> *Before we install our python environment, we need to take care of a couple requirements.*

In step 1, we will install:

- Your "Terminal"/"Shell":
  - The primary application you will use to execute coding-related commands.
- A Python Distribution (Anaconda/miniforge):
  - The fundamental infrastructure that will allow us to install Python.
- GitHub Desktop:
  - The way we will work with git repositories and the starting point for our local workflows.

Tool #1: A Linux-based bash shell/terminal:

- Linux users and MacOS users have a Terminal app pre-installed.

Tool #2: GitHub Desktop

GitHub Desktop is not currently natively available for Linux. Instead, you will be using git commands from your Terminal.

- TEST: To install GitHub Desktop for Linux, run the commands in this script in your Terminal.
  - https://gist.github.com/berkorbay/6feda478a00b0432d13f1fc0a50467f1
- Once installation is complete, open the application.
  - Log into the same GitHub account you have been using for your projects.
- Once you have logged into the app, open the Options menu
  - Select `"GitHub Desktop"` on the menu bar (top of the screen) and then select `Preferences`.
  - Select the "Integrations" tab.
  - Make sure the Shell dropdown menu says "Terminal"
    - If not, select it from the dropdown menu.
  - Click Save.

Alternative Tool #2: Git (terminal-based)

- Identify the correct installation command based on which version of Linux you are using:
  - https://git-scm.com/download/linux

Tool #3: Python Distribution - Anaconda

- Anaconda is a data-science-focused python distributable that comes with a convenient GUI program for working with our python environments.
- Follow the instructions in this guide in the "Installing Anaconda" section
  - **Stop when you get to the "Setting Up Anaconda Environments" section.**

> *You are all set to move on to the next step "2. Setting Up Your dojo-env Environment"*

2. Setting Up Your dojo-env Environment

Step 2 Overview:

- In Step 1, we installed the foundational tools needed for our local python installation.
  - While we did install a Python distribution with a basic copy of Python (Anaconda or miniforge), we have not installed all of the packages and tools that we need as data scientists.
- In Step 2, you will be creating a custom python environment called "dojo-env".
  - An "environment" is a bundle of specific python packages that are used together. Importantly, an environment specifies specific version #'s of the packages to ensure that all of the versions installed are mutually compatible.

    - You can install many environments on your computer and switch between them as needed for different projects.
    - We have designed the dojo-env to include everything you'd need for our program, so you may not have a reason to add additional environments.

Brief Summary of the Following Steps:

- Step 2.1: Clone the dojo-env-setup repository
- Step 2.2: Open the repo in your terminal/GitBash
- Step 2.3: Create the dojo-env environment
- Step 2.4: Setting dojo-env as your default.

[Steps 2.1 & 2.2 - Using Git Commands]

If you were not able to get GitHub Desktop installed and installed git for Linux instead, follow these instructions to complete steps 2.1 and 2.2.

Step 2.1 & 2.2: Clone the dojo-env-setup repository with your terminal

- In your Terminal, navigate to a folder where you would like to keep your data science materials using the cd command.
- Once you're in the folder where you want to store your data science materials, run the following command to clone the repository to your local machine:

```
git clone https://github.com/coding-dojo-data-science/dojo-env-setup.git
```

- This will download a new folder called "dojo-env-setup", which contains the environment creation files.
- Navigate your terminal to the new "dojo-env-setup" folder using the cd command

```
cd dojo-env-setup
```

Verify You Are in the Correct Folder

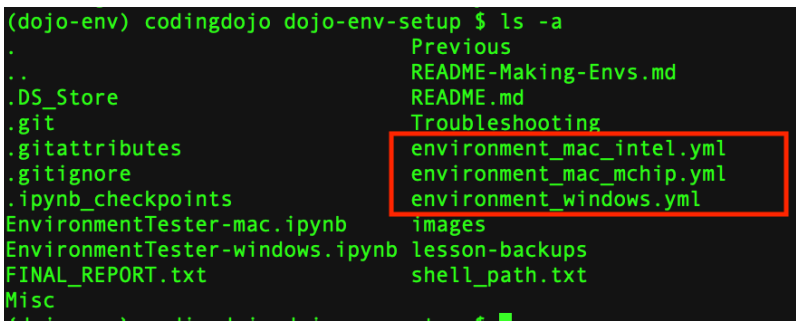
Run one last command to verify that you are indeed in the correct folder.

Run the "ls -a" command to see a detailed list of all files in the repo.

```
ls -a
```

You should see a list of all the files in the current folder.

If you are in the right folder, you should see 3 files that start with "environment" and end with ".yml" like in the screenshot below.



If so, you are all set for the next step: create the dojo-env environment!

Remember the Repo's Location

We will need to return to this folder later. Run the "pwd" command in your terminal to see the full file path to the current folder.

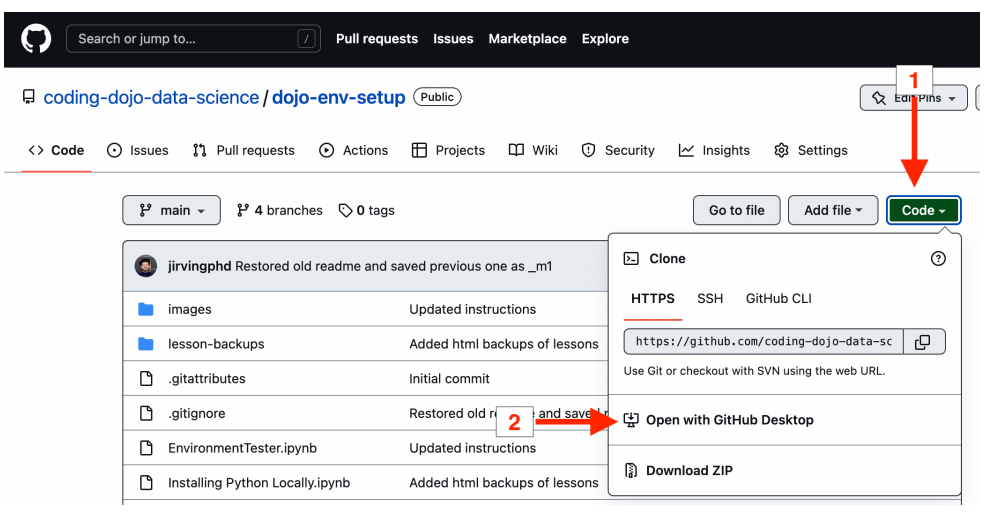
Either memorize this or write it down/copy it somewhere. Later you will need to "cd REPO_FOLDER" (where REPO_FOLDER will be whatever was displayed when you ran pwd).

[Steps 2.1 & 2.2 - Using GitHub Desktop]

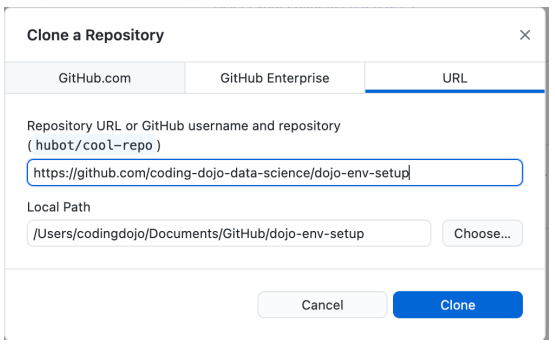
If you were able to install GitHub Desktop, you can follow these commands.

Step 2.1: Clone the dojo-env-setup repository [Using GitHub Desktop]

1. Open the dojo-env-setup repository on GitHub.com:
    1. https://github.com/coding-dojo-data-science/dojo-env-setup
2. Make sure that :
    1. you are logged in to your account on GitHub.com
    2. and you are logged into the SAME account in the GitHub Desktop app (that you installed in step 1.)
3. Click on the green `Code` button and then click `Open in GitHub desktop.`

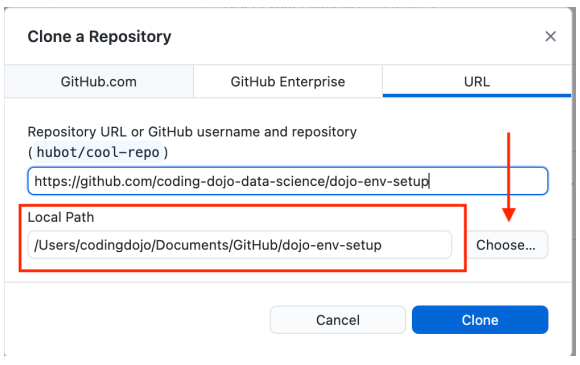


1. GitHub desktop should open automatically and ask you what folder you would like to store your repository in.



    1. Troubleshooting Note: if you are brought to the Download GitHub Desktop web page instead:
        1. It means you were not logged into the same account on GitHub.com and GitHub Desktop when you clicked Open in GitHub Desktop.
            1. Make sure you see your user profile pic in the top right of GitHub.com
            2. and check your Account in GitHub Desktop's Preferences/Options menu.
            3. and then try again.
    1. By default, GitHub Desktop will use a new "GitHub" folder in your Documents folder
        1. GitHub Desktop will create a NEW folder with the same name as the repository INSIDE of whichever folder you select.
        2. If you use the default options, then this will create a "dojo-env-setup/" folder inside of "Documents/GitHub/"
        3. Note: it is strongly recommended that you use the Documents/GitHub folder for this repository.
            1. But if you'd rather save the folder somewhere else:
                1. Use the "Choose" button (the button name may be "Browse" on Windows).
                2. A window should pop up for you to find and click on the folder where you want to create the "dojo-env-setup" folder.
                3. Once you have selected a new folder using the Browse button, you should see the full folder path displayed.

4. IMPORTANT STEP: Make sure to remember the full file path of the folder you selected! (See the screenshot below.)

1. 

Step 2.2: Open the Repo in Terminal/GitBash

Open the Repo in Terminal

Once you have cloned the repository, you will need to open a terminal/gitbash window in the same folder as the repository.

- Open a new terminal in the dojo-env-setup folder
  - First, in GitHub Desktop: make sure the left sidebar says "dojo-env-setup" in the top-left corner under Current Repository.
  - Click on the Repository menu and select " `Open in terminal` " or " `Open in gitbash` "
    - Mac Users: the menu will appear at the very top of your screen (your menu bar).
    - Alternatively, you can use the keyboard shortcut to do the same thing. The command for both Mac and Windows is:
      - Control + ` (the key above tab that also has the tilde symbol ~)
- In the terminal window that appears, type the "pwd" command (stands for print working directory) and press Enter.
  - It will display the folder name of the folder your terminal is currently located.
    - The folder path should end in "dojo-env-setup/"
    - If you used the default GitHub folder when you cloned dojo-env, the full filepath would be something similar to "/Users/yourname/Documents/GitHub/dojo-env-setup/"

Verify Step 2.2

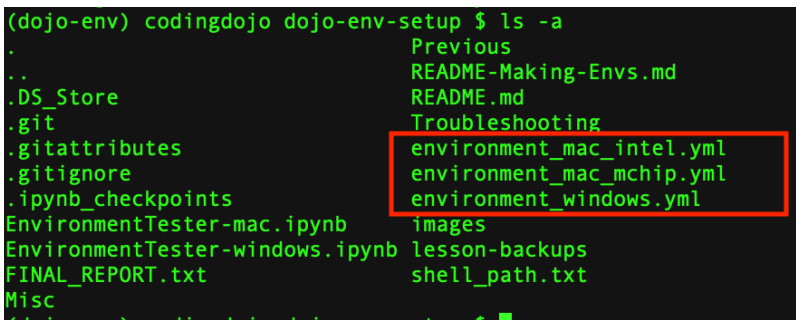
Run one last command to verify that you are indeed in the correct folder.

Run the "ls -a" command to see a detailed list of all files in the repo.

```
ls -a
```

You should see a list of all the files in the current folder.

If you are in the right folder, you should see 3 files that start with "environment" and end with ".yml" like in the screenshot below.



If so, you are all set for the next step: create the dojo-env environment!

Step 2.3 Create the dojo-env environment

1) Create the dojo-env using "conda create"

- Linux users will use the same environment file as Mac Computers with an Intel

```
## Env Creation Commands - LINUX (Same command as Macs with Intel)
conda env create -f environment_mac_intel.yml
```

- Wait (patiently) for the dojo-env to be created.
    - It can take anywhere from 3-20 minutes to finish create the environment, depending on your computer and internet connection.
    - You will see several progress bars during the process. Once it has been completed you should see a message that says

```
# To activate this environment use:
conda activate dojo-env
# To deactivate this environment use:
conda deactivate
# If conda deactivate doesn't work, activate the "base" env conda activate base
```

- Confirm your environment was installed and activate it.
    - Type `conda env list` to display the list of your locally installed environments.
        - You should see 2 environments, including dojo-env:
            - `base`
            - `dojo-env`
- If you see dojo-env in the list:
    - Success! dojo-env was successfully created! But we aren't using it yet just yet. We must first "activate" an environment to determine which version of python & packages are currently being used.

2) Activate the dojo-env

- Run the conda activate command to switch to dojo-env.

```
conda activate dojo-env
```

- You should now see "(dojo-env)" next to your prompt in your terminal (may be above the prompt, on the left, or on the right depending on your OS)

3) Add dojo-env to jupyter notebook/lab

- After confirming you now see (dojo-env) displayed next to your prompt:
    - Run the following command to make sure Jupyter Notebook/Lab knows your new environment.

```
python -m ipykernel install --user --name dojo-env --display-name "Python (dojo-env)"
```

The moment of truth...

You are all set for the next step: Testing Your New Environment!

Step 2.4: Setting dojo-env as the default + alias commands

- This section will require you to enter several commands in your Terminal.
    - Make sure that your terminal is not running jupyter notebook (you can press " `Cntrl` + `C` " to force quit the server from your terminal).
    - Alternatively, you can open a new terminal. (You can perform these steps from any folder.)

Determine If your Terminal is Using Bash or Zsh

- On Mac, there are multiple options for what shell the Terminal app uses.
    - There are 2 possible options that your mac may be using:
        - zsh
        - bash
    - The major difference between zsh and bash is which file it checks for the settings to use when you create a new terminal.
        - Otherwise, zsh and bash behave very similarly and you may not notice the difference if you switched.
        - Note: If you have a newer mac (last 1-2 years) and have not previously used your Terminal, you are likely using zsh.

- To confirm which type of shell your terminal is using:
    - in your terminal, type `echo $SHELL` and hit enter.
        - Select option A or B below based on the output.

A) If the response ends in bash

- Run the following commands to automatically activate dojo-env

```
touch ~/.bash_profile
echo "conda activate dojo-env" >> ~/.bash_profile
```

- Run the following 2 commands to add shortcuts for opening jupyter
    - NOTE: it is VERY important that you do not add any spaces on either side of the `=` sign. The command will not work correctly if you add extra spaces.

```
echo 'alias jnb="jupyter notebook"' >> ~/.bash_profile
echo 'alias lab="jupyter lab"' >> ~/.bash_profile
```

B) If the response ends in zsh:

- Run the following commands to automatically activate dojo-env

```
touch ~/.zshrc
echo "conda activate dojo-env" >> ~/.zshrc
```

- Run the following 2 commands to add shortcuts for opening jupyter
    - NOTE: it is VERY important that you do not add any spaces on either side of the `=` sign. The command will not work correctly if you add extra spaces.

```
echo 'alias jnb="jupyter notebook"' >> ~/.zshrc
echo 'alias lab="jupyter lab"' >> ~/.zshrc
```

2) Final Verification Steps

Confirm `dojo-env` is your default

- To confirm that `dojo-env` is now your default environment:
    - You should see `(dojo-env)` appear next to your prompt.

Confirm the shortcut aliases work

- Try running the command "jnb" in your terminal/kitbash.
    - If jupyter notebook launches, you're all set!
    - If not, contact your instructor or a TA for assistance.

Note: you can still move on to the next step even if you

could not successfully complete this step 2.4

- These steps are for convenience and are not strictly required.
- You should still contact your instructor or a TA for assistance to successfully complete this step at your earliest convenience.

Step 2.5: Testing the Environment

To test that your installation and packages are working properly. We are going to run a specific Environment Testing notebook that is also located in the "dojo-env-setup" folder.

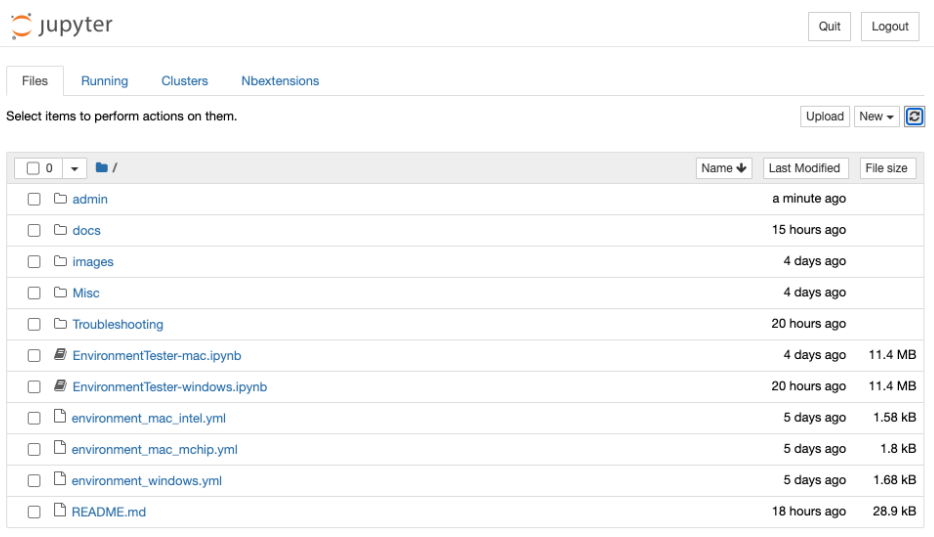
Running the environment tester notebook with jupyter notebook

- Next, you will close all of your previous Terminal/GitBash windows BUT before you do:
    - if your terminal is still running jupyter notebook and you do not see the prompt waiting for a command:
        - You must press "Control +C" to force-quit jupyter.
        - Make sure to reply "y" if asked for confirmation.
    - If the cursor appears waiting for a new command, you are all set.

- **If you used git terminal commands for steps 2.1-2.2:**
    - You should have remembered the file path to the dojo-env-setup repo in step 2.1.
    - In a new terminal window, use the cd command to cd back to the dojo-env-setup folder.

- **If you used GitHub Desktop for steps 2.1-2.2:**
    - Now, return to GitHub desktop and click on the "Open in Terminal/GitBash" to open a terminal in the dojo-env-setup folder.
        - Type pwd to confirm it says dojo-env-setup.
        - Note: if you do not see the button for Open in Terminal:
            - Click on the menu for "Repository" at the very top of the window (if using Windows) or at the very top of your screen on your menu bar (if using a Mac).
            - You should see the word "Repository" next to the FIle, Edit, View menus.
                - From the Repository menu: click on Open in Terminal/GitBash

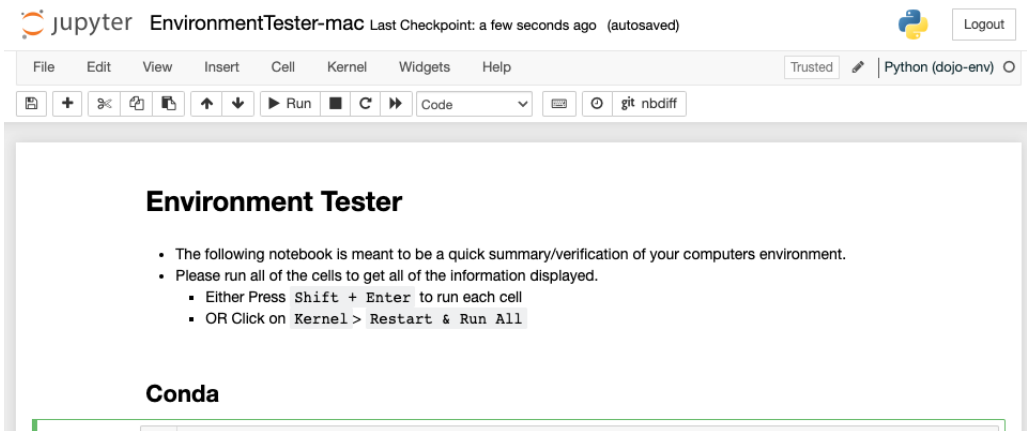
- **In the new window that opens, start jupyter notebook by entering the `jupyter notebook` command in your terminal (or the "jnb" shortcut!)**
-     - A new tab should open in your web browser that shows the File view for jupyter notebook.
    - You should see all of the files that were in the dojo-env-folder.



- There are 2 "EnvironmentTester" notebooks:
    - "EnvironmentTester-mac.ipynb" for macs (both Intel and Apple Chip macs)
    - "EnvironmentTester-windows.ipynb" for Windows.

- Click on the test notebook for your OS to open it.

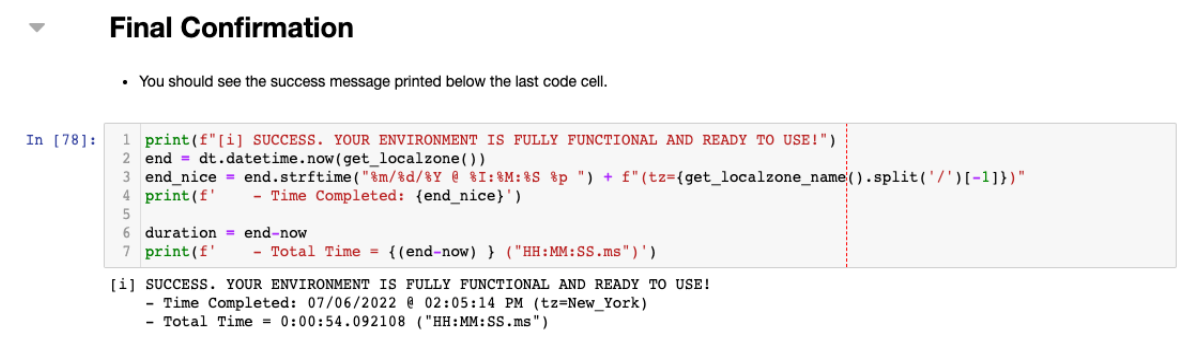
    - Linux users should use the "EnvironmentTester-mac.ipynb" notebook
        - Once the notebook interface has loaded, you should see a toolbar with several menu choices.



- We want to run all of the cells in this notebook and confirm it can make it to the end without errors.

- To Run the Entire Notebook:
    - Select the "Kernel" Menu > "Restart and Run All"
    - Wait patiently. The testing notebook is going to run through several modeling and EDA steps to confirm that the packages are working correctly.
        - This could take anywhere from 2-10 minutes to run.
        - You will see the web browser tab icon turn to an hourglass when the notebook is running and back to an orange notebook icon when it is done.

- Scroll down to the bottom of the notebook and confirm the cells have run:
    - Check if the very last cell printed the success message.



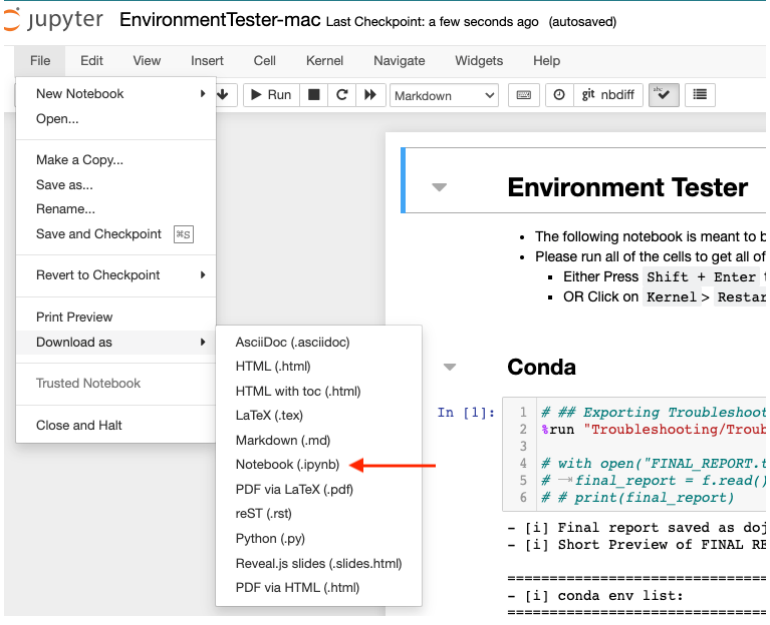
- If the entire notebook ran successfully
    - Congrats! Your dojo-env is fully functional and you can move on to the next step/lesson!

- If your notebook did not run the entire notebook successfully:
    - You need to contact your instructor or a TA for assistance.
    - Before contacting them, please follow the instructions below to prepare the troubleshooting files to give to your instructor.

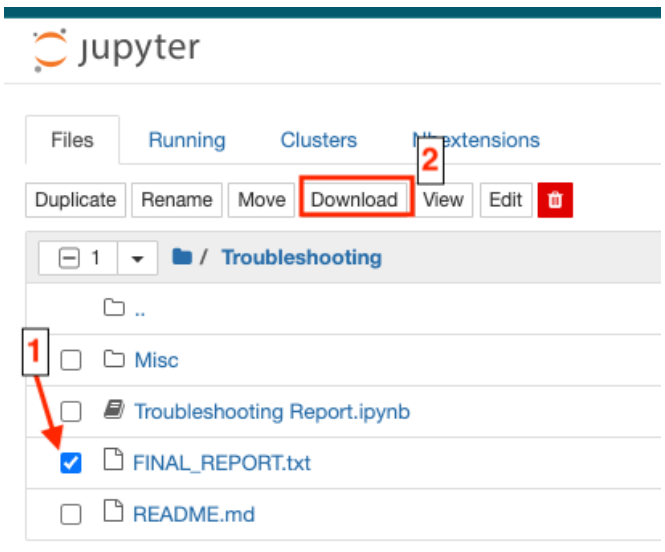
To Get Help Troubleshooting Your Environment.

- There are 2 files that you should share with your instructor/TA
    1. A copy of your Environment Tester notebook that error'd.
    2. A copy of "FINAL_REPORT.txt" file that is in the Troubleshooting folder of the repo.

1. To share your notebook with an instructor/TA for help:
    - Click File > Save & Checkpoint.
    - Click File > Download As > Notebook (.ipynb)
    - Your web browser should save a copy of the notebook to your normal "Downloads" folder.



2. To share a copy of your FINAL_REPORT.txt:
    - In the first Files tab that opened when you started jupyter notebook you should see a folder called "Troubleshooting"
    - Click on the Troubleshooting folder.
    - Inside the folder you should have a file called "FINAL_REPORT.txt".
    - Check the checkbox next to the file and click on the "Download" button that appears at the top of the list of files.
    - Your web browser will also save this file to your Downloads folder.



- Send an email to your instructor
    - Attach the 2 files listed above. They are located in your Downloads folder.
    - Add any additional details or info you think may be helpful for us to know.
        - For example:
            - "my computer is really old and I think that may be part of the problem."
            - "I share this computer with someone else who also uses python"
- An instructor or TA will get back to you within 1 business day with the next steps for you to try.
    - You will most likely need to set up a Zoom call and share your screen for us to help.

Step 2.6 Adding nbextensions

Jupyter Notebook Extensions Resources

- Documentation
- Official `nbextensions` Installation Instructions (also detailed below)

Installing Using Pip

- Below is an abbreviated version of the official instructions for Installing jupyter-contrib-nbextensions (Documentation):
    1. Install extensions

```
pip install jupyter_contrib_nbextensions
```

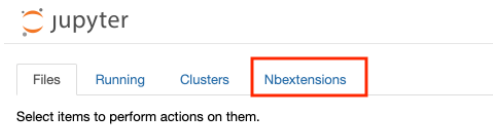
    1. Install additional requirements (Install javascript and css file):

```
jupyter contrib nbextension install --user
```

1. Activate the extension configurator

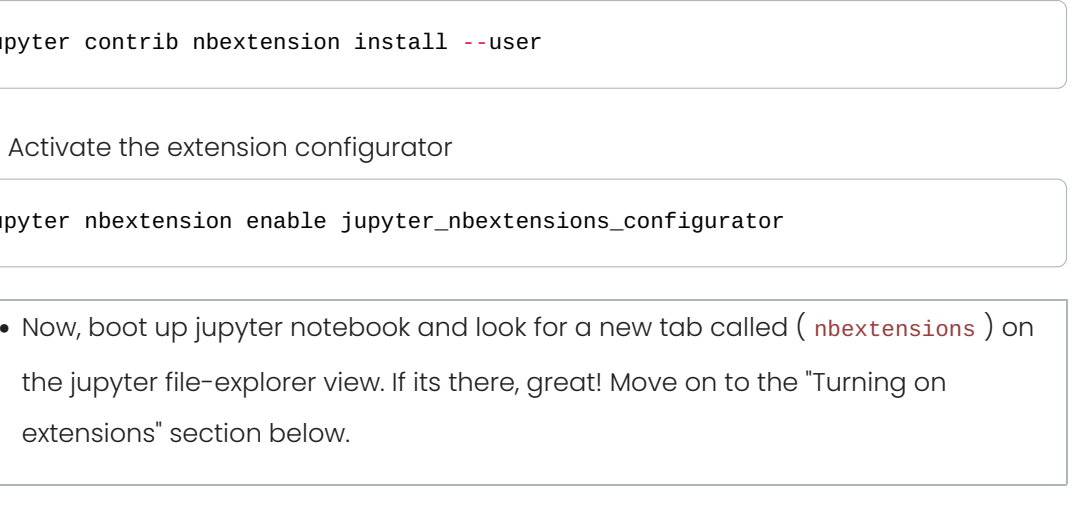
```
jupyter nbextension enable jupyter_nbextensions_configurator
```

- Now, boot up jupyter notebook and look for a new tab called (`nbextensions`) on the jupyter file-explorer view. If its there, great! Move on to the "Turning on extensions" section below.



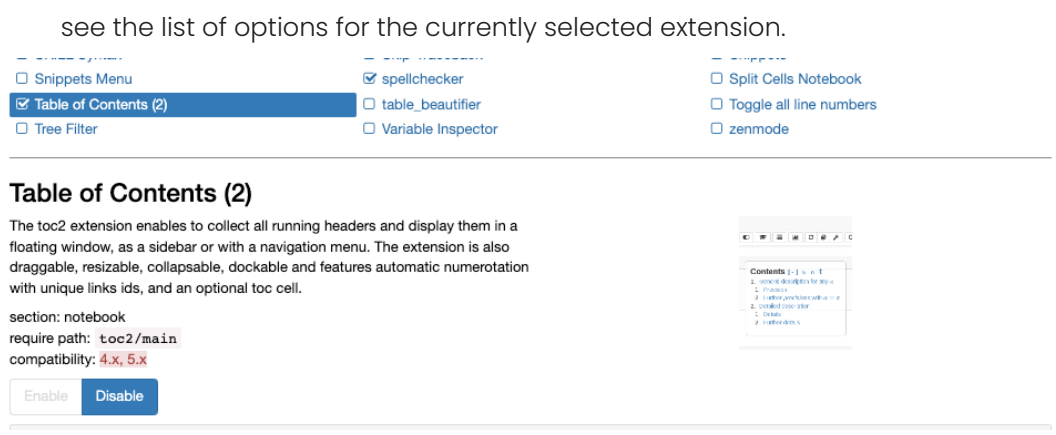
Activating Specific Extensions:

- When you boot up jupyter notebook, there should be a new tab at the top called `nbextensions` . Click on the tab to open the list of available extensions.
- This opens a menu of all of the available extensions with checkboxes to activate them;
  - *NOTE: If the list of available notebook extensions is grayed out like the screenshot below:*



    - Uncheck " `disable configuration for nbextensions without explicit compatibility (they may break your notebook environment, but can be useful to show for nbextension development)` " at the top of the page next to the search box.

- To enable the recommended extensions:
  - Click on the checkbox next to the extensions name to activate the extension.
- To change the settings for an extension:
  - Click on the name of the extension to select it. Now, if you scroll down, you should see the list of options for the currently selected extension.



- Note: any extensions that you enable or settings that you change are *automatically* saved.

Recommended Extensions & Settings

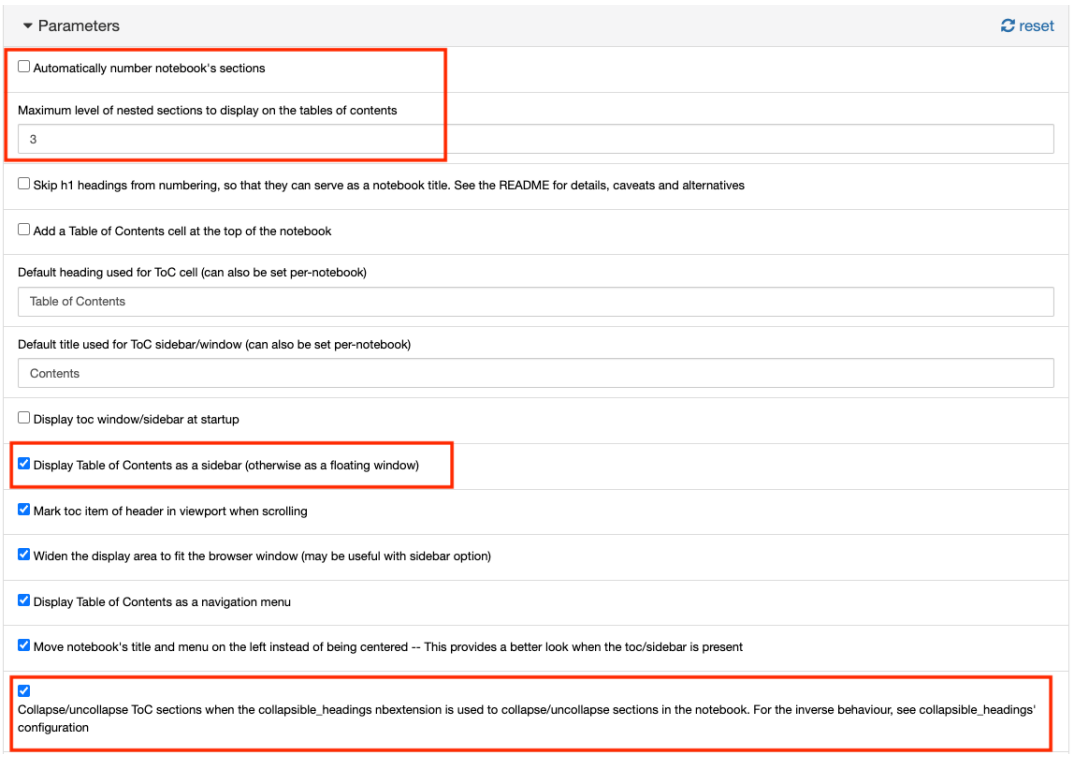
The following section will walk you through each of the recommended extensions and their recommended settings.

Brief Summary of Extensions to Enable

- Table of Contents (2)
- Collapsible Headings
- Live Markdown Preview
- Ruler
- spellchecker

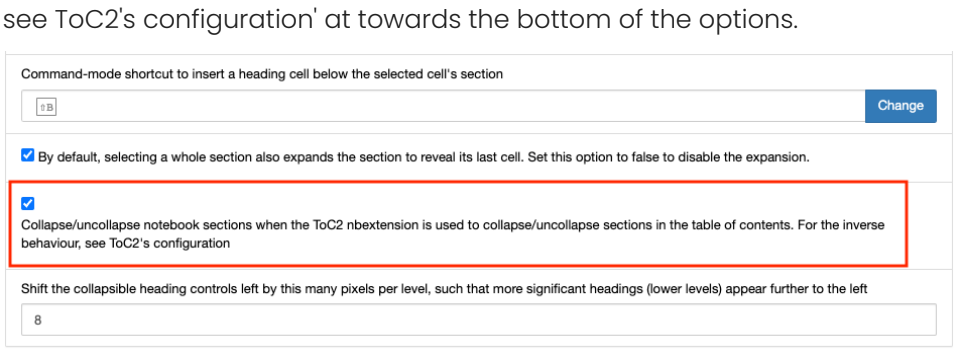
1. Table of Contents (2):

- Clickable sidebar with markdown headers as bookmarks/links.
- Recommended options:
  - Uncheck Automatically number notebook sections
  - Change `Maximum level of nested sections to display on the tables of contents` to 3.
  - Check `Display Table of Contents as a sidebar (otherwise as a floating window)`
  - Check `Collapse/uncollapse ToC sections when the collapsible_headings nbextension is used to collapse/uncollapse sections in the notebook. For the inverse behaviour, see collapsible_headings' configuration`



2. Collapsible Headings

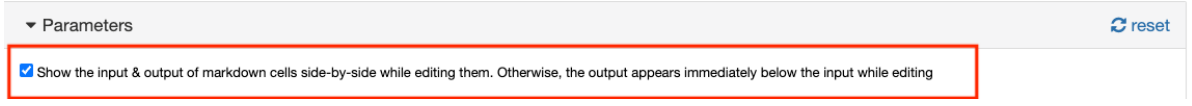
- Collapse sections of your notebook using markdown headers.
- Recommended options:
  - Check 'Collapse/uncollapse notebook sections when the ToC2 nbextension is used to collapse/uncollapse sections in the table of contents. For the inverse behaviour, see ToC2's configuration' at towards the bottom of the options.



3. Live Markdown Preview

Shows a preview of what the markdown cell you are editing will look like once you render it with Shift+Enter

- - Recommended options:
    - Check `Show the input & output of markdown cells side-by-side while editing them.`



4. Ruler (not Ruler in Editor)

- Adds a vertical red line in code cells at 80 characters. Python code should not cross this line (to match Python's style guide)
- Settings:
    - No settings to change.

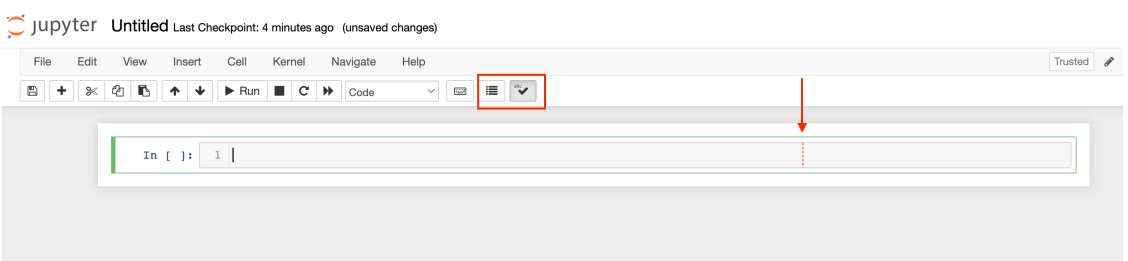
5. spellchecker

Checks markdown cells for spelling and highlights words in red that are misspelled. Note: it cannot correct misspelled words, only highlight them.

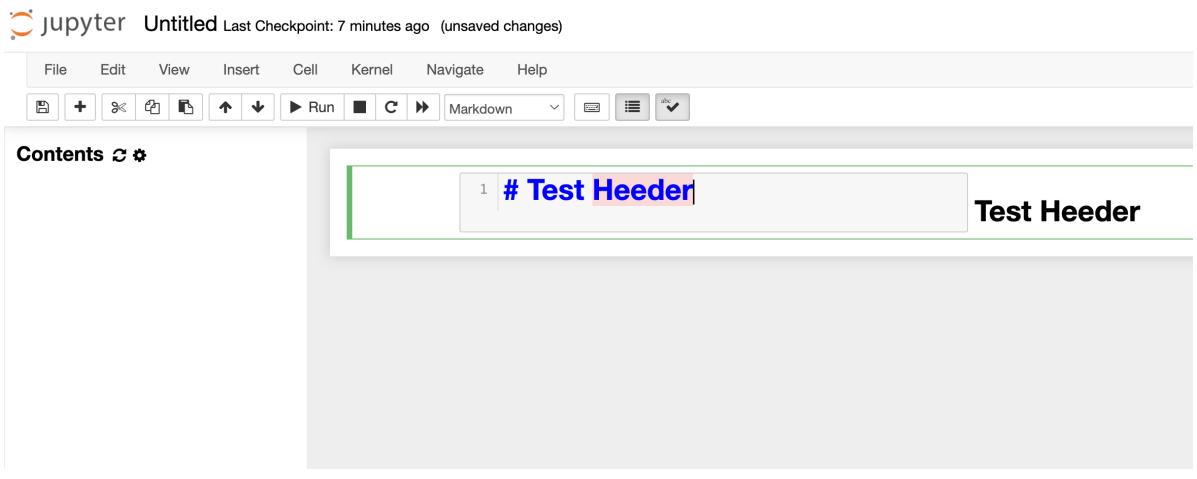
Confirming Extensions are Enabled

- Go back to the Files tab and create a new notebook with the `New` button on the top-right.
    - Select `Python(dojo-env)` for your kernel

- Once your new Untitled notebook opens, you will notice a few new elements to the interface:

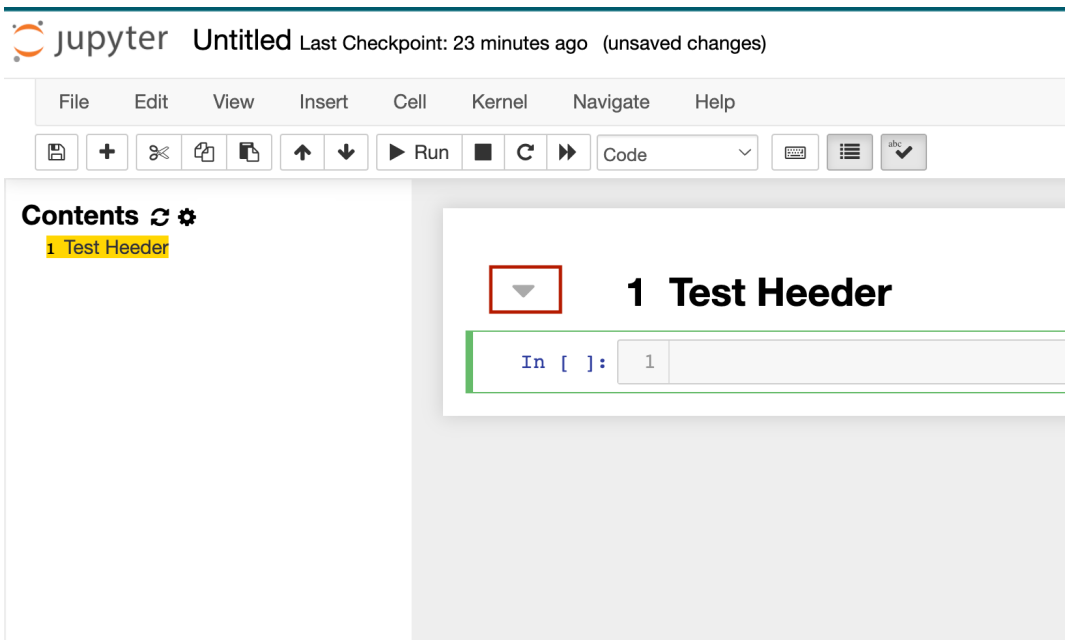


- First, confirm that you have two new buttons on your toolbar:
    - One that looks like a list (this is your table of contents extension)
    - One that looks like a checkmark (this is your spellchecker)
- Second, confirm that you see a red dashed line in your code cell. (the Ruler extension)
- Third, click on the button for the table of contents (the one that looks like a list).
    - An empty sidebar should appear on the left.
- Fourth, change your code cell to a markdown cell.
    - You can click on the dropdown menu on your toolbar that currently says "code".
        - Change this to Markdown.
        - In the markdown cell, type the following text but do NOT run the cell yet.
        - `# TEST HEEDER` (misspelled on purpose).

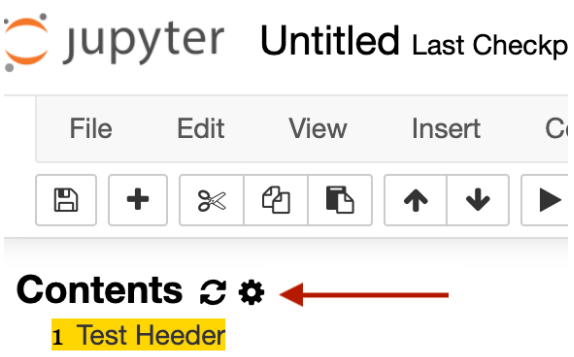


- Confirm that you see a preview of your markdown text off to the right.
    - this is your Live Markdown Preview extension.
- Confirm that the word "HEEDER" is highlighted in red.
    - This is your spellchecker.

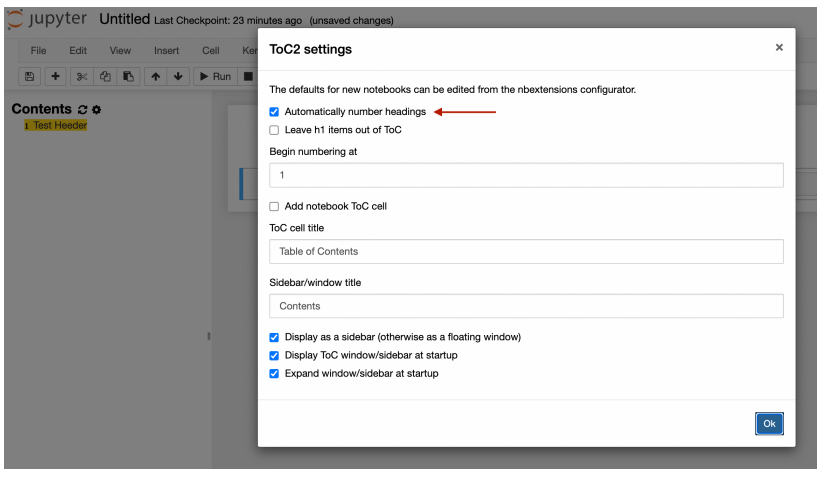
- Finally, run the cell "Shift+Enter" and confirm:
    - that the header appears in the table of contents on the left.
    - that a dropdown arrow appears to the left of the header in the notebook.



- You may notice that the ToC automatically numbered the header and added `1` next to Test Heeder.
    - If you prefer to disable this, click on the small gear icon next to the word Contents:



    - In the menu that appears, uncheck "Automatically number headings"



Now you are all set with your Jupyter Notebook extensions! Onto the final step 5. Install a Code Editor - VS Code.

Step 3: Install a Code Text Editor

NOTE: Step 3

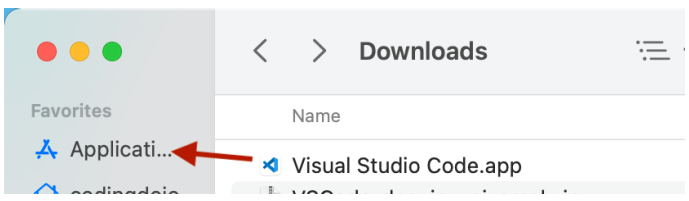
Visual Studio Code

- The final tool to install is a text editor that is designed for programmers.
    - There are several text editors available, but we will be using Visual Studio Code.
- Visual Studio Code (A.K.A "VS Code") is a free editor that is highly customizable and supports many languages.
    - It is maintained by Microsoft and has a robust community of extensions and add-ons.
    - It is very popular and is used by many companies (e.g. Facebook/Meta)
- How will we use VS Code?
    - We could technically run all of our jupyter notebooks using VS Code, but this is not recommended at this point in your education .
        - While VS Code is convenient for quickly opening and working with a repository

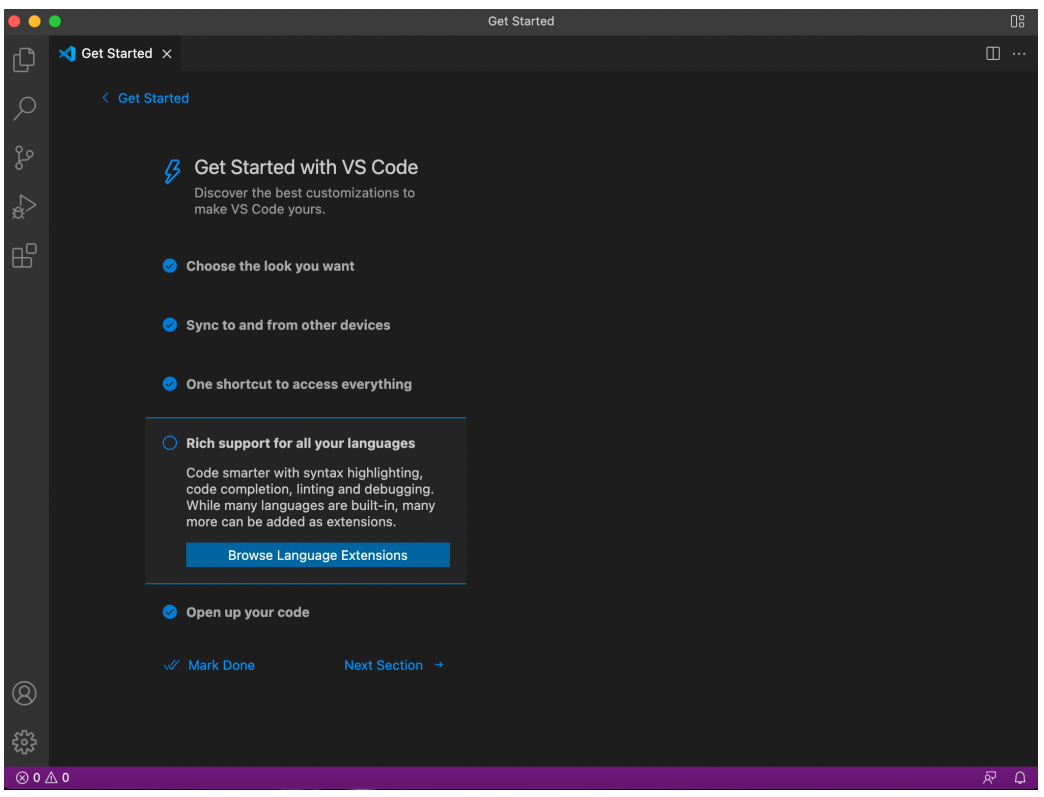
or viewing a notebook, it has some limitations in how notebooks look and some quirks to the interface for notebooks.
        - Instead, we will focus on using jupyter notebook or jupyter lab in the lessons and live class.
            - You are welcome to try VS Code for notebooks, but it is recommended you become comfortable with jupyter first.
- We will use VS Code for editing simple code files or hidden files.
    - We can open and edit the settings file for your terminal (e.g.: " `~/.bash_profile"` .or "~/.zshrc"
    - We will use it to create and store credentials for APIs (Stack 4)
    - We can use VS Code to edit your projects' README files while previewing them in real time!
    - Finally, while beyond the scope of the standard curriculum, we can also use VS Code to store functions in external files that we can use just like pandas, matplotlib,

Install Visual Studio Code

- Go to https://code.visualstudio.com/
    - It should auto-recognize your OS and have a blue Download button with a version for your OS.
    - Click Download to download the installer.
- For Mac Users:
    - Click on the installer to unzip it.
    - Once the Application is unzipped, drag the icon for Visual Studio Code.app to your applications folder on your sidebar in Finder.

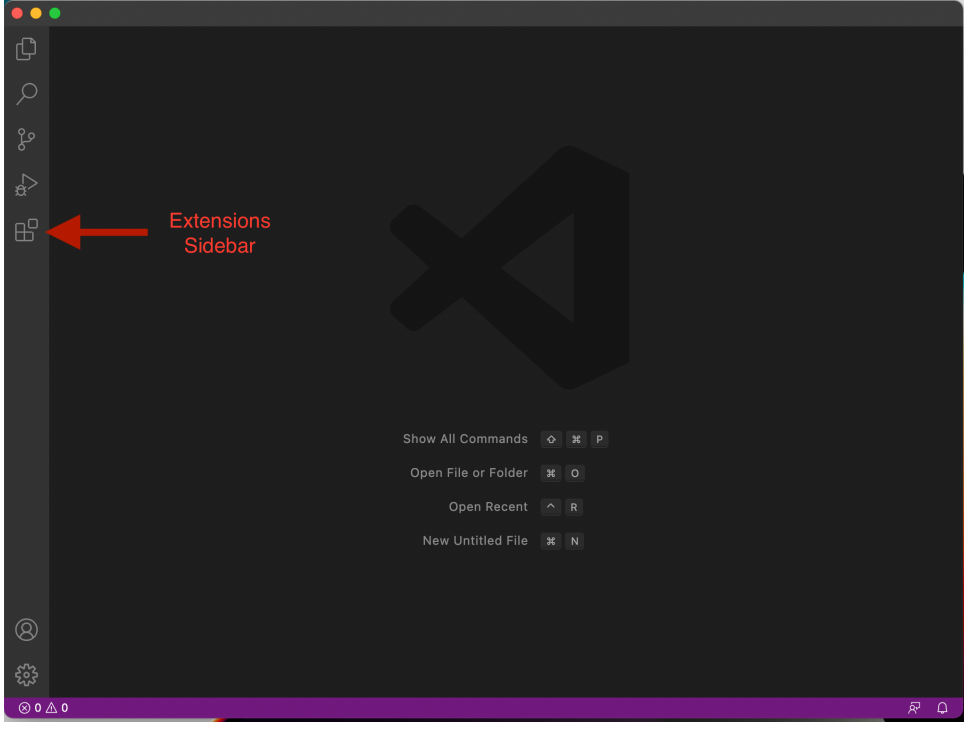


- Once Visual Studio Code installation is completed, open it!
    - Mac Users: check your Applications folder in Finder.

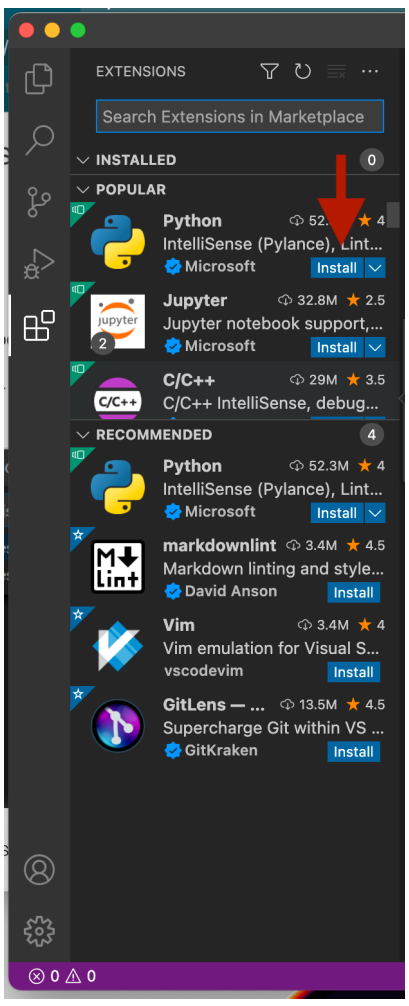


Install Python Extensions

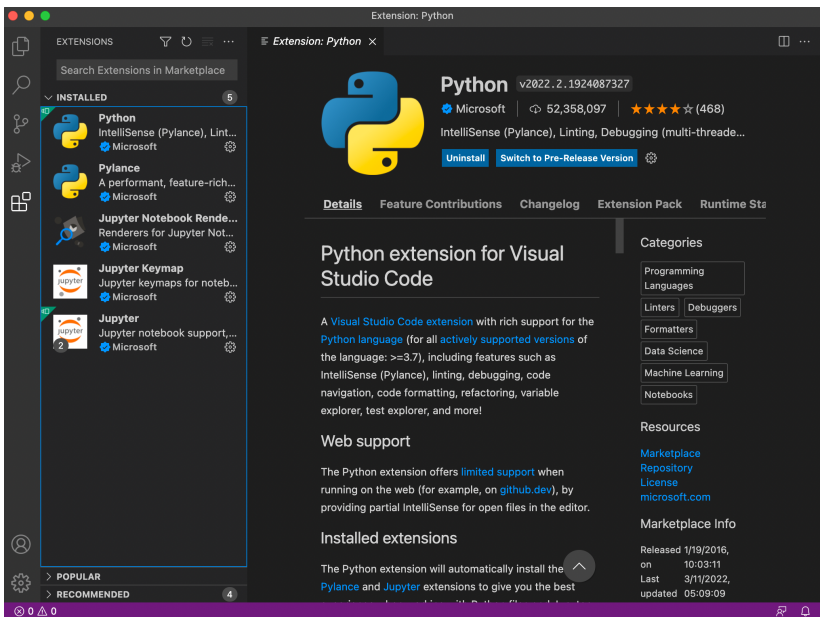
- On the left sidebar, there are several icons for different menus.
- Click on the Extensions sidebar icon (5th down, looks like 4 squares).



- On the Extension sidebar, there will be several sections (INSTALLED/POPULAR /RECOMMENDED).
    - Right now you should have nothing under the INSTALLED menu.
    - You should see "Python" listed under POPULAR.
        - If not, you can enter "Python" in the search box at the top of the sidebar
        - OR you can click on [this link to the extension]() on the extension marketplace website.
    - Click on the "Install" button for the Python extension.

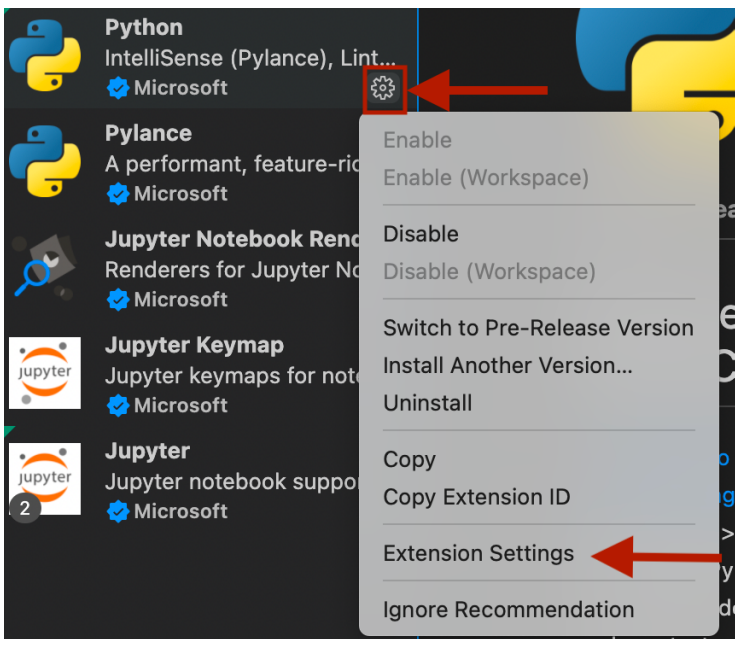


- Note: the Python extension will also install several required extensions. When installation is complete, you should see the following under the "INSTALLED" section:
    - Python, Pylance, Jupyter Notebook renderer, Jupyter, and Jupyter Keymap

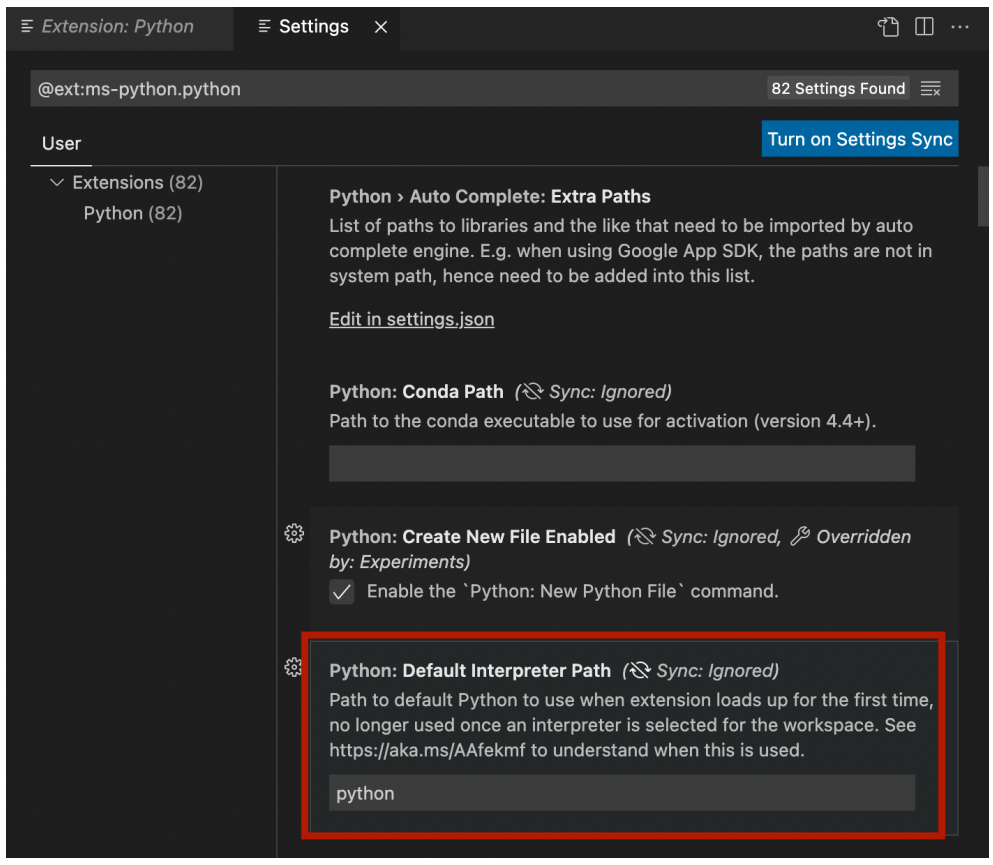


Setting VS Code to use your `dojo-env` as the default Python installation

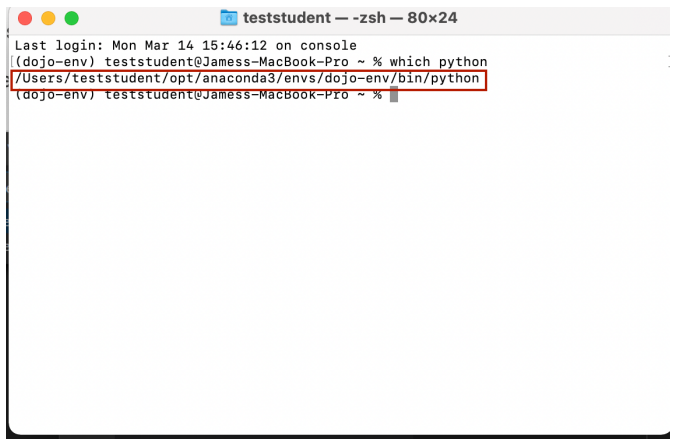
- We must teach the Python extension where to find our `dojo-env` 's version of Python.
- On the extension sidebar, click on the Gear icon for the Python extension and select "Extension Settings"



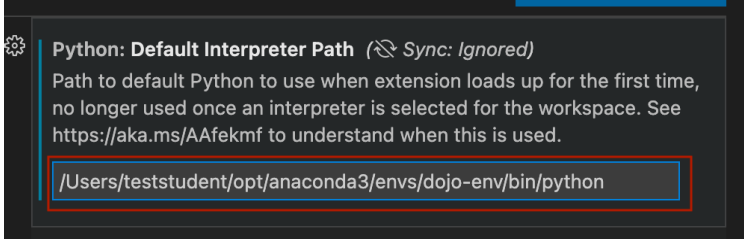
- You should see a new "Settings" pane open in the main window.
    - Take note of the "Default Interpreter Path".
        - It is currently set to just "python".



- We need to change this setting to match the exact filepath for our `dojo-env` 's python.
- In your terminal or GitBash:
    - Make sure your dojo-env is activated
    - Run the command: `which python`
        - It will print out a filepath to your dojo-env.



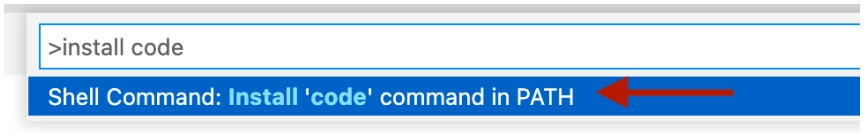
    - Copy and paste that exact file path into the "Default Interpreter Path" field in the Python extension settings.



Mac Users Only: Add the `code` command to your terminal

- We want to be able to type the word "code" in our terminal and have that open up VS Code.
    - Mac Users must run 1 more command from VS Code.

1. Open the Command Palette:
    - Either click on View in the menu bar and select "Command Palette"
    - OR use the keyboard shortcut (`Cmd` + `Shift` + `p`)
2. In the small pop-up window, type "install code" and you should see it auto-suggest the option for "Shell Command: Install 'code' command in PATH".
    - Click on this option.



Test the `code` command

- Open a new terminal or GitBash window.
- Run the command `code` to verify that VS Code opens.
- Note: You can add a specific folder or filename to open, after the word code.
    - To open the current folder `code .`
- If it opens, great!