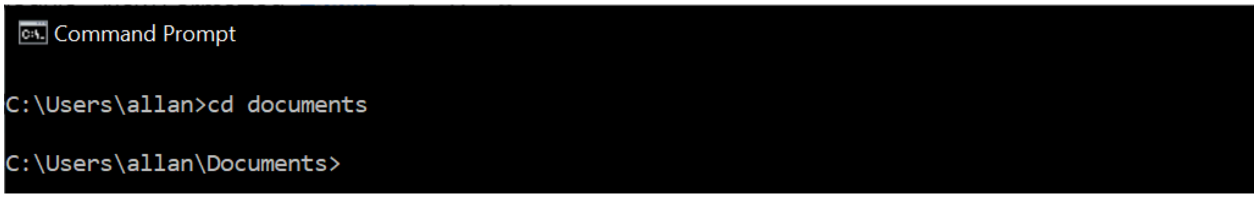
Follow the steps below to navigate your computer through the CLI.

1. Type `dir`, then press enter. This command will allow you to list all contents within the current directory. Every time 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 folders in your user account's home folder, like Downloads, Documents and more.



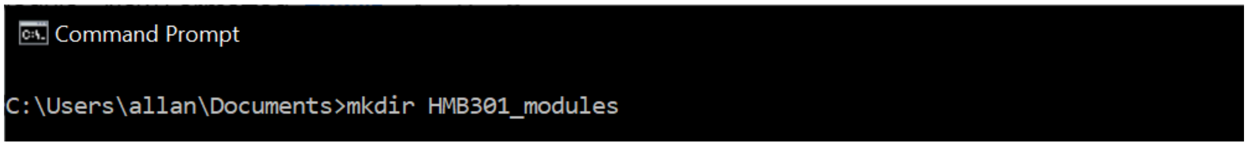
```
Command Prompt
Microsoft Windows [Version 10.0.19045.4780]
(c) Microsoft Corporation. All rights reserved.

C:\Users\allan>dir
 Volume in drive C has no label.
 Volume Serial Number is 60EB-A418

 Directory of C:\Users\allan

2024-07-18  01:07 AM    <DIR>          .
2024-07-18  01:07 AM    <DIR>          ..
2023-01-16  02:24 AM             6,877 -1.14-windows.xml
```

2. Type `cd Documents`, then press enter. The `cd` command allows you to enter into a specific directory, if it already exists. 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 as seen below.



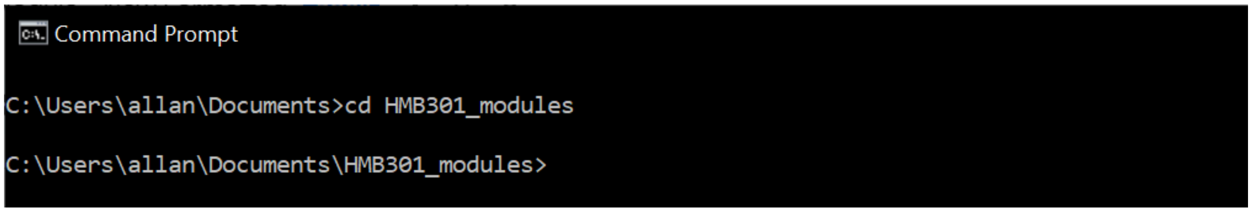
```
Command Prompt

C:\Users\allan>cd documents

C:\Users\allan\Documents>
```

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



```
Command Prompt

C:\Users\allan\Documents>mkdir HMB301_modules
```

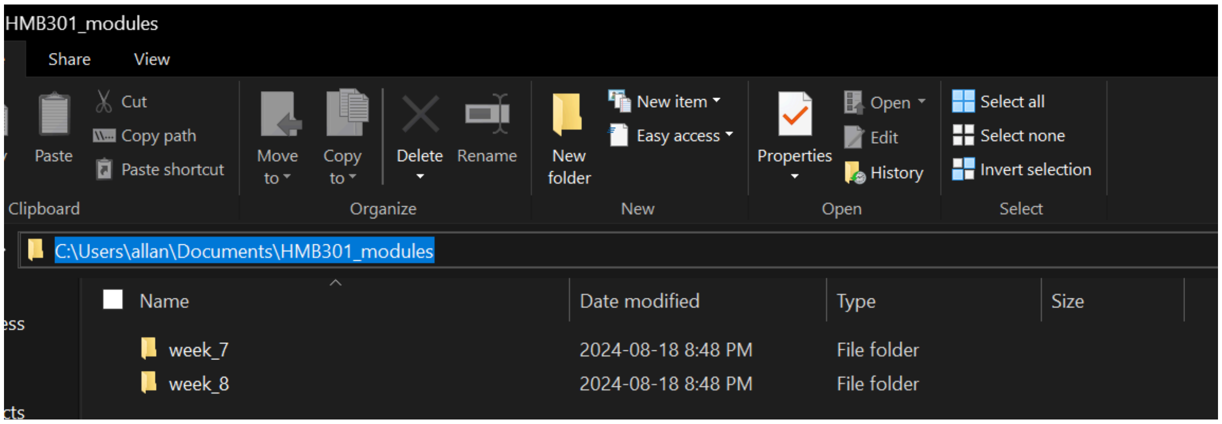
4. Using the command we learned in Step 3, enter the directory we just created. Type `cd HMB301_modules`, then press enter. This will allow you to enter the directory you just created. We can now see from the command line that we are within our new directory.



```
Command Prompt

C:\Users\allan\Documents>cd HMB301_modules

C:\Users\allan\Documents\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.

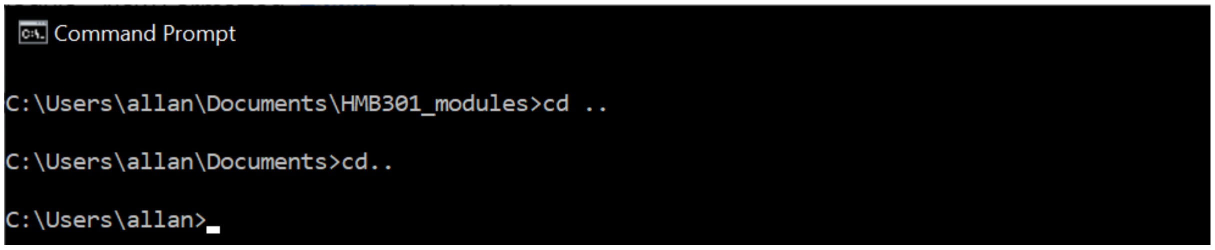
You can check these have been correctly added by entering C:\Users\<username>\Documents\HMB301_modules into the address bar of the *Folders* application. Here we can see the two sub directories we created.



C:\Users\allan\Documents\HMB301_modules

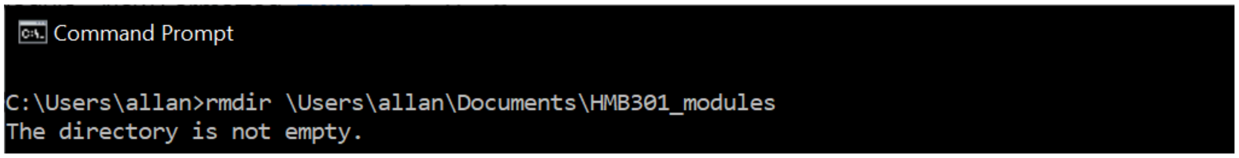
Name	Date modified	Type	Size
week_7	2024-08-18 8:48 PM	File folder	
week_8	2024-08-18 8:48 PM	File folder	

6. Using the `cd ..` command twice, navigate back to our main directory where we started in Step 2. You should now see that nothing appears after your username, which means you have successfully made it back to your home directory.



```
Command Prompt

C:\Users\allan\Documents\HMB301_modules>cd ..

C:\Users\allan\Documents>cd..

C:\Users\allan>_
```

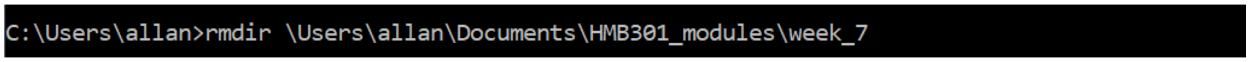
7. Now as an exercise we will try to remove some of the directories we just created. Type `rmdir \Users\<username>\Documents\HMB301_modules`, then press enter. This command works in the format `rmdir <directory name>` . However, this command should not work and you should get an error telling you that you cannot remove directories that are not empty.



```
Command Prompt

C:\Users\allan>rmdir \Users\allan\Documents\HMB301_modules
The directory is not empty.
```

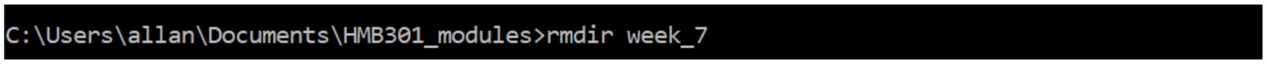
This is because you can only remove a directory when there is no content in it. If you want to remove the whole directory **including all its contents**, you will need to use the command in the format `rmdir -r <directory name>` where the -r means the computer will remove all files and subdirectories inside the parent directory.

8. Now knowing this, let's try to successfully remove a directory. Type `rmdir \Users\<username>\Documents\HMB301_modules\week_7` , then press enter. This should now work.



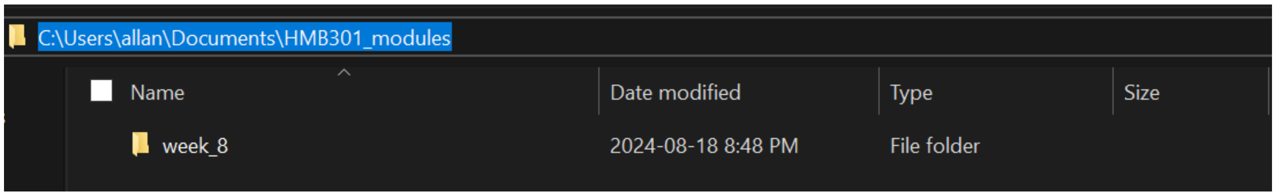
```
C:\Users\allan>rmdir \Users\allan\Documents\HMB301_modules\week_7
```

Above we navigated to the root folder of the directory we wanted to remove in our command line, however this could have also been done using the `cd` command to get back to the HMB301_modules directory, then `rmdir week_7` .



```
C:\Users\allan\Documents\HMB301_modules>rmdir week_7
```

Once again we can check we have successfully deleted the directory by entering C:\Users\<username>\Documents\HMB301_modules into the address bar of the folders application. Here we can see the week_7 folder is now gone.



C:\Users\allan\Documents\HMB301_modules

Name	Date modified	Type	Size
week_8	2024-08-18 8:48 PM	File folder	

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.

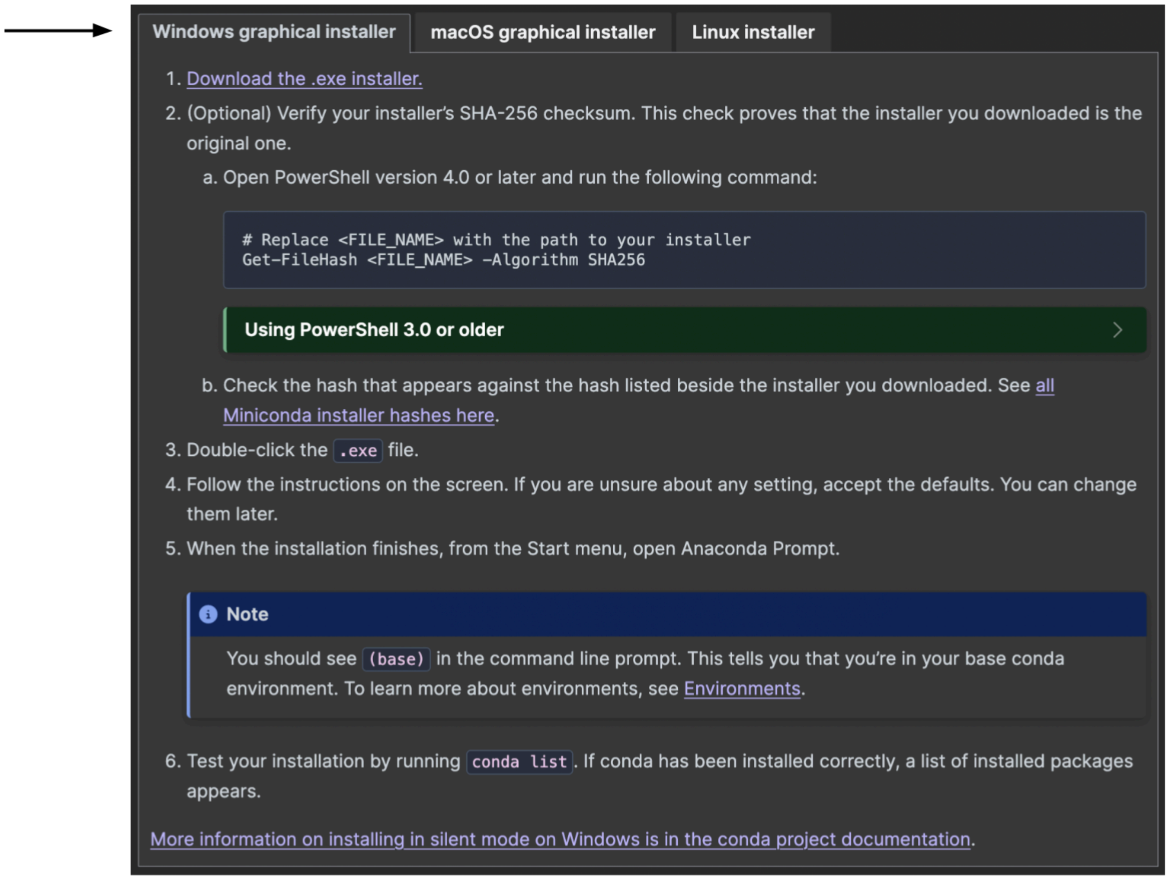
An **environment** includes the various system configurations that are required to run a specific program, often referred to as the system setup. The setup is a very general concept; it includes the hardware of your computer (i.e which GPU you use), the operating system (i.e. macOS or Windows), the Python 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. For a Windows based operating system, follow the instructions provided through the link to install Conda on your computer https://docs.anaconda.com/miniconda/miniconda-install/



1. Download the .exe installer.
2. (Optional) Verify your installer's SHA-256 checksum. This check proves that the installer you downloaded is the original one.
   a. Open PowerShell version 4.0 or later and run the following command:

   ```
   # Replace <FILE_NAME> with the path to your installer
   Get-FileHash <FILE_NAME> -Algorithm SHA256
   ```

   Using PowerShell 3.0 or older

   b. Check the hash that appears against the hash listed beside the installer you downloaded. See all Miniconda installer hashes here.
3. Double-click the `.exe` file.
4. Follow the instructions on the screen. If you are unsure about any setting, accept the defaults. You can change them later.
5. When the installation finishes, from the Start menu, open Anaconda Prompt.

Note

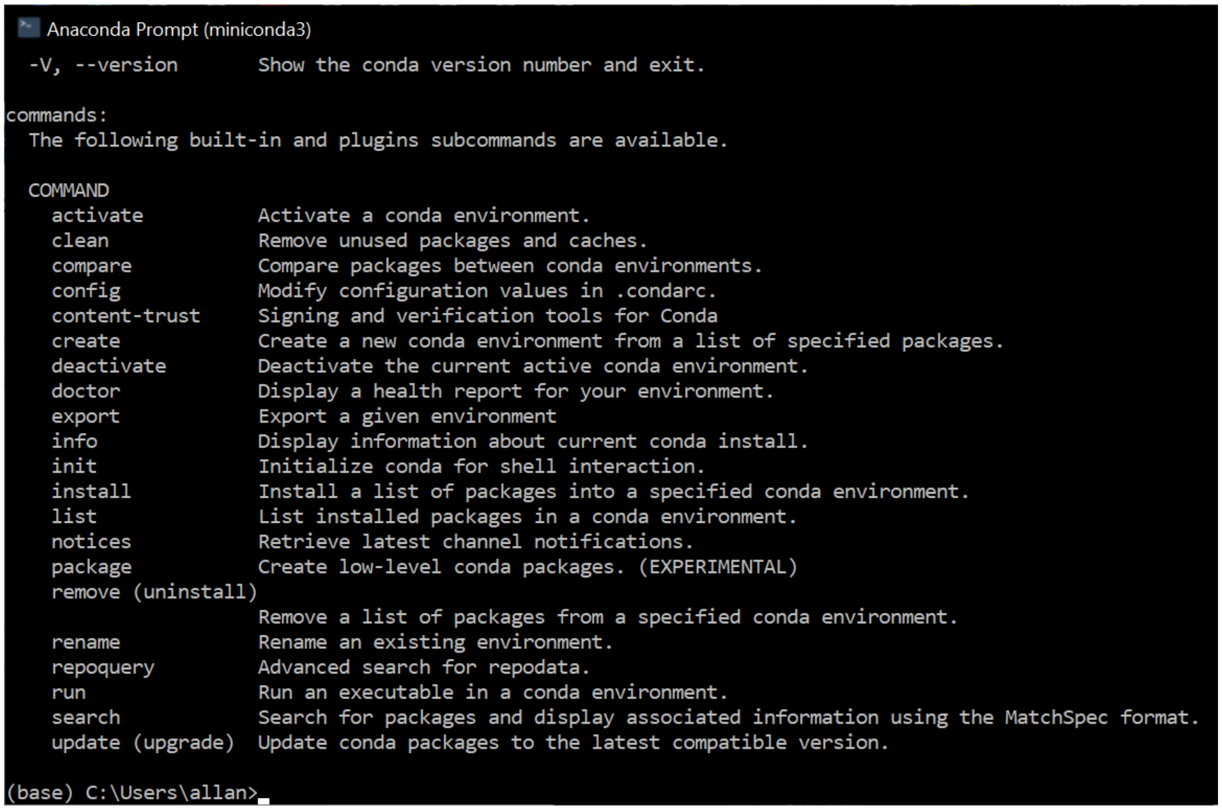
You should see `(base)` in the command line prompt. This tells you that you're in your base conda environment. To learn more about environments, see Environments.

6. Test your installation by running `conda list`. If conda has been installed correctly, a list of installed packages appears.

More information on installing in silent mode on Windows is in the conda project documentation.

2. After you have successfully installed Conda on your computer, open the *Anaconda Prompt* application. This will open a new window, called a "terminal", where you can type in commands for your computer to execute.

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 windows user.

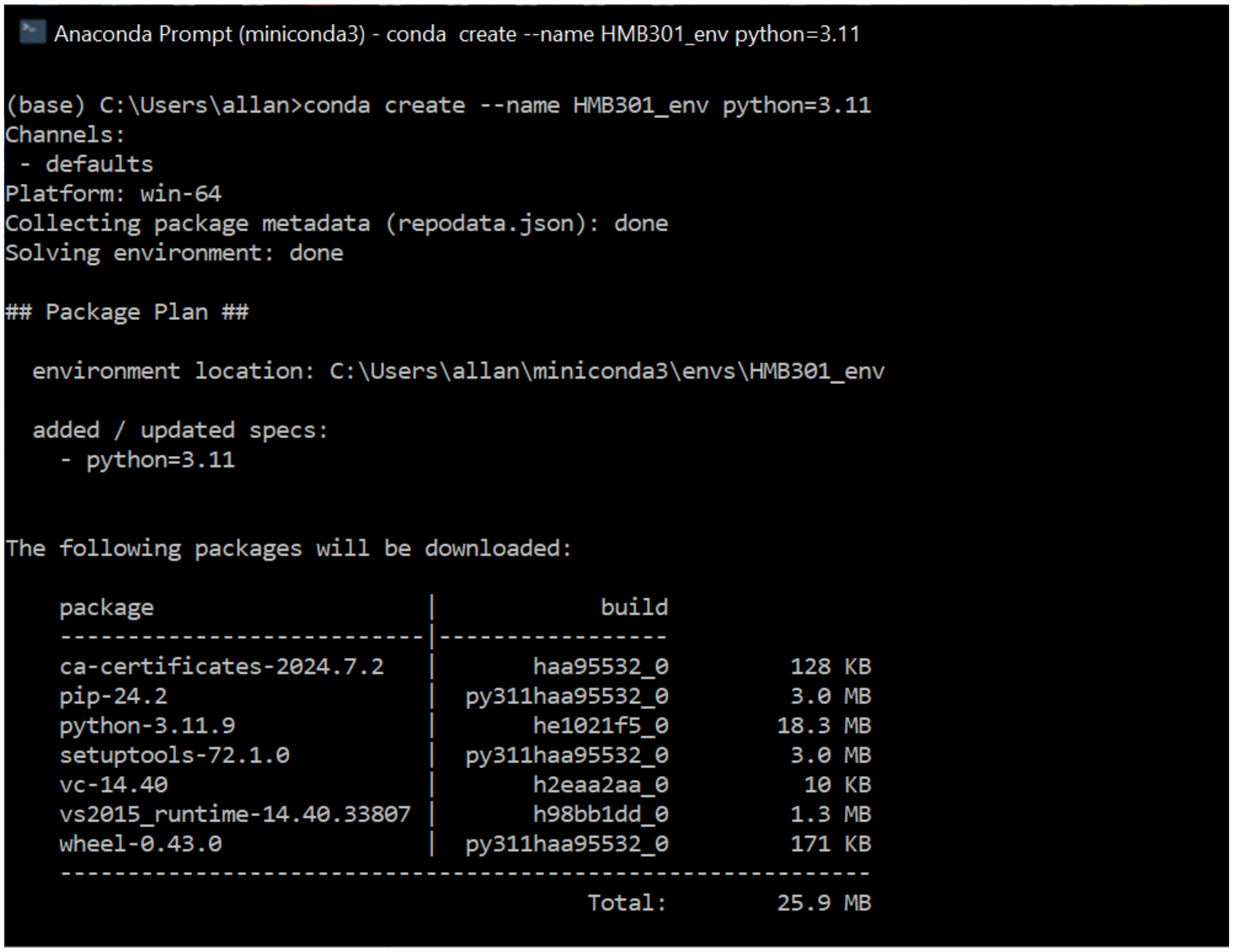


Anaconda Prompt (miniconda3)

```
  -V, --version       Show the conda version number and exit.

commands:
  The following built-in and plugins subcommands are available.

  COMMAND
    activate          Activate a conda environment.
    clean             Remove unused packages and caches.
    compare           Compare packages between conda environments.
    config            Modify configuration values in .condarc.
    content-trust     Signing and verification tools for Conda
    create            Create a new conda environment from a list of specified packages.
    deactivate        Deactivate the current active conda environment.
    doctor            Display a health report for your environment.
    export            Export a given environment
    info              Display information about current conda install.
    init              Initialize conda for shell interaction.
    install           Install a list of packages into a specified conda environment.
    list              List installed packages in a conda environment.
    notices           Retrieve latest channel notifications.
    package           Create low-level conda packages. (EXPERIMENTAL)
    remove (uninstall)
                      Remove a list of packages from a specified conda environment.
    rename            Rename an existing environment.
    repoquery         Advanced search for repodata.
    run               Run an executable in a conda environment.
    search            Search for packages and display associated information using the MatchSpec format.
    update (upgrade)  Update conda packages to the latest compatible version.

(base) C:\Users\allan>
```

Step 2: Create a Python environment

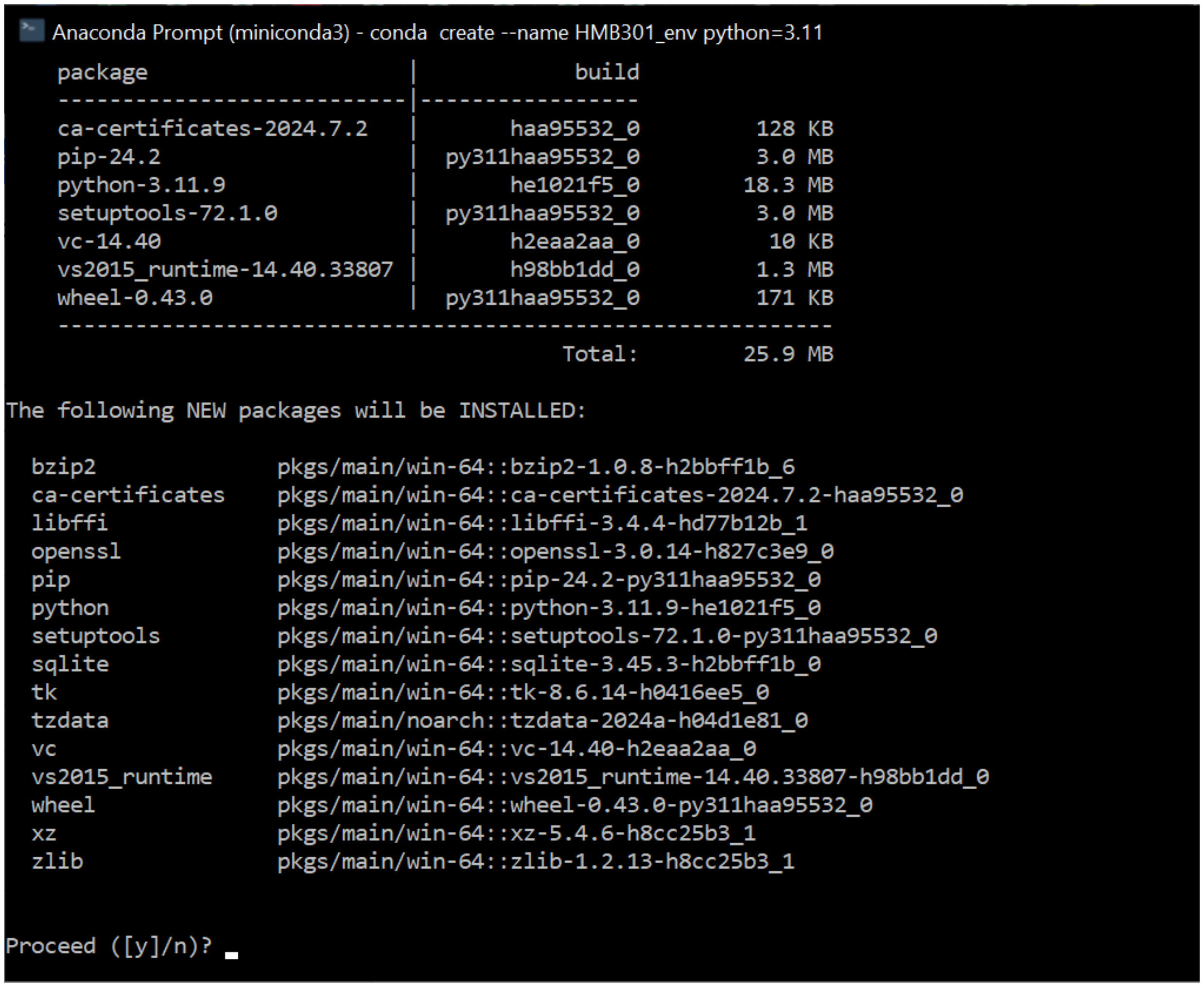
1. Staying within *Anaconda Prompt*, type `conda create --name HMB301_env python=3.11`, then press enter. This is a command telling Conda to create a new environment named “HMB301_env” with Python version 3.11 installed. The format of the command for Conda to create a new environment is
`conda create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



Anaconda Prompt (miniconda3) - conda create --name HMB301_env python=3.11

```
(base) C:\Users\allan>conda create --name HMB301_env python=3.11
Channels:
 - defaults
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\Users\allan\miniconda3\envs\HMB301_env

  added / updated specs:
    - python=3.11


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    ca-certificates-2024.7.2   |       haa95532_0         128 KB
    pip-24.2                   |  py311haa95532_0         3.0 MB
    python-3.11.9              |       he1021f5_0        18.3 MB
    setuptools-72.1.0          |  py311haa95532_0         3.0 MB
    vc-14.40                   |       h2eaa2aa_0          10 KB
    vs2015_runtime-14.40.33807 |       h98bb1dd_0         1.3 MB
    wheel-0.43.0               |  py311haa95532_0         171 KB
    ------------------------------------------------------------
                                           Total:        25.9 MB
```

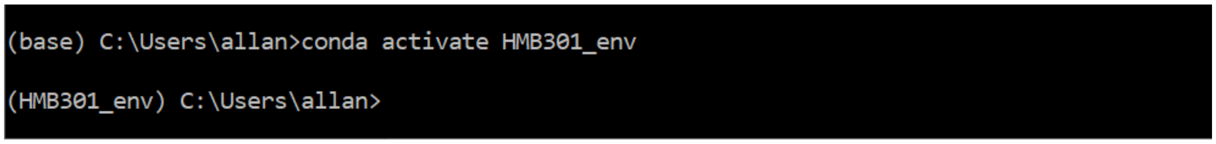
2. Halfway through the process of creating an environment, you should be prompted on whether you want to install python and its standard packages. Type y, then press enter.



```
Anaconda Prompt (miniconda3) - conda  create --name HMB301_env python=3.11
    package                    |            build
    ---------------------------|-----------------
    ca-certificates-2024.7.2   |       haa95532_0         128 KB
    pip-24.2                   |  py311haa95532_0         3.0 MB
    python-3.11.9              |       he1021f5_0        18.3 MB
    setuptools-72.1.0          |  py311haa95532_0         3.0 MB
    vc-14.40                   |       h2eaa2aa_0          10 KB
    vs2015_runtime-14.40.33807 |       h98bb1dd_0         1.3 MB
    wheel-0.43.0               |  py311haa95532_0         171 KB
    ------------------------------------------------------------
                                           Total:        25.9 MB

The following NEW packages will be INSTALLED:

  bzip2              pkgs/main/win-64::bzip2-1.0.8-h2bbff1b_6
  ca-certificates    pkgs/main/win-64::ca-certificates-2024.7.2-haa95532_0
  libffi             pkgs/main/win-64::libffi-3.4.4-hd77b12b_1
  openssl            pkgs/main/win-64::openssl-3.0.14-h827c3e9_0
  pip                pkgs/main/win-64::pip-24.2-py311haa95532_0
  python             pkgs/main/win-64::python-3.11.9-he1021f5_0
  setuptools         pkgs/main/win-64::setuptools-72.1.0-py311haa95532_0
  sqlite             pkgs/main/win-64::sqlite-3.45.3-h2bbff1b_0
  tk                 pkgs/main/win-64::tk-8.6.14-h0416ee5_0
  tzdata             pkgs/main/noarch::tzdata-2024a-h04d1e81_0
  vc                 pkgs/main/win-64::vc-14.40-h2eaa2aa_0
  vs2015_runtime     pkgs/main/win-64::vs2015_runtime-14.40.33807-h98bb1dd_0
  wheel              pkgs/main/win-64::wheel-0.43.0-py311haa95532_0
  xz                 pkgs/main/win-64::xz-5.4.6-h8cc25b3_1
  zlib               pkgs/main/win-64::zlib-1.2.13-h8cc25b3_1


Proceed ([y]/n)?
```

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.

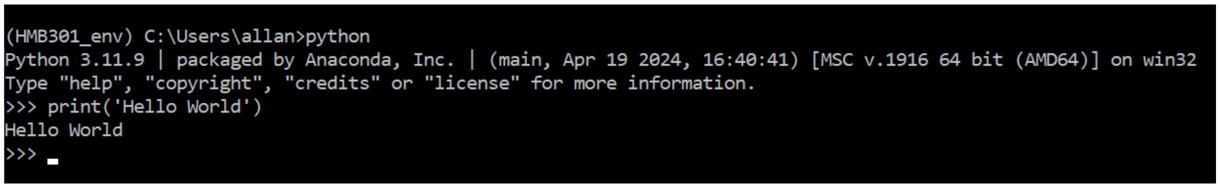


```
(base) C:\Users\allan>conda activate HMB301_env

(HMB301_env) C:\Users\allan>
```

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.

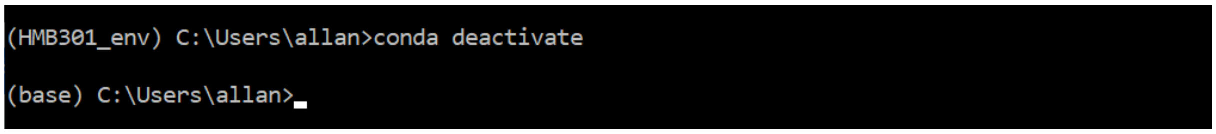
You should get something that looks like this if you are a windows user.



```
(HMB301_env) C:\Users\allan>python
Python 3.11.9 | packaged by Anaconda, Inc. | (main, Apr 19 2024, 16:40:41) [MSC v.1916 64 bit (AMD64)] on win32
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.

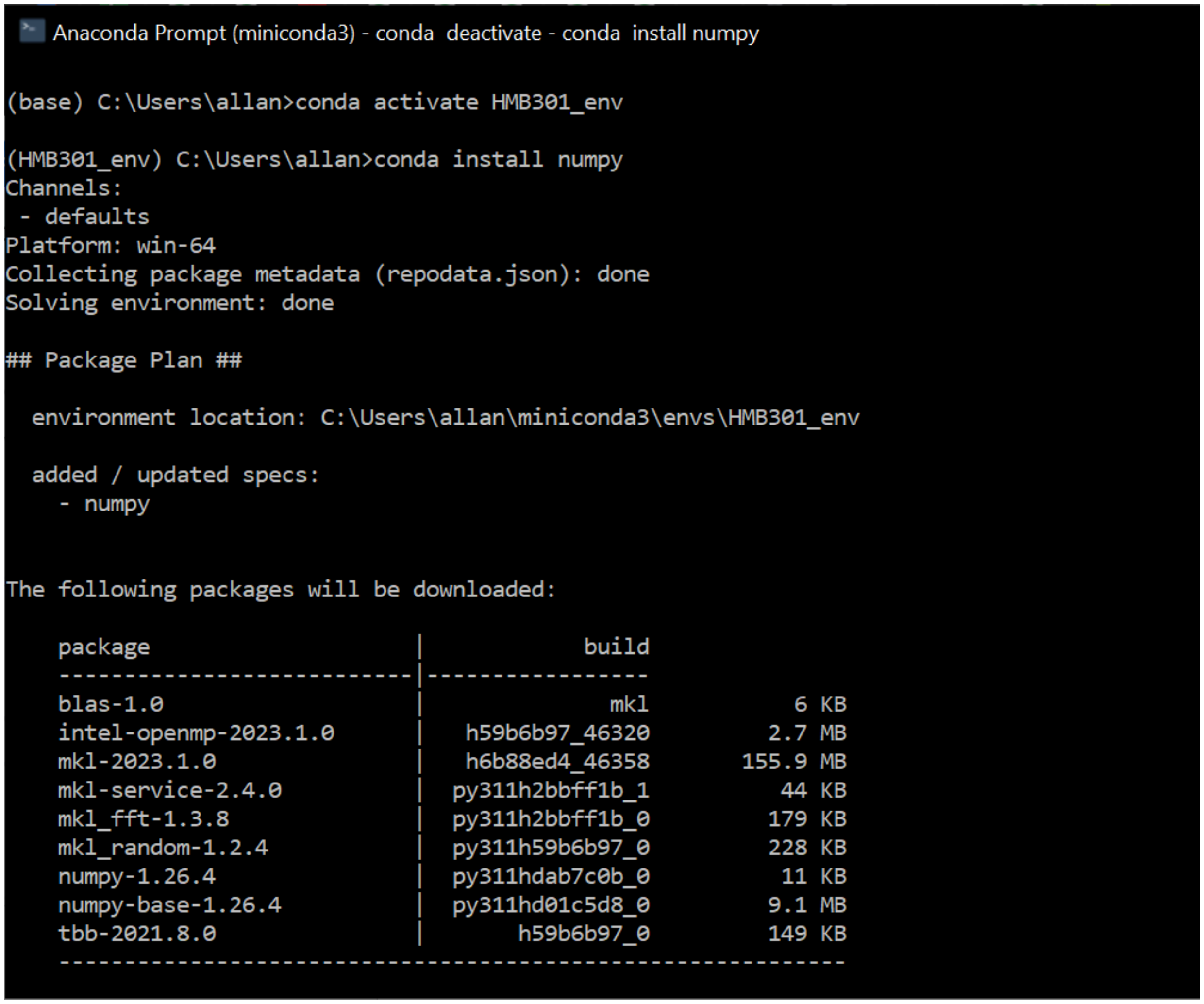


```
(HMB301_env) C:\Users\allan>conda deactivate

(base) C:\Users\allan>
```

Step 4: Install packages in your new environment

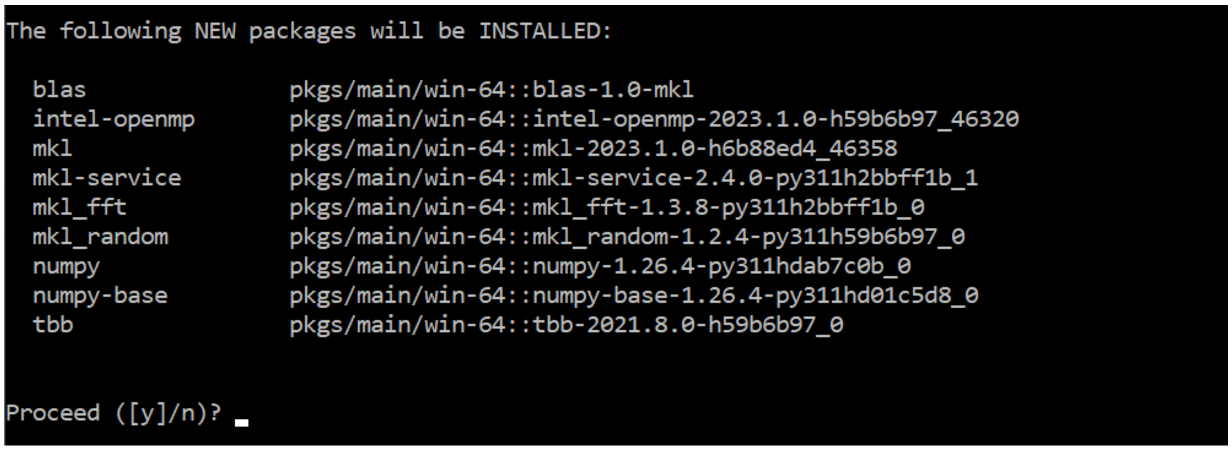
1. In order to install packages to your environment, you need to first activate your environment. Similar to before, type `conda activate HMB301_env`, then press enter.

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.



```
Anaconda Prompt (miniconda3) - conda  deactivate - conda  install numpy

(base) C:\Users\allan>conda activate HMB301_env

(HMB301_env) C:\Users\allan>conda install numpy
Channels:
 - defaults
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\Users\allan\miniconda3\envs\HMB301_env

  added / updated specs:
    - numpy


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    blas-1.0                   |              mkl           6 KB
    intel-openmp-2023.1.0      |   h59b6b97_46320         2.7 MB
    mkl-2023.1.0               |   h6b88ed4_46358       155.9 MB
    mkl-service-2.4.0          |  py311h2bbff1b_1          44 KB
    mkl_fft-1.3.8              |  py311h2bbff1b_0         179 KB
    mkl_random-1.2.4           |  py311h59b6b97_0         228 KB
    numpy-1.26.4               |  py311hdab7c0b_0          11 KB
    numpy-base-1.26.4          |  py311hd01c5d8_0         9.1 MB
    tbb-2021.8.0               |       h59b6b97_0         149 KB
    ------------------------------------------------------------
```

3. Halfway through the process, you should be prompted on whether you want to proceed with the installation. Type `y`, then press enter.



```
The following NEW packages will be INSTALLED:

  blas               pkgs/main/win-64::blas-1.0-mkl
  intel-openmp       pkgs/main/win-64::intel-openmp-2023.1.0-h59b6b97_46320
  mkl                pkgs/main/win-64::mkl-2023.1.0-h6b88ed4_46358
  mkl-service        pkgs/main/win-64::mkl-service-2.4.0-py311h2bbff1b_1
  mkl_fft            pkgs/main/win-64::mkl_fft-1.3.8-py311h2bbff1b_0
  mkl_random         pkgs/main/win-64::mkl_random-1.2.4-py311h59b6b97_0
  numpy              pkgs/main/win-64::numpy-1.26.4-py311hdab7c0b_0
  numpy-base         pkgs/main/win-64::numpy-base-1.26.4-py311hd01c5d8_0
  tbb                pkgs/main/win-64::tbb-2021.8.0-h59b6b97_0


Proceed ([y]/n)?
```

4. 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

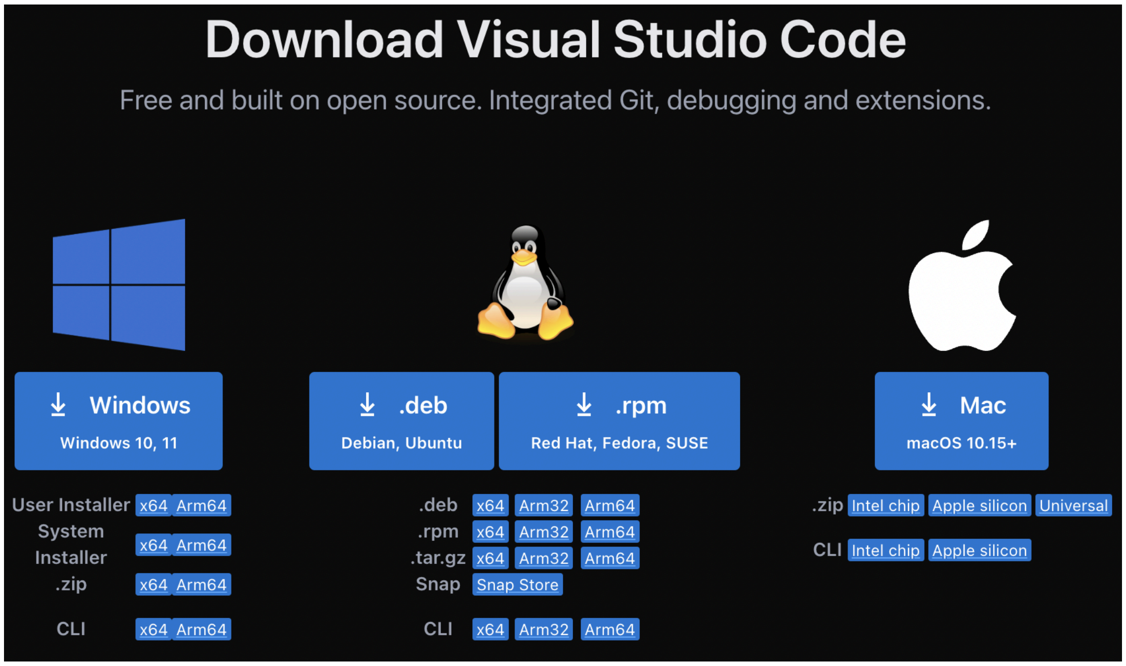
An **integrated development environment (IDE)** is a program you can install on your computer that provides a comprehensive set of tools to help you write more efficiently. IDEs present a unified interface for installing new packages, writing and running code, and provides hints and tools for analysing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



Download Visual Studio Code

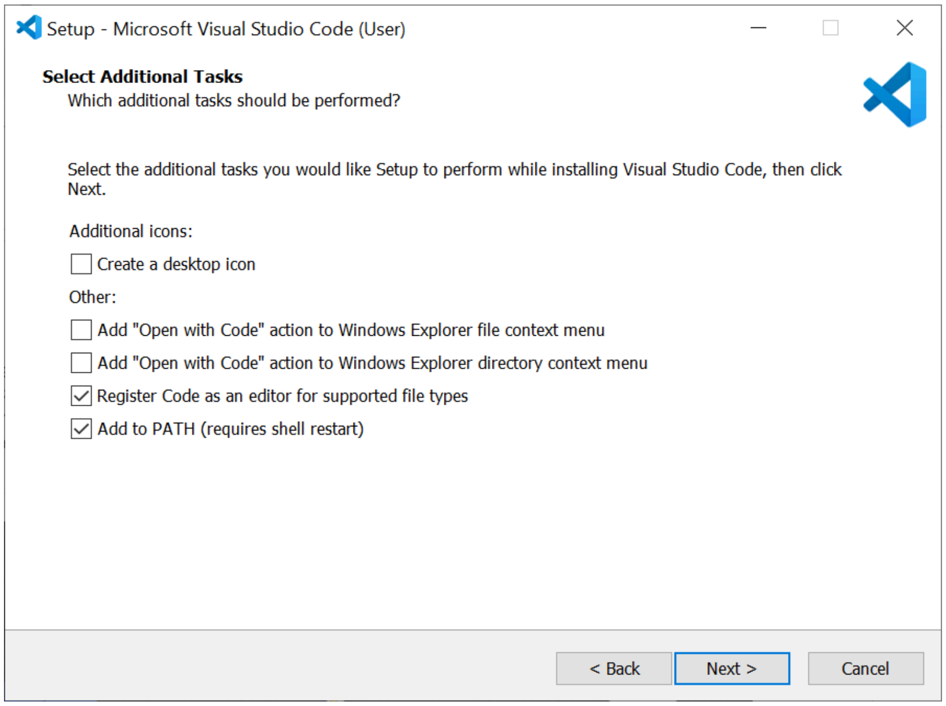
Free and built on open source. Integrated Git, debugging and extensions.

Windows	.deb	.rpm	Mac
Windows 10, 11	Debian, Ubuntu	Red Hat, Fedora, SUSE	macOS 10.15+

User Installer	x64 Arm64
System Installer	x64 Arm64
.zip	x64 Arm64
CLI	x64 Arm64

.deb	x64 Arm32 Arm64
.rpm	x64 Arm32 Arm64
.tar.gz	x64 Arm32 Arm64
Snap	Snap Store
CLI	x64 Arm32 Arm64

.zip	Intel chip Apple silicon Universal
CLI	Intel chip Apple silicon



Setup - Microsoft Visual Studio Code (User)

Select Additional Tasks
Which additional tasks should be performed?

Select the additional tasks you would like Setup to perform while installing Visual Studio Code, then click Next.

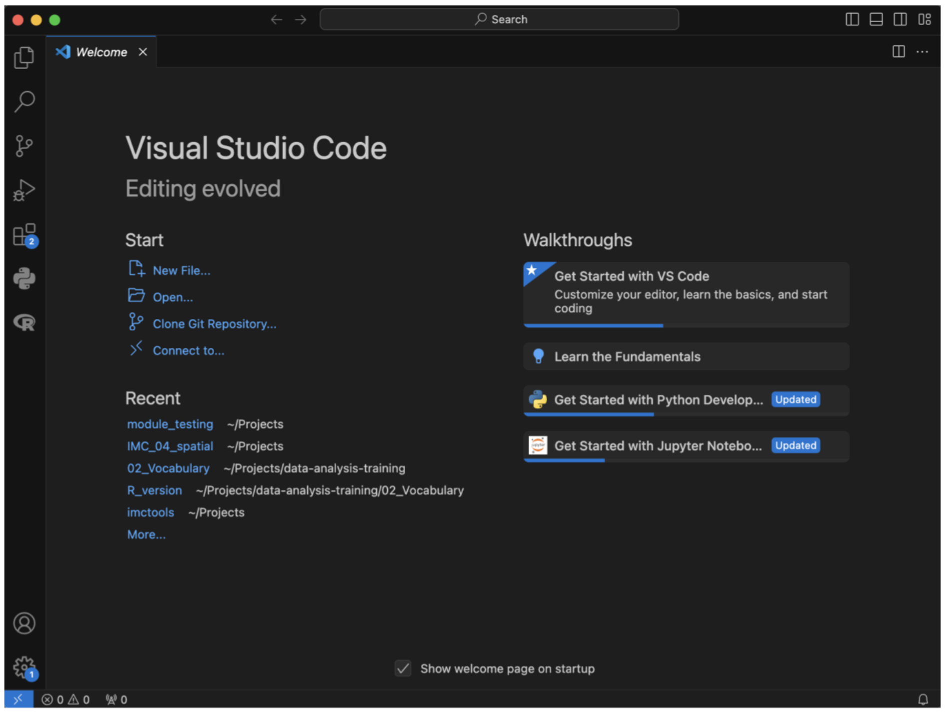
Additional icons:

- [ ] Create a desktop icon

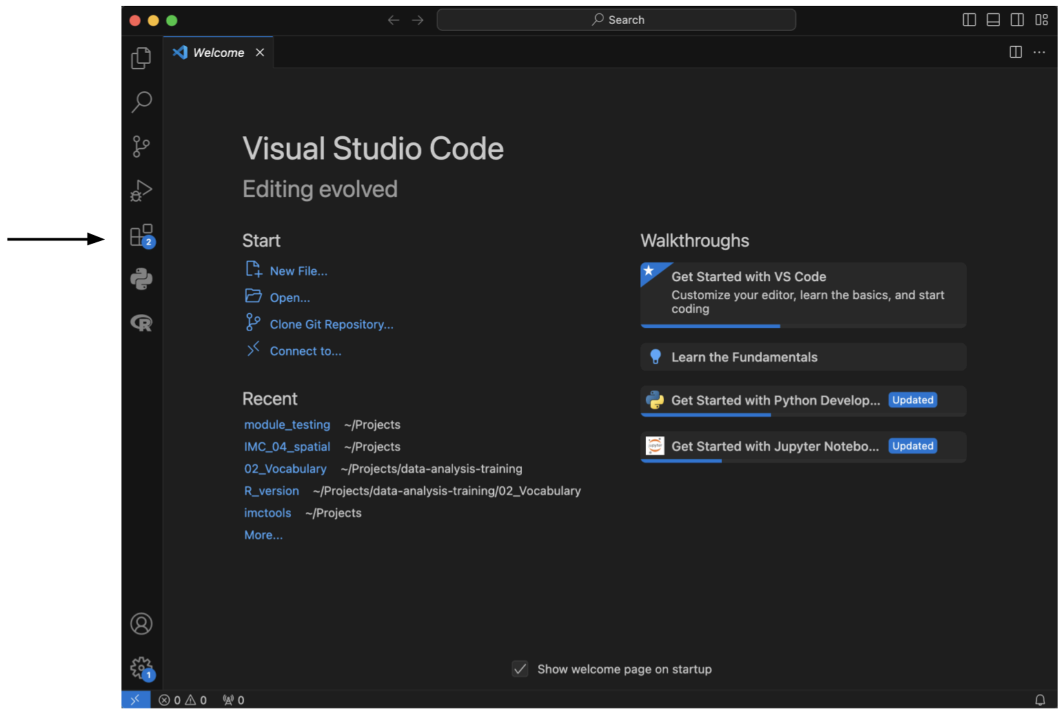
Other:

- [ ] Add "Open with Code" action to Windows Explorer file context menu
- [ ] Add "Open with Code" action to Windows Explorer directory context menu
- [✓] Register Code as an editor for supported file types
- [✓] Add to PATH (requires shell restart)

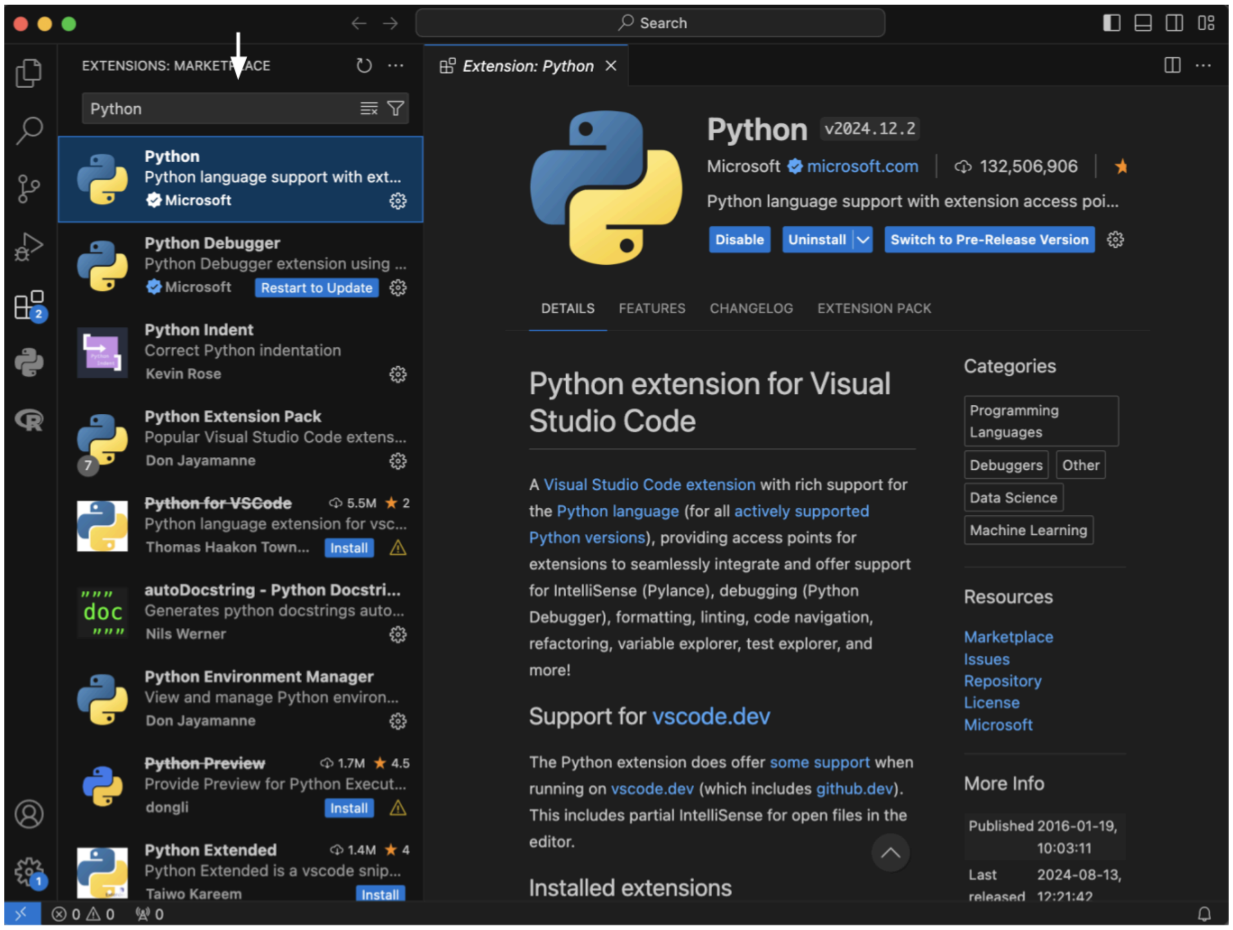
2. Open VSCode. You will be greeted with a Welcome page where you can explore various options and settings.


Search
Welcome
Visual Studio Code
Editing evolved
Start
New File...
Open...
Clone Git Repository...
Connect to...
Recent
module_testing ~/Projects
IMC_04_spatial ~/Projects
02_Vocabulary ~/Projects/data-analysis-training
R_version ~/Projects/data-analysis-training/02_Vocabulary
imctools ~/Projects
More...
Walkthroughs
Get Started with VS Code
Customize your editor, learn the basics, and start coding
Learn the Fundamentals
Get Started with Python Develop... Updated
Get Started with Jupyter Notebo... Updated
Show welcome page on startup

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.


Search
Welcome
Visual Studio Code
Editing evolved
Start
New File...
Open...
Clone Git Repository...
Connect to...
Recent
module_testing ~/Projects
IMC_04_spatial ~/Projects
02_Vocabulary ~/Projects/data-analysis-training
R_version ~/Projects/data-analysis-training/02_Vocabulary
imctools ~/Projects
More...
Walkthroughs
Get Started with VS Code
Customize your editor, learn the basics, and start coding
Learn the Fundamentals
Get Started with Python Develop... Updated
Get Started with Jupyter Notebo... Updated
Show welcome page on startup

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.



Python
Python language support with ext...
Microsoft

Python Debugger
Python Debugger extension using ...
Microsoft — Restart to Update

Python Indent
Correct Python indentation
Kevin Rose

Python Extension Pack
Popular Visual Studio Code extens...
Don Jayamanne

~~Python for VSCode~~ 5.5M ★ 2
Python language extension for vsc...
Thomas Haakon Town... — Install

autoDocstring - Python Docstri...
Generates python docstrings auto...
Nils Werner

Python Environment Manager
View and manage Python environ...
Don Jayamanne

~~Python Preview~~ 1.7M ★ 4.5
Provide Preview for Python Execut...
dongli — Install

Python Extended 1.4M ★ 4
Python Extended is a vscode snip...
Taiwo Kareem — Install

Python v2024.12.2
Microsoft microsoft.com | 132,506,906 | ★
Python language support with extension access poi...
Disable | Uninstall | Switch to Pre-Release Version

DETAILS FEATURES CHANGELOG EXTENSION PACK

Python extension for Visual Studio Code

A Visual Studio Code extension with rich support for the Python language (for all actively supported Python versions), providing access points for extensions to seamlessly integrate and offer support for IntelliSense (Pylance), debugging (Python Debugger), formatting, linting, code navigation, refactoring, variable explorer, test explorer, and more!

Support for vscode.dev

The Python extension does offer some support when running on vscode.dev (which includes github.dev). This includes partial IntelliSense for open files in the editor.

Installed extensions

Categories
Programming Languages
Debuggers
Other
Data Science
Machine Learning

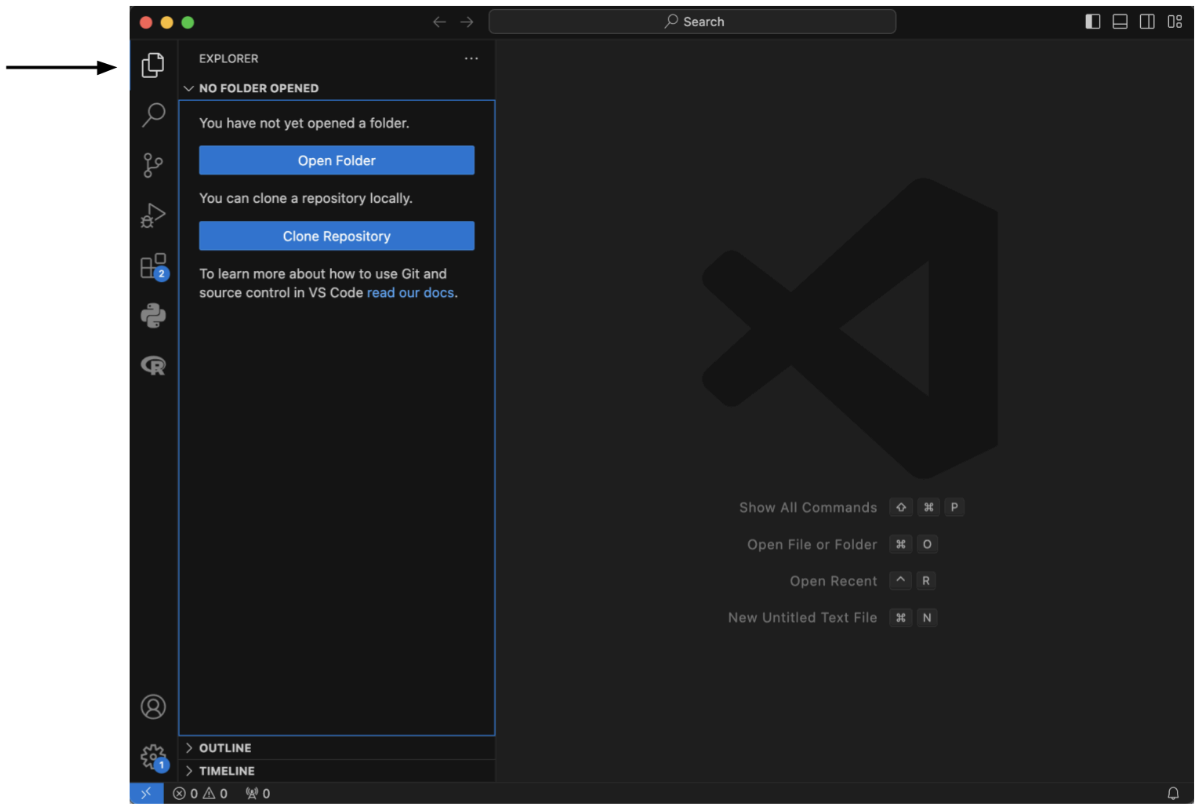
Resources
Marketplace
Issues
Repository
License
Microsoft

More Info
Published 2016-01-19, 10:03:11
Last released 2024-08-13, 12:21:42

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.

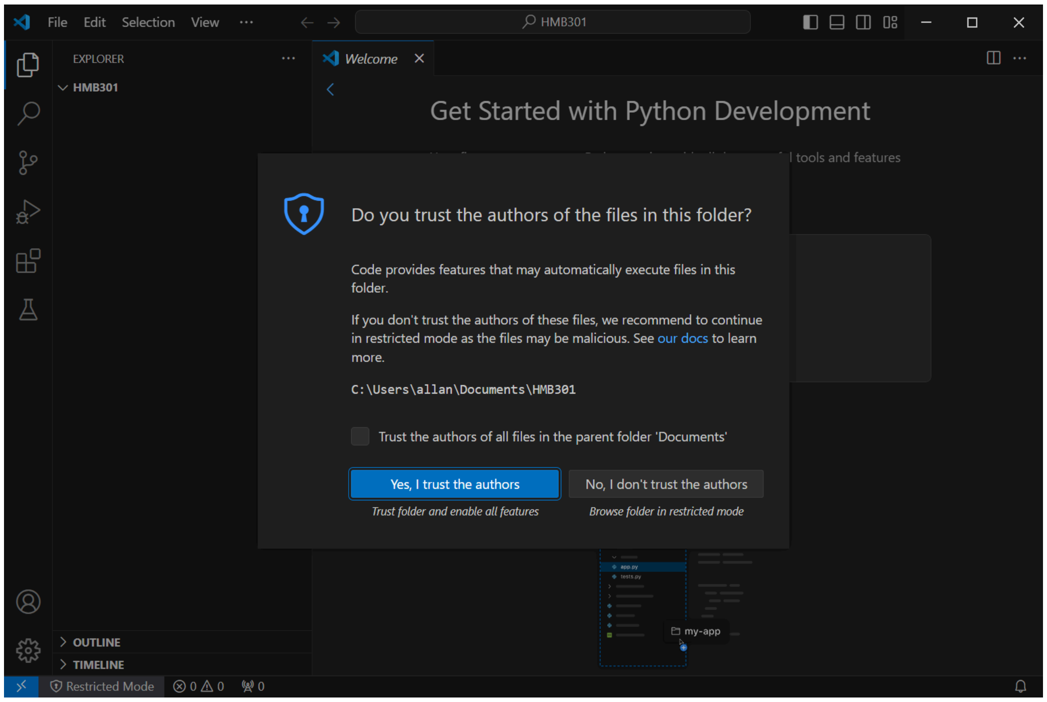
2. Go back to VSCode and click on the explorer tab on the left. This will take you to a new tab where you can search all available folders and files. This should be empty for now because we have not opened any folders or files yet.


You have not yet opened a folder.

You can clone a repository locally.

To learn more about how to use Git and source control in VS Code read our docs.

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”.



Do you trust the authors of the files in this folder?

Code provides features that may automatically execute files in this folder.

If you don't trust the authors of these files, we recommend to continue in restricted mode as the files may be malicious. See our docs to learn more.

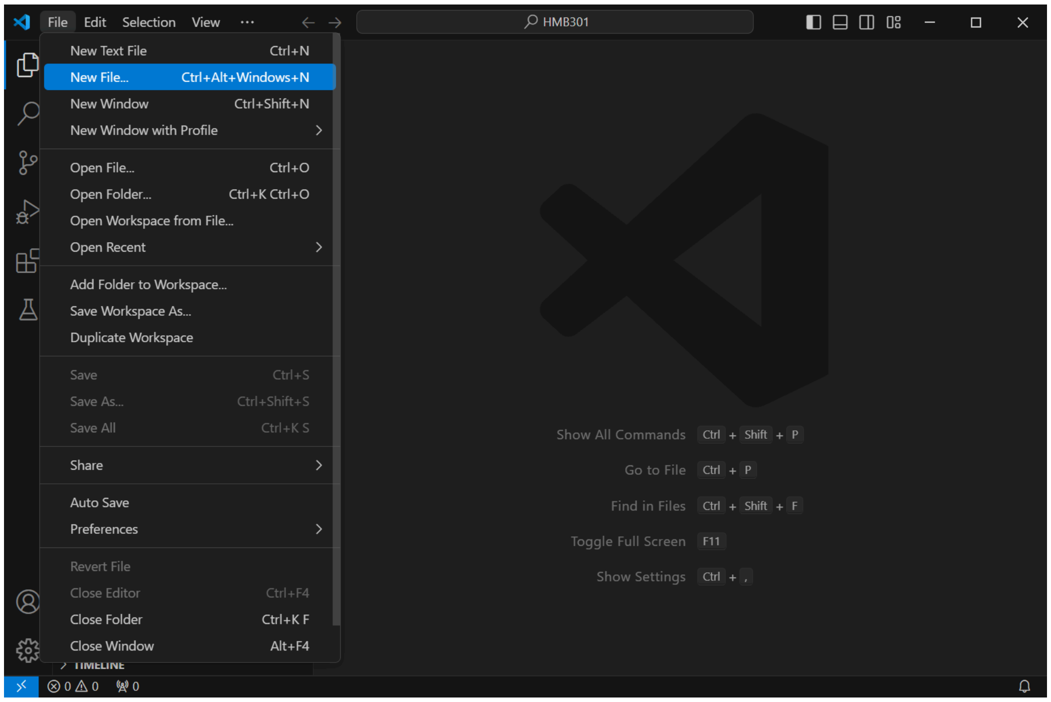
```
C:\Users\allan\Documents\HMB301
```

Trust the authors of all files in the parent folder 'Documents'

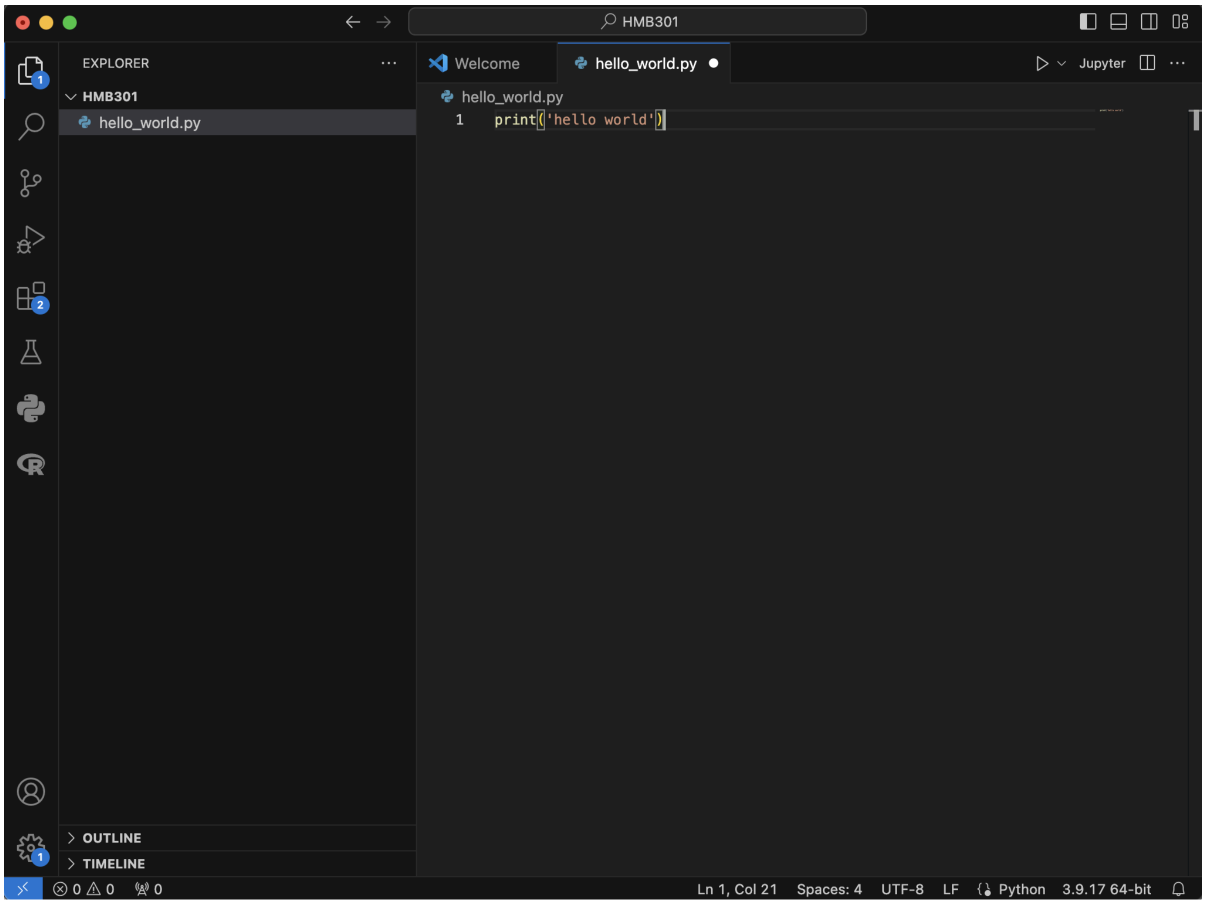
Yes, I trust the authors — Trust folder and enable all features

No, I don't trust the authors — Browse folder in restricted mode

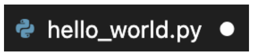
4. Next create a new Python file via the New File... button or the top menu bar as seen below.



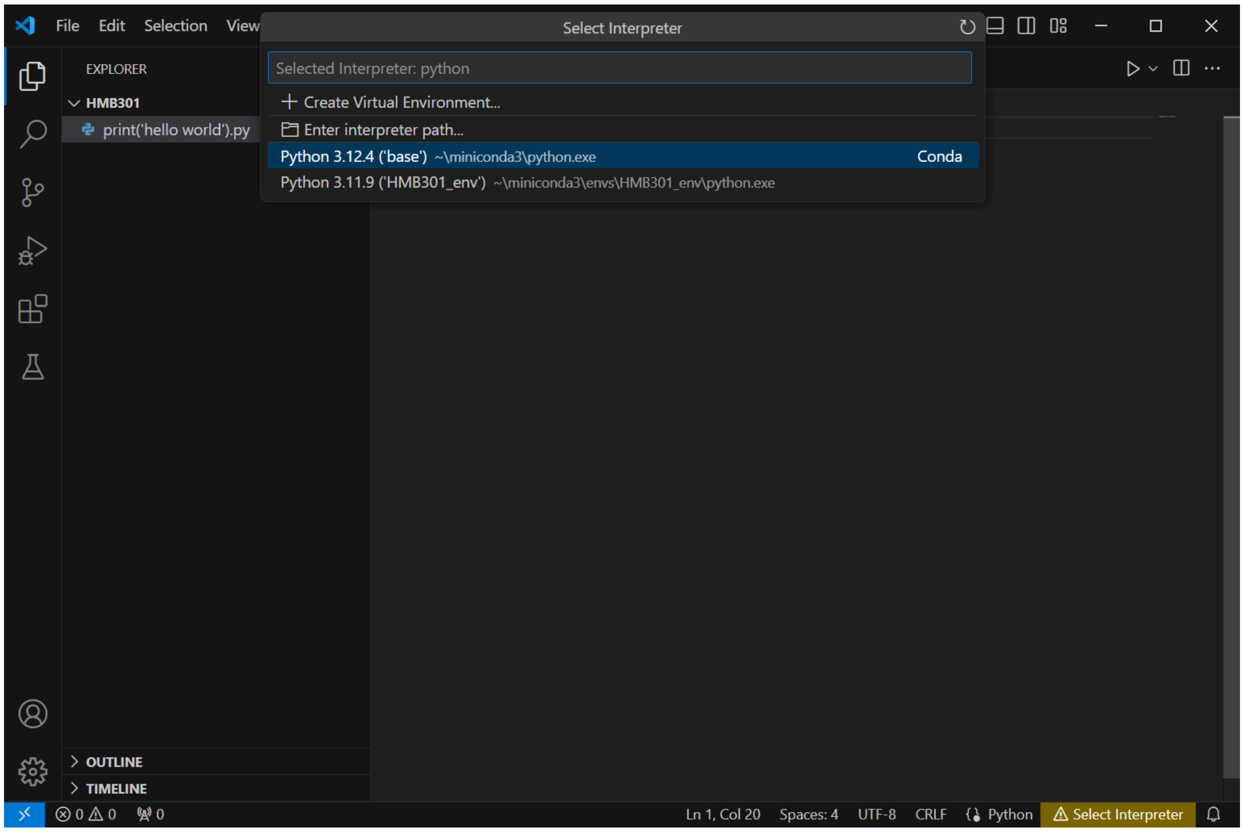
5. Type `print('hello world')` into the new Python file. Your window should now look like this.



```python
print('hello world')
```

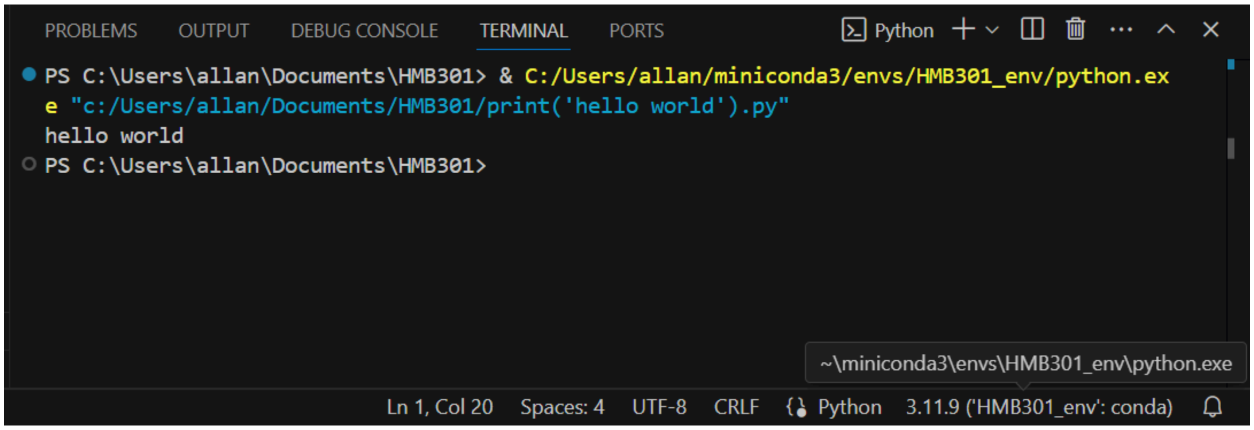
6. Notice that a circle on the right side of the file name  indicates that the change is not saved yet. Save the file like you would with most other applications, using Ctrl+S for windows, or navigate to the File menu to find the Save option.

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”.



```
Selected Interpreter: python
+ Create Virtual Environment...
Enter interpreter path...
Python 3.12.4 ('base') ~\miniconda3\python.exe                Conda
Python 3.11.9 ('HMB301_env') ~\miniconda3\envs\HMB301_env\python.exe
```

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.

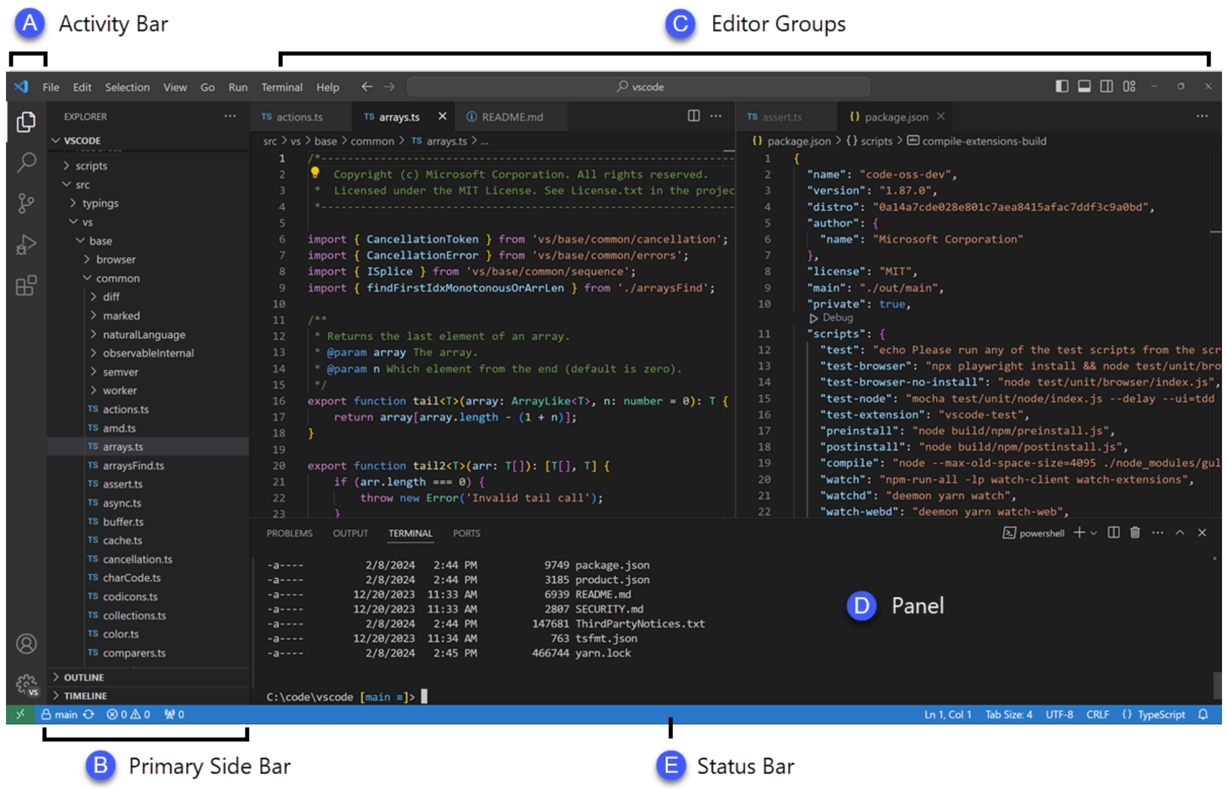


```
PS C:\Users\allan\Documents\HMB301> & C:/Users/allan/miniconda3/envs/HMB301_env/python.ex
e "c:/Users/allan/Documents/HMB301/print('hello world').py"
hello world
PS C:\Users\allan\Documents\HMB301>
```

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.



```
/*---------------------------------------------------------------------------
 *  Copyright (c) Microsoft Corporation. All rights reserved.
 *  Licensed under the MIT License. See License.txt in the projec
 *---------------------------------------------------------------------------

import { CancellationToken } from 'vs/base/common/cancellation';
import { CancellationError } from 'vs/base/common/errors';
import { ISplice } from 'vs/base/common/sequence';
import { findFirstIdxMonotonousOrArrLen } from './arraysFind';

/**
 * Returns the last element of an array.
 * @param array The array.
 * @param n Which element from the end (default is zero).
 */
export function tail<T>(array: ArrayLike<T>, n: number = 0): T {
    return array[array.length - (1 + n)];
}

export function tail2<T>(arr: T[]): [T[], T] {
    if (arr.length === 0) {
        throw new Error('Invalid tail call');
    }
```

```
{
  "name": "code-oss-dev",
  "version": "1.87.0",
  "distro": "0a14a7cde028e801c7aea8415afac7ddf3c9a0bd",
  "author": {
    "name": "Microsoft Corporation"
  },
  "license": "MIT",
  "main": "./out/main",
  "private": true,
  "scripts": {
    "test": "echo Please run any of the test scripts from the scr
    "test-browser": "npx playwright install && node test/unit/bro
    "test-browser-no-install": "node test/unit/browser/index.js",
    "test-node": "mocha test/unit/node/index.js --delay --ui=tdd
    "test-extension": "vscode-test",
    "preinstall": "node build/npm/preinstall.js",
    "postinstall": "node build/npm/postinstall.js",
    "compile": "node --max-old-space-size=4095 ./node_modules/gul
    "watch": "npm-run-all -lp watch-client watch-extensions",
    "watchd": "deemon yarn watch",
    "watch-webd": "deemon yarn watch-web",
```

```
-a----          2/8/2024   2:44 PM           9749 package.json
-a----          2/8/2024   2:44 PM           3185 product.json
-a----        12/20/2023  11:33 AM           6939 README.md
-a----        12/20/2023  11:33 AM           2807 SECURITY.md
-a----          2/8/2024   2:44 PM         147681 ThirdPartyNotices.txt
-a----        12/20/2023  11:34 AM            763 tsfmt.json
-a----          2/8/2024   2:45 PM         466744 yarn.lock


C:\code\vscode [main ≡]>
```

(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.