

MR2020: Python for METOC

Module 0: Getting Started

Summer Quarter 2024

"Why do I need to code? I'm never going to use this in the fleet."

But you will use it during your time at NPS. METOC theses inherently involve data analysis—sometimes of very large amounts of data. Coding—in Python, MATLAB, Fortran, Perl, C++, JavaScript, or whatever language is suitable—is essential for completing a thesis.

Coding is an essential skill in the 21st century. Programming languages are the underpinning that holds today's technologically dependent world together. Perhaps brushing up on this skill a little could come into use after retirement...

"Why Python?"

Python is free and open-source and has a huge community support and software development base. There is almost nothing you can think of that can't be done in Python. While there are many free, open-source languages and compilers, Python is known as an easy-to-read and intuitive language. It is also one of the preferred languages of infrastructure development for machine learning.

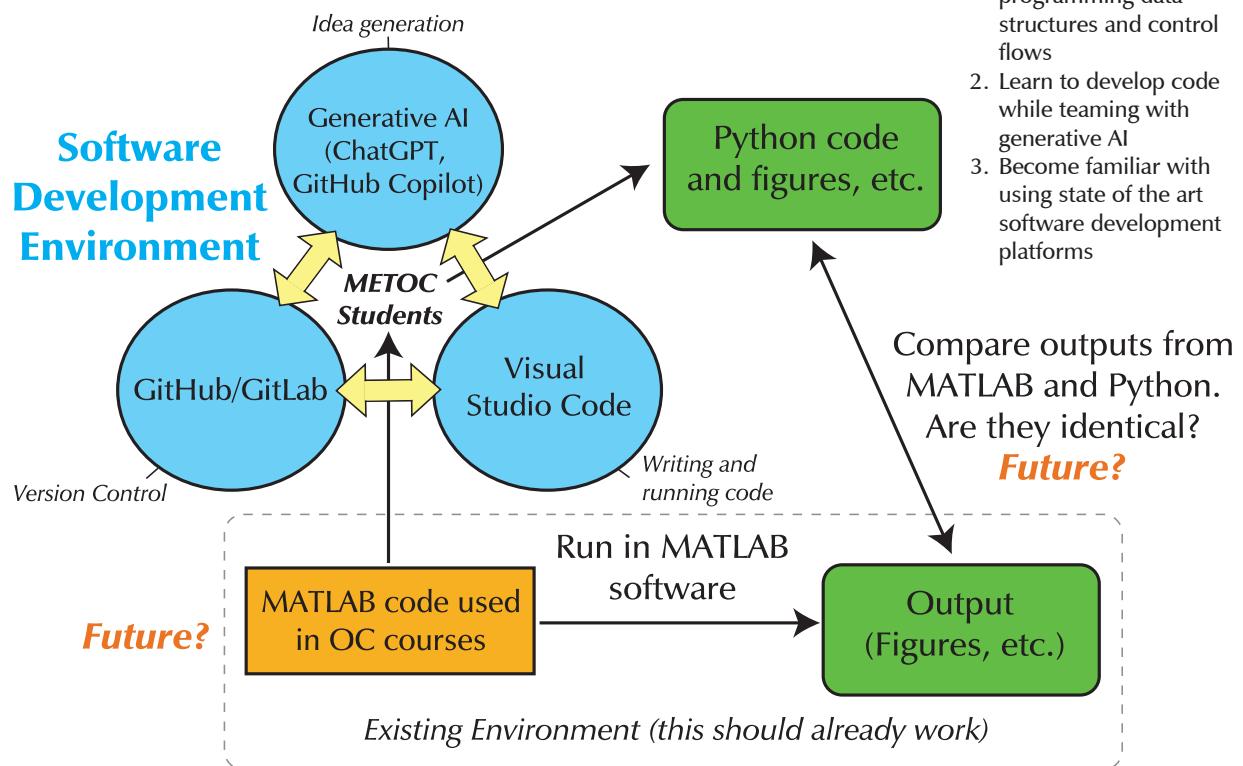
Some languages (e.g., MATLAB, Mathematica) are proprietary. For large users like the Navy, annual licensing can cost a significant amount. However, anything that can be done in MATLAB can also be done in Python. MATLAB's functionality is especially easy to replicate given the recent rise of generative AI such as GPT.

A few basic tools

MR/OC2020: Visual Outline

Course Objectives

1. Learn basics of programming data structures and control flows
2. Learn to develop code while teaming with generative AI
3. Become familiar with using state of the art software development platforms



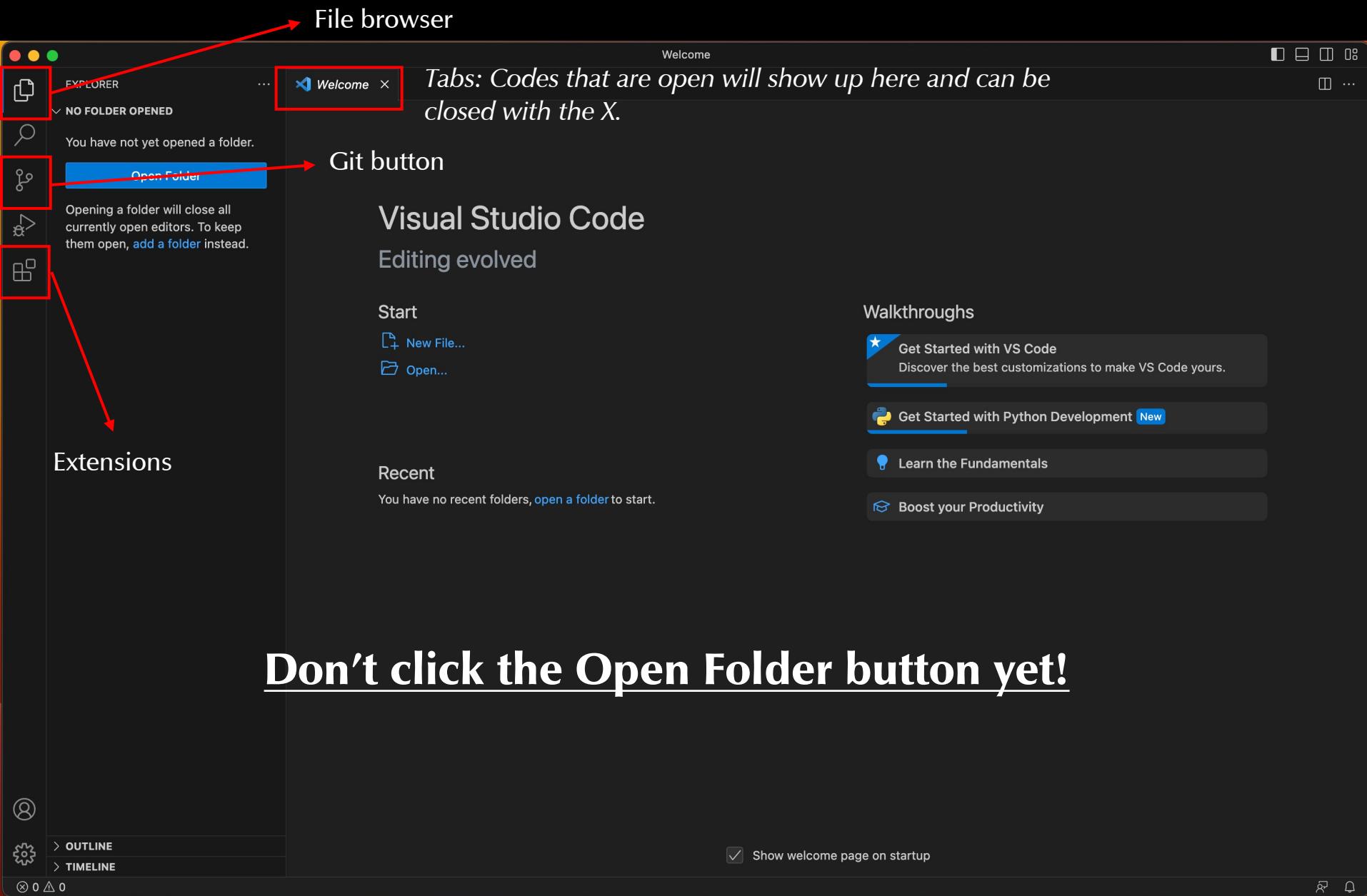
Setup

We need a few tools to get started:

1. Visual Studio Code (configure)
2. Miniconda (download then setup python environment)
3. GitLab repository (through NPS login at gitlab.nps.edu)
4. OpenAI ChatGPT account (online interface)

Please read through each slide carefully and don't just ignore stuff and jump ahead!

Click the magnifying glass in the top right of the Desktop; Type Visual Studio Code and press enter. A Visual Studio Code start screen like the one displayed below will appear.

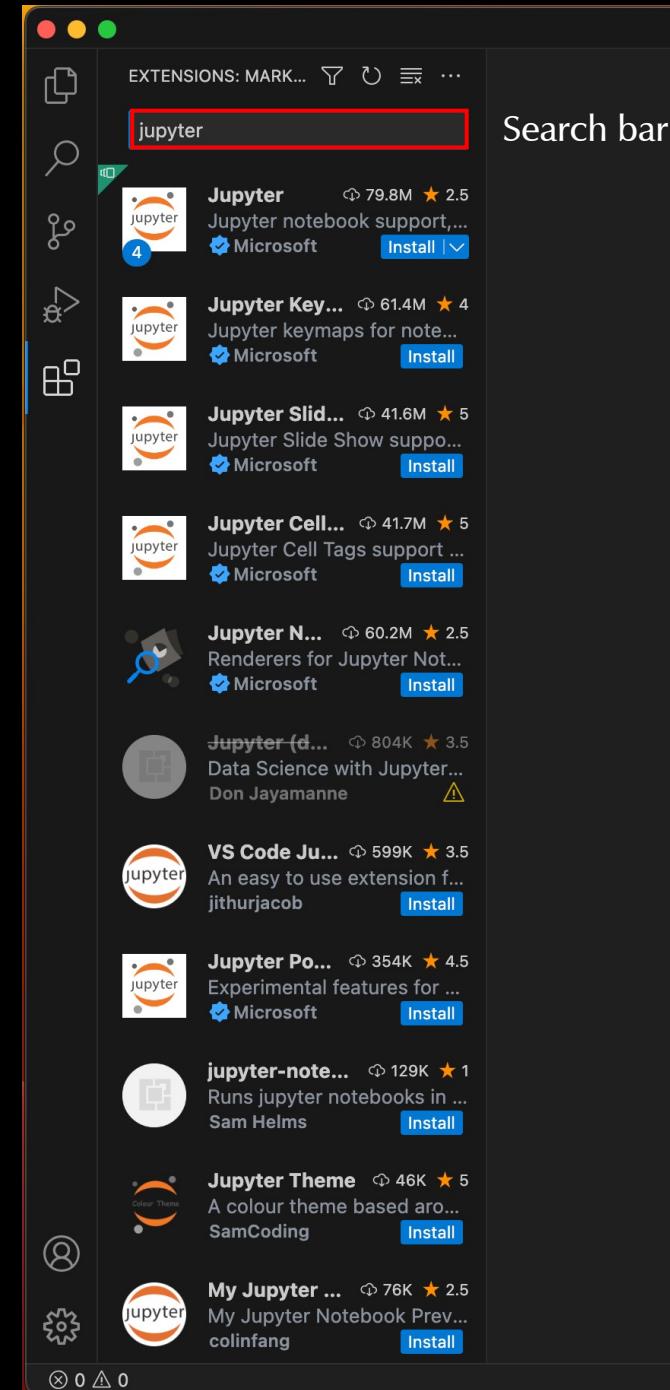


Choose the extensions button on the left-hand side. Type in Python in the search bar near the top. Install the Python extension from Microsoft.

Repeat by searching for jupyter and installing. This is the example shown to the right.

Finally, install the MATLAB extension from Mathworks.

The creator of the extension (e.g., Microsoft, Mathworks) can be seen just left of the install button.



Search bar

Configuring git (This is the hardest part.)

1. Open a new terminal. Type in “bash” and hit enter.
2. Enter “xcode-select --install” and hit enter. Click install and accept the license agreement and any other prompts. The download may take a few minutes. While it’s running skip to the next slide, and we’ll install miniconda while this download and install is underway.

Install miniconda

1. Close the old terminal and open a new terminal.
2. Enter "pwd" (In this class, whenever you see "" for entering something into a Linux command terminal, don't enter the ""). You should see something like "/Users/scott.powell" show up, replacing scott.powell with your name. If for some reason you don't enter "cd /Users/NPSLOGIN", replacing NPSLOGIN with your real login name.
3. Copy and paste the following code into the terminal:

```
mkdir -p ~/miniconda3  
curl https://repo.anaconda.com/miniconda/Miniconda3-latest-MacOSX-arm64.sh -o ~/miniconda3/miniconda.sh  
bash ~/miniconda3/miniconda.sh -b -u -p ~/miniconda3  
rm -rf ~/miniconda3/miniconda.sh
```

4. You should get a message that says "done installation finished." **Check the next slide for what your terminal window might look like***. If you do, copy and paste the following code (**After step 4, you should get something that looks like the screen shown in two slides.**)

```
~/miniconda3/bin/conda init bash  
~/miniconda3/bin/conda init zsh
```

5. Close the terminal window.

You should see something like this between Steps 3 and 4.

```
Terminal — -tcsh — 80x27

Last login: Wed Jul  3 12:47:44 on console
[[METWKC-MAC-006:~] scott.powell% pwd
/Users/scott.powell
[[METWKC-MAC-006:~] scott.powell% mkdir -p ~/miniconda3
curl https://repo.anaconda.com/miniconda/Miniconda3-latest-MacOSX-arm64.sh -o ~/miniconda3/miniconda.sh
bash ~/miniconda3/miniconda.sh -b -u -p ~/miniconda3
rm -rf ~/miniconda3/miniconda.sh[[METWKC-MAC-006:~] scott.powell% curl https://repo.anaconda.com/miniconda/Miniconda3-latest-MacOSX-arm64.sh -o ~/miniconda3/miniconda.sh
% Total    % Received % Xferd  Average Speed   Time     Time      Time  Current
                   Dload  Upload   Total   Spent    Left  Speed
100  105M  100  105M    0      0  64.4M    0  0:00:01  0:00:01 --:--:-- 64.5M
[[METWKC-MAC-006:~] scott.powell% bash ~/miniconda3/miniconda.sh -b -u -p ~/miniconda3
PREFIX=/Users/scott.powell/miniconda3
Unpacking payload ...

Installing base environment...

Preparing transaction: ...working... done
Executing transaction: ...working...
done
installation finished.
[[METWKC-MAC-006:~] scott.powell% rm -rf ~/miniconda3/miniconda.sh
[[METWKC-MAC-006:~] scott.powell%
```

Something like this
will appear after Step
4.

```
Terminal — tcsh — 122x52
rm -rf ~/miniconda3/miniconda.sh[METWKC-MAC-006:~] scott.powell% curl https://repo.anaconda.com/miniconda/Miniconda3-lates
t-MacOSX-arm64.sh -o ~/miniconda3/miniconda.sh
      % Total      % Received % Xferd  Average Speed   Time     Time      Time  Current
                                         Dload  Upload   Total   Spent    Left  Speed
100  105M  100  105M    0     0  64.4M      0  0:00:01  0:00:01 --:--:-- 64.5M
[METWKC-MAC-006:~] scott.powell% bash ~/miniconda3/miniconda.sh -b -u -p ~/miniconda3
PREFIX=/Users/scott.powell/miniconda3
Unpacking payload ...

Installing base environment...

Preparing transaction: ...working... done
Executing transaction: ...working...
done
installation finished.
[METWKC-MAC-006:~] scott.powell% rm -rf ~/miniconda3/miniconda.sh
[METWKC-MAC-006:~] scott.powell% ~/miniconda3/bin/conda init bash
~/miniconda3/bin/conda init zsh
[
no change      /Users/scott.powell/miniconda3/condabin/conda
no change      /Users/scott.powell/miniconda3/bin/conda
no change      /Users/scott.powell/miniconda3/bin/conda-env
no change      /Users/scott.powell/miniconda3/bin/activate
no change      /Users/scott.powell/miniconda3/bin/deactivate
no change      /Users/scott.powell/miniconda3/etc/profile.d/conda.sh
no change      /Users/scott.powell/miniconda3/etc/fish/conf.d/conda.fish
no change      /Users/scott.powell/miniconda3/shell/condabin/Conda.psm1
modified       /Users/scott.powell/miniconda3/shell/condabin/conda-hook.ps1
no change      /Users/scott.powell/miniconda3/lib/python3.12/site-packages/xontrib/conda.xsh
no change      /Users/scott.powell/miniconda3/etc/profile.d/conda.csh
modified       /Users/scott.powell/.bash_profile

==> For changes to take effect, close and re-open your current shell. <==

[METWKC-MAC-006:~] scott.powell% ~/miniconda3/bin/conda init zsh
no change      /Users/scott.powell/miniconda3/condabin/conda
no change      /Users/scott.powell/miniconda3/bin/conda
no change      /Users/scott.powell/miniconda3/bin/conda-env
no change      /Users/scott.powell/miniconda3/bin/activate
no change      /Users/scott.powell/miniconda3/bin/deactivate
no change      /Users/scott.powell/miniconda3/etc/profile.d/conda.sh
no change      /Users/scott.powell/miniconda3/etc/fish/conf.d/conda.fish
no change      /Users/scott.powell/miniconda3/shell/condabin/Conda.psm1
no change      /Users/scott.powell/miniconda3/shell/condabin/conda-hook.ps1
no change      /Users/scott.powell/miniconda3/lib/python3.12/site-packages/xontrib/conda.xsh
no change      /Users/scott.powell/miniconda3/etc/profile.d/conda.csh
modified       /Users/scott.powell/.zshrc

==> For changes to take effect, close and re-open your current shell. <==

[METWKC-MAC-006:~] scott.powell%
[METWKC-MAC-006:~] scott.powell%
```

Create a Python environment

1. Open a new terminal. Enter “zsh”. You should see something like “(base)” appear at the beginning of the command line after this, meaning that miniconda is functioning in the terminal and that the base environment is active. We’re not going to use the base environment though. We’re going to create a new one.
2. Enter the following command to add the package repository conda-forge to your download options:

```
conda config --add channels conda-forge
```

3. Then enter the following to create a Python environment using packages in that repo:

```
conda create -c conda-forge -n mr2020 numpy scipy pandas matplotlib ipython  
metpy xarray ipykernel
```

This will install the Python packages numpy, scipy, pandas, matplotlib, ipython, metpy, and xarray from the channel/package repository called `conda_forge`. Don’t worry about installing everything you may ever need. You can always add on to your environment later. The “`-n mr2020`” means that this Python environment will be called `mr2020`. This will be useful to know soon. A bunch of stuff will pop up. Enter ‘y’, press enter, and wait for the transaction to execute. When done, close out the terminal (Command + q.)

Configuring git (This is the hardest part.)

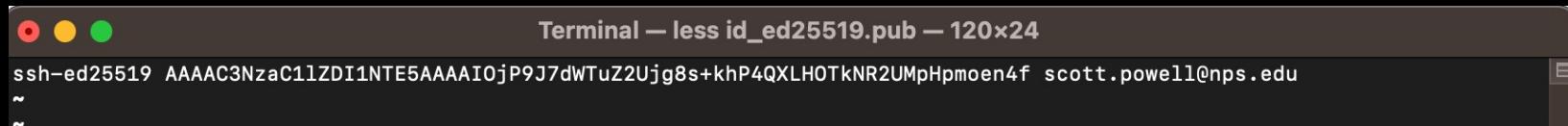
3. Quit Visual Studio Code and reopen it.

4. Click on the git icon. You should now see a button called “Initialize Repository” or “Clone Repository”. This means that your git installation worked!

5. In a terminal (open a new one if needed), enter the following commands (hit enter 3 times after the first line when prompted):

```
ssh-keygen -t ed25519 -C your_NPSname@nps.edu  
cd ~/.ssh  
less id_ed25519.pub
```

After the less command you should see something like the below. Expand the window to get everything on one line if necessary and leave the window open. We'll come back to it soon.

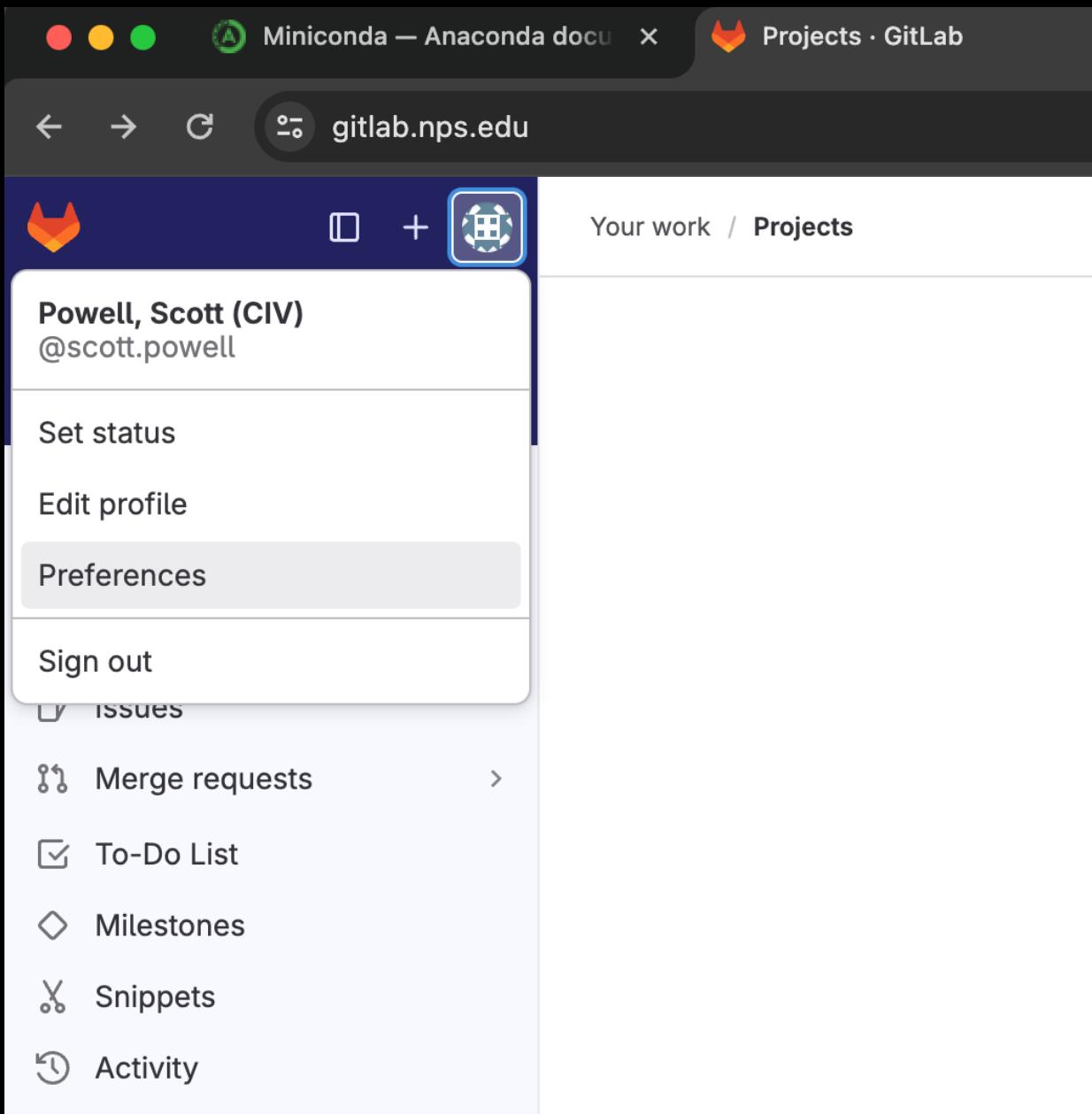


A screenshot of a terminal window titled "Terminal — less id_ed25519.pub — 120x24". The window shows the command "less id_ed25519.pub" being run. The output consists of a long string of characters: "ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAI0jP9J7dWTuZ2Ujg8s+khP4QXLHOTkNR2UMpHpmoen4f scott.powell@nps.edu". Above the terminal window, there are three small colored circles (red, yellow, green) and a horizontal line.

6. Open a browser window (e.g., Google Chrome, Safari). Go to gitlab.nps.edu and Click the “NPS Account” button. Enter your login credentials and go through the 2FA prompts using Microsoft Authenticator.

The screenshot shows a web browser window titled "Sign in - GitLab" with the URL "gitlab.nps.edu/users/sign_in". The page has a light gray background with a central orange cat logo. At the top left, it says "NPS GitLab". Below the logo, there are two sections: "NPS users" and "External users". The "NPS users" section contains a bullet point: "Sign in using the "NPS account" button below." The "External users" section contains a bullet point: "Sign in using your email and password." Below these sections is a paragraph about managing Git repositories. Further down, there is a bolded statement: "By logging in, you agree to the terms described within the [DoD Acceptable Use Policy](#)". To the right of the text fields, there is a "Forgot your password?" link. Below the text fields are "Remember me" and "Sign in" buttons. A note below the "Sign in" button states: "By signing in you accept the [Terms of Use](#) and acknowledge the [Privacy Statement](#) and [Cookie Policy](#)". Below this, there is a "or sign in with" section featuring a red-bordered "NPS account" button with a blue square icon, and a "Remember me" checkbox. At the bottom of the page, there are links for "About GitLab" and "Community forum", and a language selector for "English".

7. In the browser, enter the circular avatar icon in Gitlab (see screenshot), select Preferences. After a new page loads, select “SSH keys” from the new menu bar on the left.



8. Click "Add a key".

The screenshot shows the 'User Settings / SSH Keys' page on gitlab.nps.edu. On the left, there's a sidebar with 'User settings' (Profile, Account, Applications). The main area has a search bar and a section titled 'SSH Keys' with the sub-section 'Your SSH keys'. At the bottom right of this section, there is a blue button labeled 'Add new key' which is highlighted with a red box.

9. Copy and paste the entire SSH key from the open terminal window into the Key box as shown to the right.

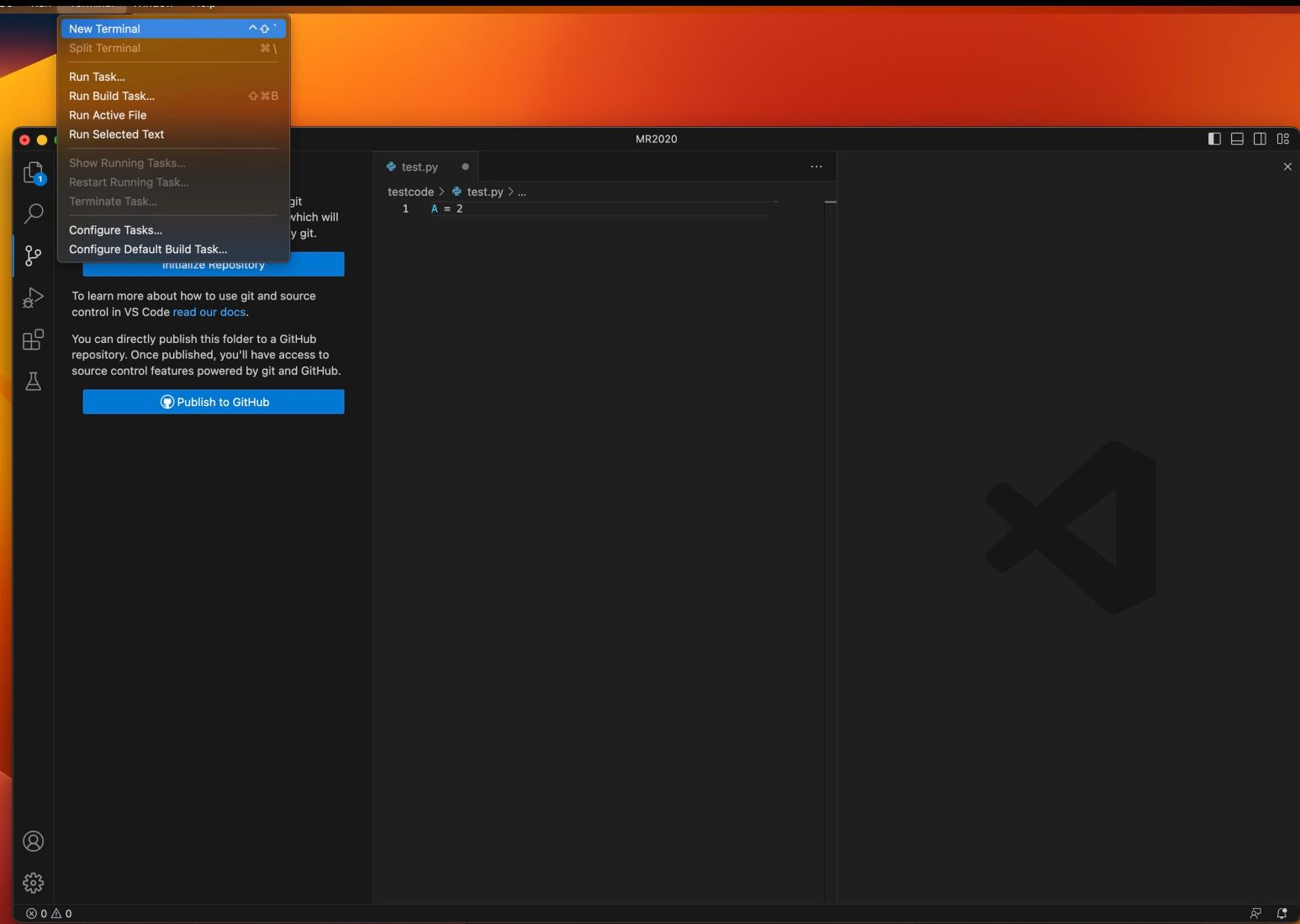
For Title, enter the name of the machine you are working on. It will be the part in the red box in the terminal inset in bottom right of image.

Then, click Add Key.

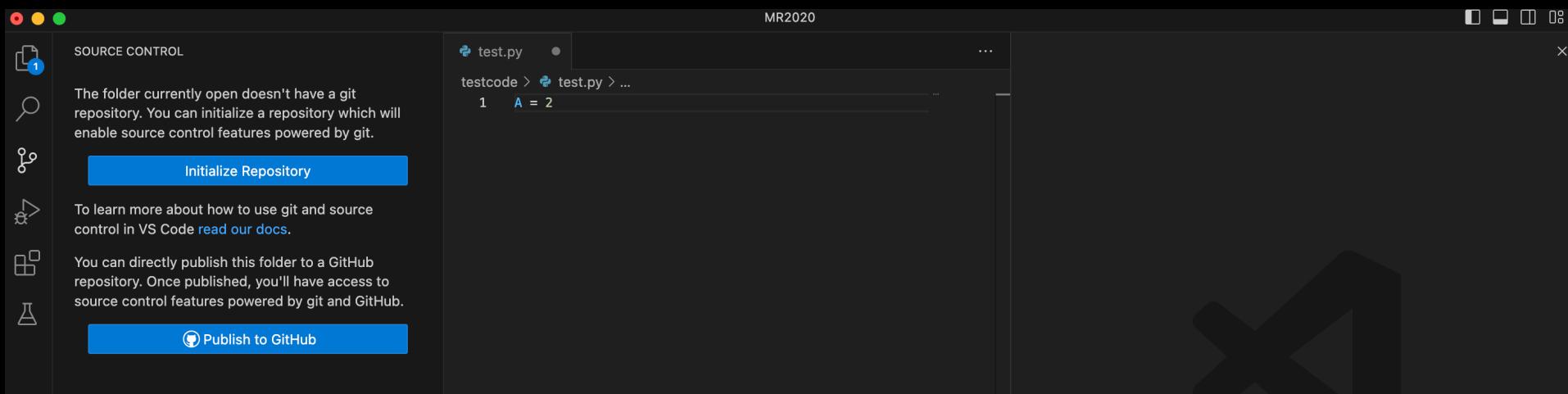
The screenshot shows the 'Add an SSH key' form. The 'Key' field contains the copied SSH key: `ssh-ed25519 AAAAC3NzaC1I2D1NT5AAAAIOjP9J7dWTuZ2UjgBs+khP4QXLHOTkNR2UMpHpmoen4f scott.powell@nps.edu`. Below the form is a terminal window showing the command and its output. The title 'metwkc-mac-006' is highlighted with a red box in the terminal's title bar.

```
Enter same passphrase again:
Your identity has been saved in '/Users/scott.powell/.ssh/id_ed25519'.
Your public key has been saved in '/Users/scott.powell/.ssh/id_ed25519.pub'.
The key's fingerprint is:
SHA256:FWV/DXRxFyK4MeR0v9r1sSWu1mCT4elby3dd2z quo scott.powell@nps.edu
The key's randomart image is:
+--[ED25519 200]--+
|          .o|
|         ..o+ |
|        .+=#B+|
|       .+=#X+|
|      .+=X+|
|     .+=O+|
|    .+=+|
|   .+| |
|  .+| |
| +| |
+---[SHA256]---+
[METWKC-MAC-006:Developer/CommandLineTools/usr] scott.powell% cd .ssh
[METWKC-MAC-006:Developer/CommandLineTools/usr] scott.powell% cd -./.ssh
[METWKC-MAC-006:~/.ssh] scott.powell% ls
id_ed25519      id_ed25519.pub
[METWKC-MAC-006:~/.ssh] scott.powell% less id_ed25519.pub
[METWKC-MAC-006:~/.ssh] scott.powell%
```

10. In Visual Studio Code, look at the menu bar at the top and open a new terminal.



11. In the VSC terminal window, enter ssh -T git@gitlab.nps.edu. Type and enter “yes” when prompted. You should get a message that says, “Welcome to Gitlab, @username!”



Make sure that TERMINAL is white with a blue line beneath it.

A screenshot of the VS Code interface focusing on the bottom navigation bar. The "TERMINAL" tab is highlighted with a red box. The terminal window shows a command-line session:

```
[METWKC-MAC-006:~/Desktop/MR2020] scott.powell% ssh -T git@gitlab.nps.edu
You are accessing a U.S. Government (USG) Information System (IS) that is provided for USG-authorized use only. By using this IS (which includes any device attached to this IS), you consent to the following conditions: -The USG routinely intercepts and monitors communications on this IS for purposes including, but not limited to, penetration testing, COMSEC monitoring, network operations and defense, personnel misconduct (PM), law enforcement (LE), and counterintelligence (CI) investigations. -At any time, the USG may inspect and seize data stored on this IS. -Communications using, or data stored on, this IS are not private, are subject to routine monitoring, interception, and search, and may be disclosed or used for any USG-authorized purpose. -This IS includes security measures (e.g., authentication and access controls) to protect USG interests—not for your personal benefit or privacy. -Notwithstanding the above, using this IS does not constitute consent to PM, LE or CI investigative searching or monitoring of the content of privileged communications, or work product, related to personal representation or services by attorneys, psychotherapists, or clergy, and their assistants. Such communications and work product are private and confidential. See User Agreement for details.
Welcome to GitLab, @scott.powell!
[METWKC-MAC-006:~/Desktop/MR2020] scott.powell%
```

The terminal window has a white background with a blue line underneath the tabs.

12. Back in the browser window with Gitlab, hit the fox in the top left. In the top right of the screen that displays, there is a blue button that says, “New Project”. Click it. Follow the screenshots to create a repository called “mr2020”. Under Project name, enter “mr2020”. Under “Project URL”, choose your user name from the drop-down menu. Under “Project slug”, enter “mr2020”. Then, click the blue “Create Project” button.



Create blank project
Create a blank project to store your files, plan your work, and collaborate on code, among other things.

Project name
 Must start with a lowercase or uppercase letter, digit, emoji, or underscore. Can also contain dots, pluses, dashes, or spaces.

Project URL / **Project slug**
 scott.powell

Pick a group or namespace where you want to create this project.

Visibility Level [?](#)
  Private
Project access must be granted explicitly to each user. If this project is part of a group, access is granted to members of the group.
  Internal
The project can be accessed by any logged in user except external users.
  Public
The project can be accessed without any authentication.

Project Configuration

Initialize repository with a README
Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.

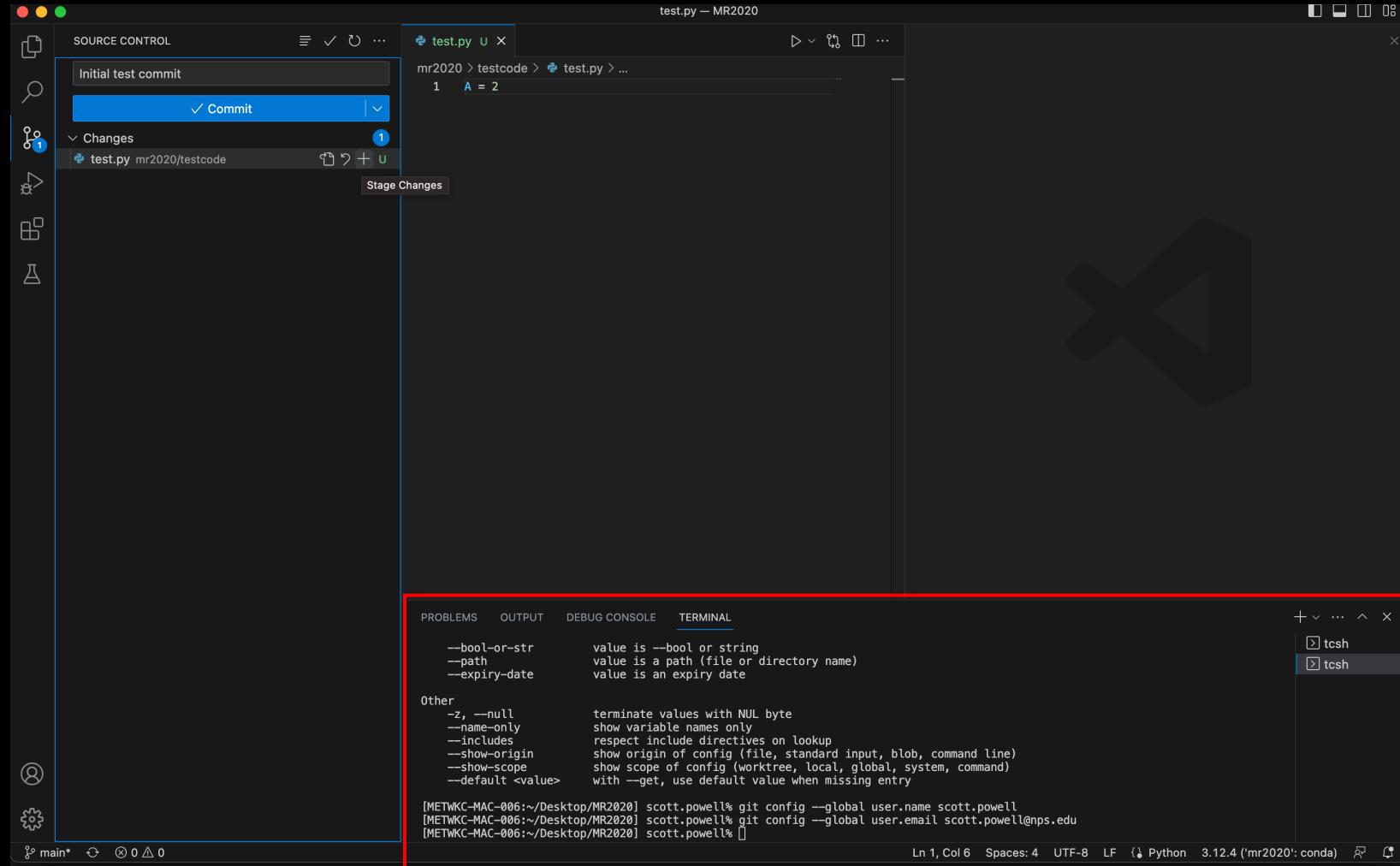
Enable Static Application Security Testing (SAST)
Analyze your source code for known security vulnerabilities. [Learn more](#).

Create project **Cancel**

13. Back in Visual Studio Code, in the terminal enter the following (replacing scott.powell with your NPS login name):

```
git config --global user.name scott.powell
```

```
git config --global user.email scott.powell@nps.edu
```



14. We need to clone the repo. Right-click on your Desktop and create a new folder. Call it MR2020. In the browser, click the blue “Code” button in the top right, and copy the text under “Clone with SSH”.

The screenshot shows a GitLab repository named 'mr2020'. The 'Code' dropdown menu is open, displaying two cloning options: 'Clone with SSH' (git@gitlab.nps.edu:scott.powell/) and 'Clone with HTTPS' (https://gitlab.nps.edu/scott.powell). The 'Clone with HTTPS' URL is highlighted and has a 'Copy URL' button. The 'Project information' sidebar on the right provides details about the repository, including 1 commit, 1 branch, 0 tags, and 3 KiB of project storage. The 'Getting started' section contains a list of recommended tools and links to add a LICENSE, CHANGELOG, CONTRIBUTING, Kubernetes cluster, Wiki, and Configure Integrations. The 'Add your files' section includes options to create or upload files or add files via command line, with a terminal window showing the git commands to clone the repository.

Project 'mr2020' was successfully created.

M mr2020

main / mr2020 / +

Initial commit
Powell, Scott (CIV) authored just now

Name	Last commit
README.md	Initial commit

README.md

mr2020

Getting started

To make it easy for you to get started with GitLab, here's a list of recommended ne...

Already a pro? Just edit this README.md and make it your own. Want to make it ea...

Add your files

Create or upload files
 Add files using the command line or push an existing Git repository with the following command:

```
cd existing_repo
git remote add origin https://gitlab.nps.edu/scott.powell/mr2020.git
git branch -M main
git push -uf origin main
```

History Find file Edit Code

Clone with SSH
git@gitlab.nps.edu:scott.powell/

Clone with HTTPS
https://gitlab.nps.edu/scott.powell/

Open in your IDE
Visual Studio Code (SSH)
Visual Studio Code (HTTPS)
IntelliJ IDEA (SSH)
IntelliJ IDEA (HTTPS)

Download source code
zip
tar.gz
tar.bz2
tar

Project information

- o- 1 Commit
- 1 Branch
- 0 Tags
- 3 KiB Project Storage

README

Auto DevOps enabled

+ Add LICENSE
+ Add CHANGELOG
+ Add CONTRIBUTING
+ Add Kubernetes cluster
+ Add Wiki
+ Configure Integrations

Created on
July 03, 2024

15. Back in VSC, in the terminal, type and enter the following two lines one at a time:

```
cd  
cd Desktop  
mkdir MR2020  
cd MR2020
```

Then, in the terminal, type in git clone and paste the command you copied from the browser into the terminal as seen below and press enter. (See below.) Your repo is now set up and linked to VSC!

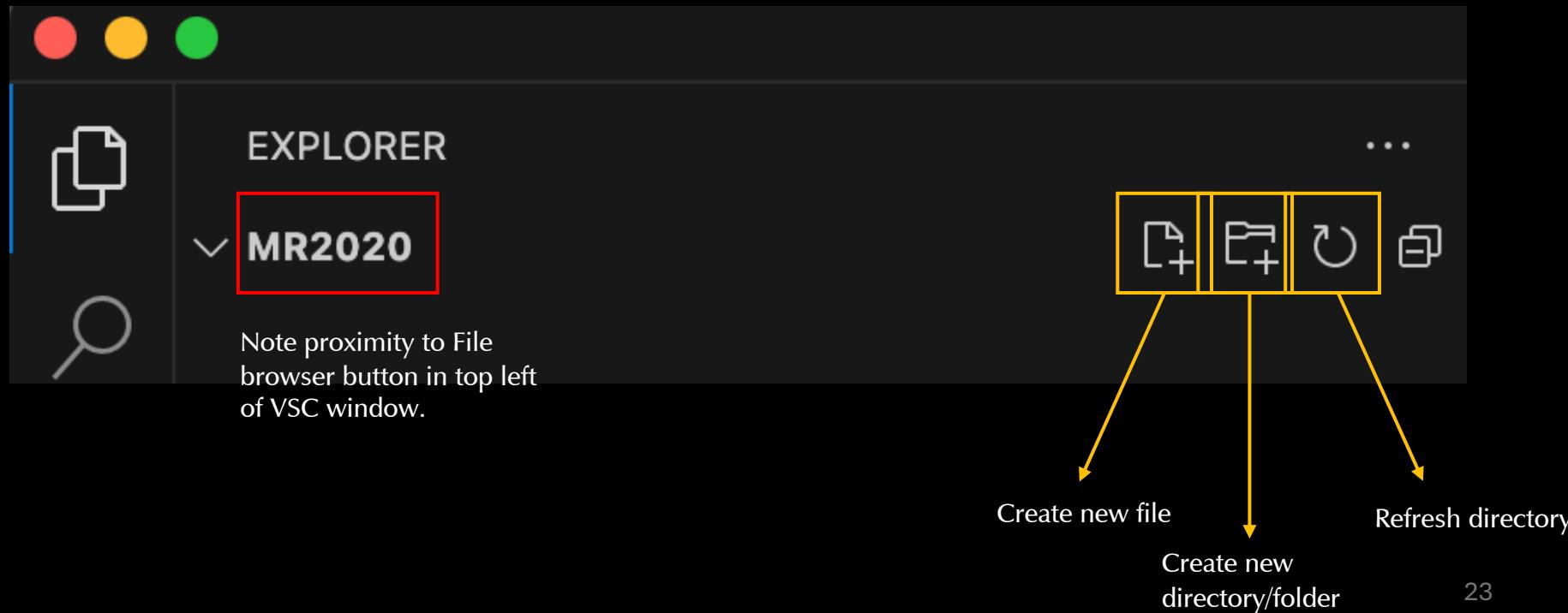
```
ata stored on this IS. -Communications using, or data stored on, this IS are not private, are subject to  
or any USG-authorized purpose. -This IS includes security measures (e.g., authentication and access controls).  
Notwithstanding the above, using this IS does not constitute consent to PM, LE or CI investigative search and seizure.  
related to personal representation or services by attorneys, psychotherapists, or clergy, and their associates.  
See User Agreement for details.  
Welcome to GitLab, @scott.powell!  
[METWKC-MAC-006:~/Desktop/MR2020] scott.powell% cd Desktop  
Desktop: No such file or directory.  
[METWKC-MAC-006:~/Desktop/MR2020] scott.powell% ls  
testcode  
[METWKC-MAC-006:~/Desktop/MR2020] scott.powell% cd  
[METWKC-MAC-006:~] scott.powell% cd Desktop/MR2020  
[METWKC-MAC-006:~/Desktop/MR2020] scott.powell% ls  
testcode  
[METWKC-MAC-006:~/Desktop/MR2020] scott.powell% git clone git@gitlab.nps.edu:scott.powell/mr2020.git
```

⑧ 0 ⚠ 0

Updating your repo and pushing to GitLab

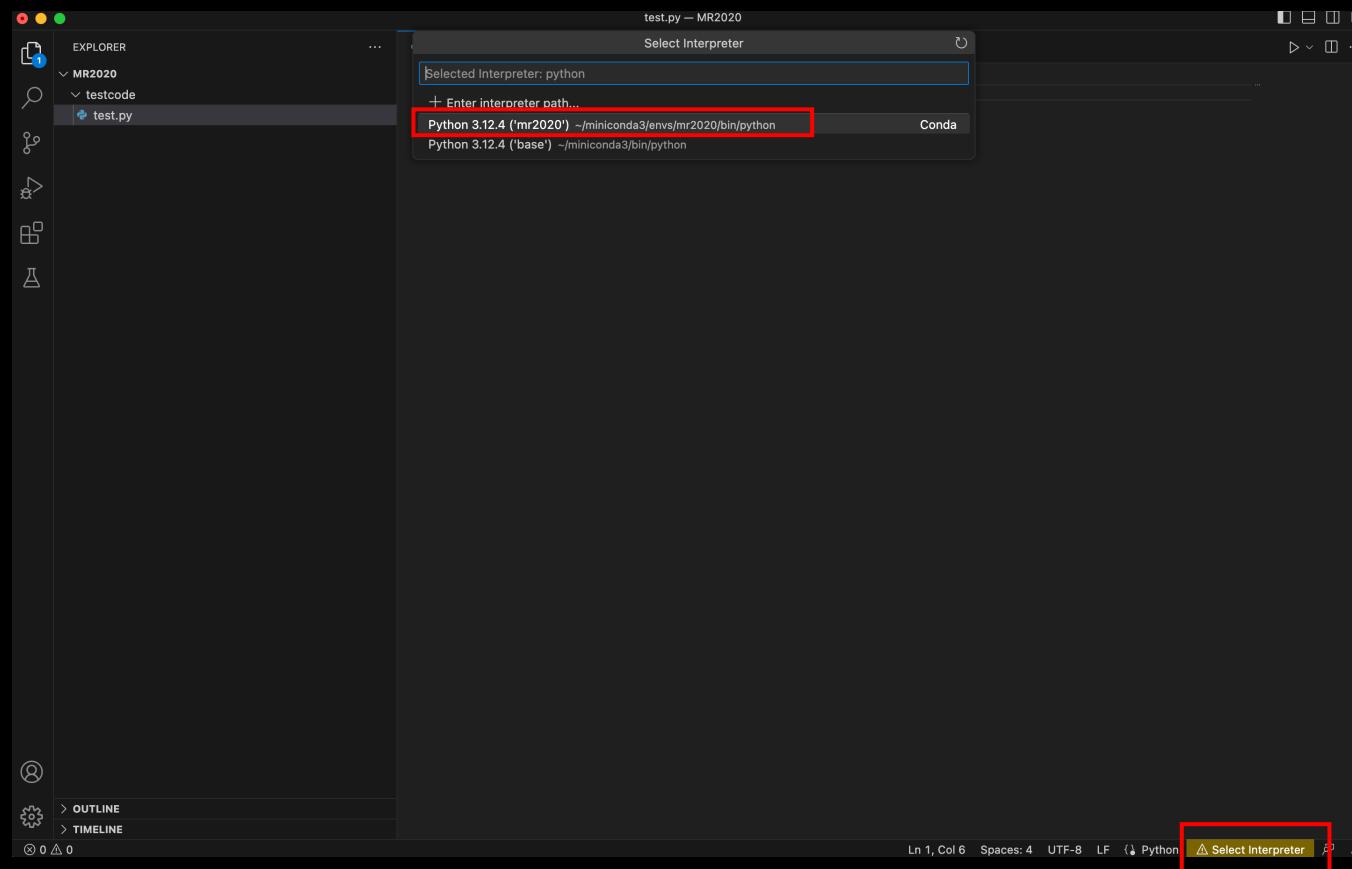
16. Next, let's put something meaningful into the repository you just created. Click the File browser button in VSC. Click the blue "Open Folder" button. Choose "Desktop -> MR2020 -> mr2020". If prompted, trust the author (yourself). After a few seconds, you should see a README file populate the file browser since your repo was initialized with one.

17. If you hover over the area near the "MR2020" text in the upper right (see red box on image below), you'll see some icons pop up nearby (in the orange box). See the image below for details about these icons.



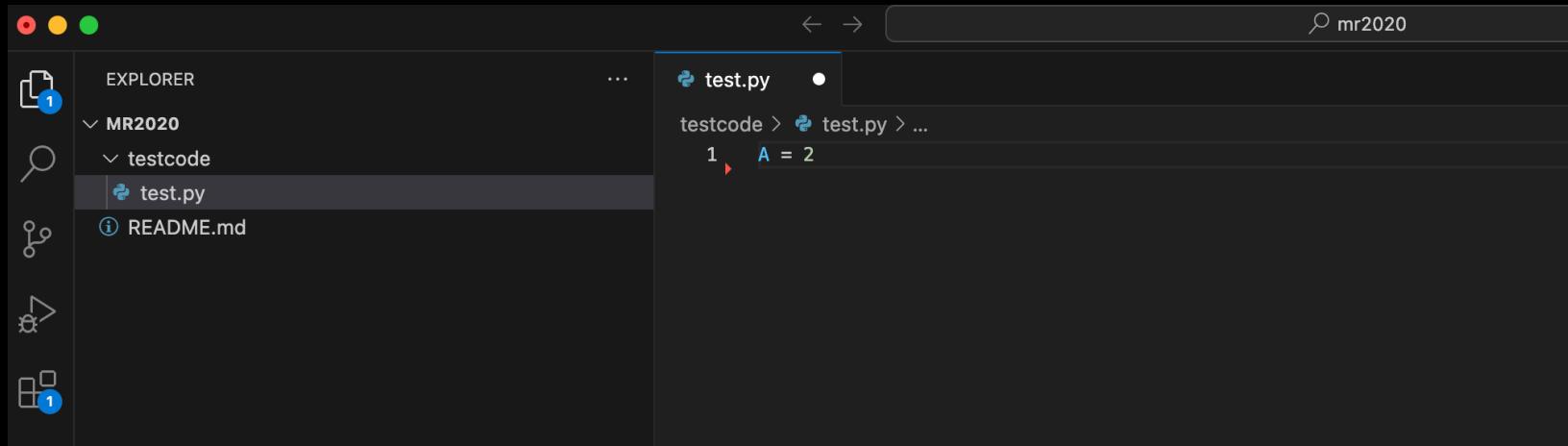
18. Click the new directory button and create a directory called “testcode”. Click on that folder, then create a new file and call it “test.py”. Click on test.py to cause a tab to open to the right.

19. In the bottom right, hit “Select Interpreter”. Select mr2020 from the dropdown menu. This is the Python environment you created earlier.



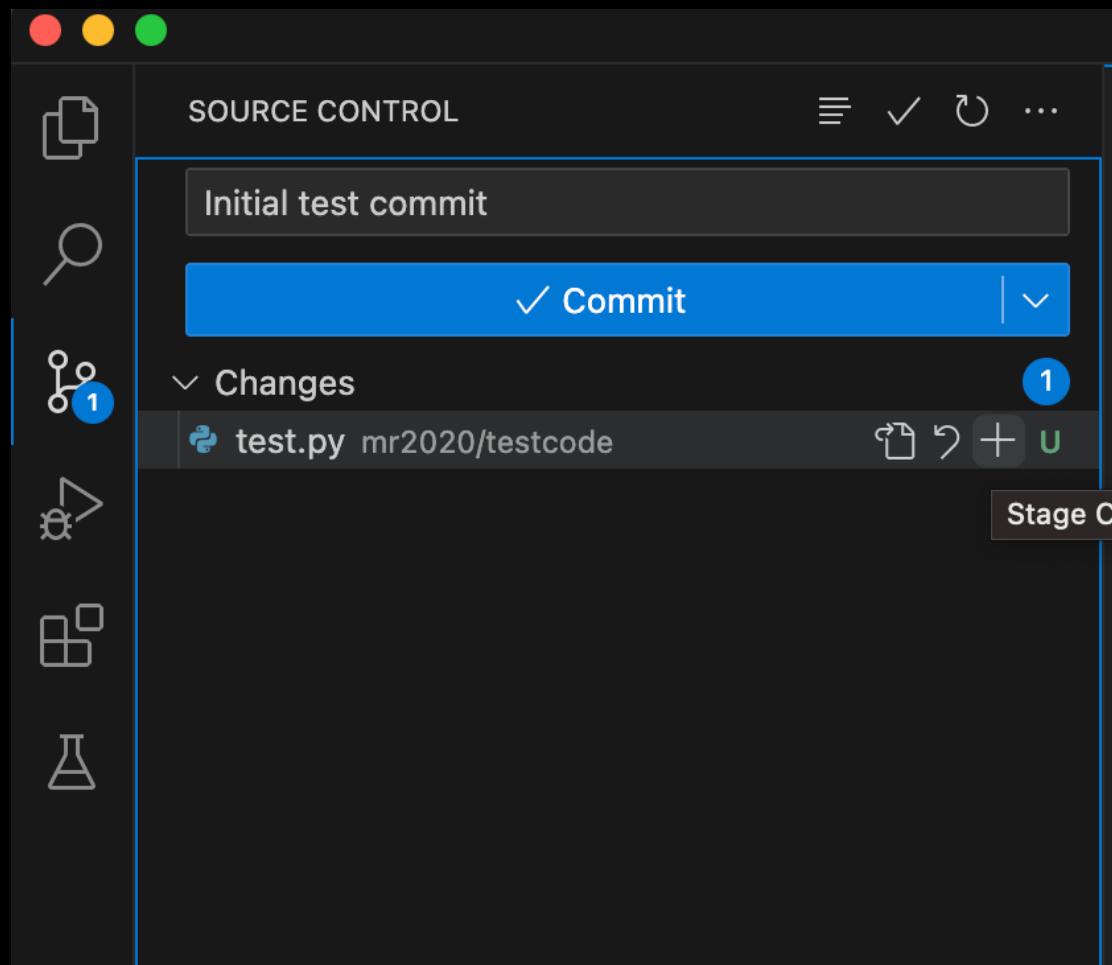
20. Enter the following code for test.py:

A = 2



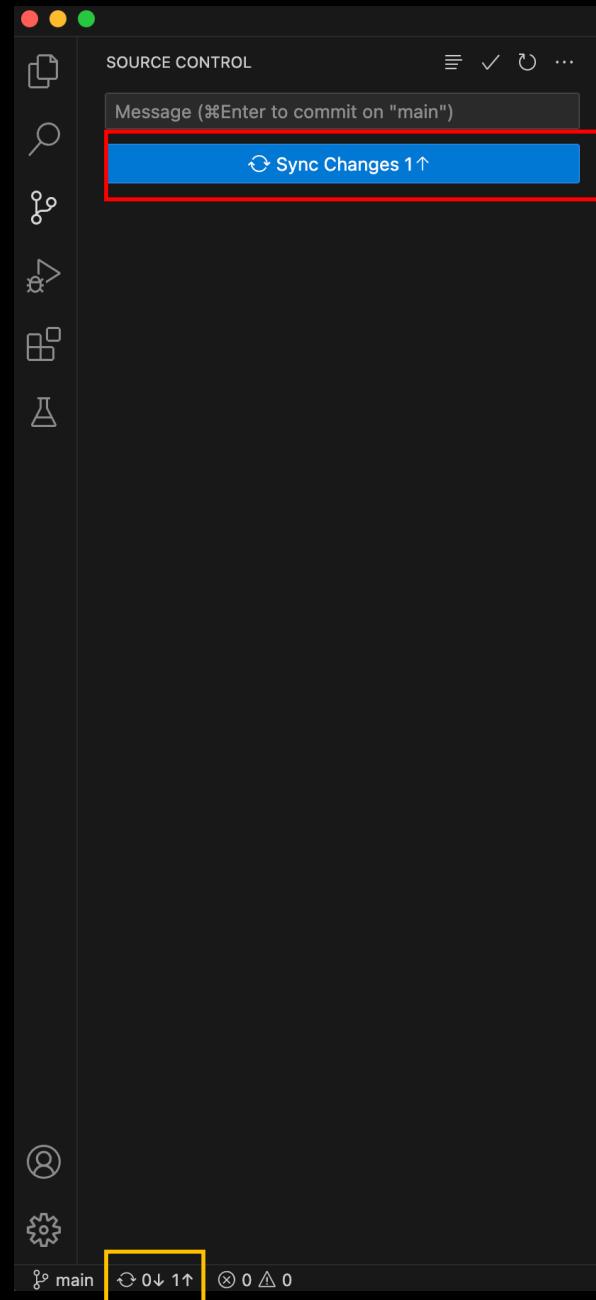
After you do so, a little blue bubble with a number in it should appear on the git icon on the left menu bar of VSC. (There are blue bubbles for other things shown in the image above but not for the git icon.) The number in the git icon bubble indicates how many files have been changed since the last push to GitLab, i.e., how many files have progress that has not been backed up.

21. Click the git button. You will see a list of files that need to be pushed to GitLab for backup. Click the plus next to the file you want to push. It will move to a tab labeled “Staged Changes”. Enter a message in the box above the blue Commit button (such as “Initial test commit”). Then click Commit.



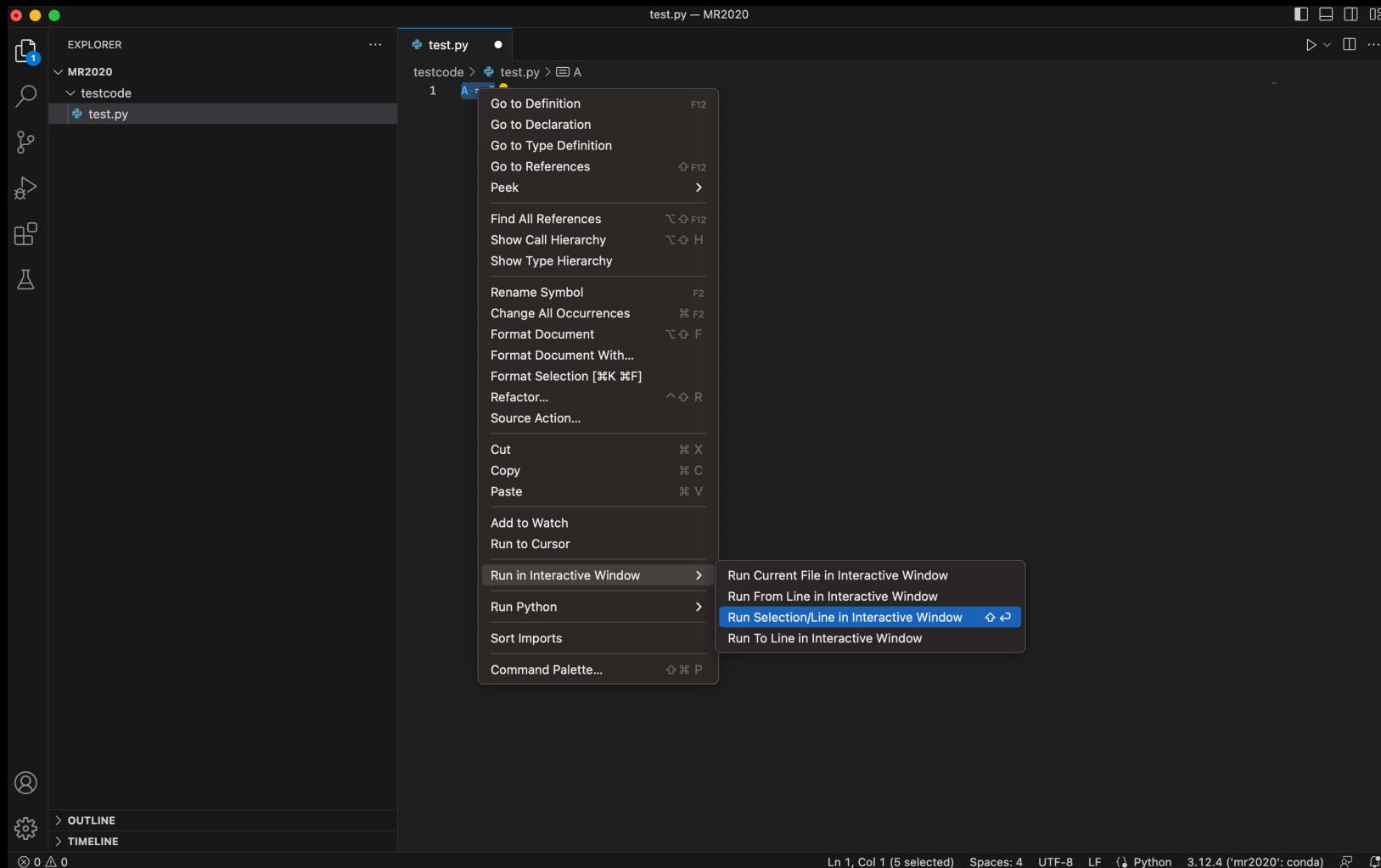
22. You haven't finished the push yet! Make sure you click the Sync button(red box in the image below) after the Commit button goes away. If no sync button appears, look in the bottom left of the VSC window and click the up arrow with a number next to it (orange box in image below).

Refresh the browser window in Gitlab, and you should see your new directory appear. Inside the directory, if you click it, you will see the test.py code. You may now close out the browser window. You should not need to login to Gitlab via the browser anymore.

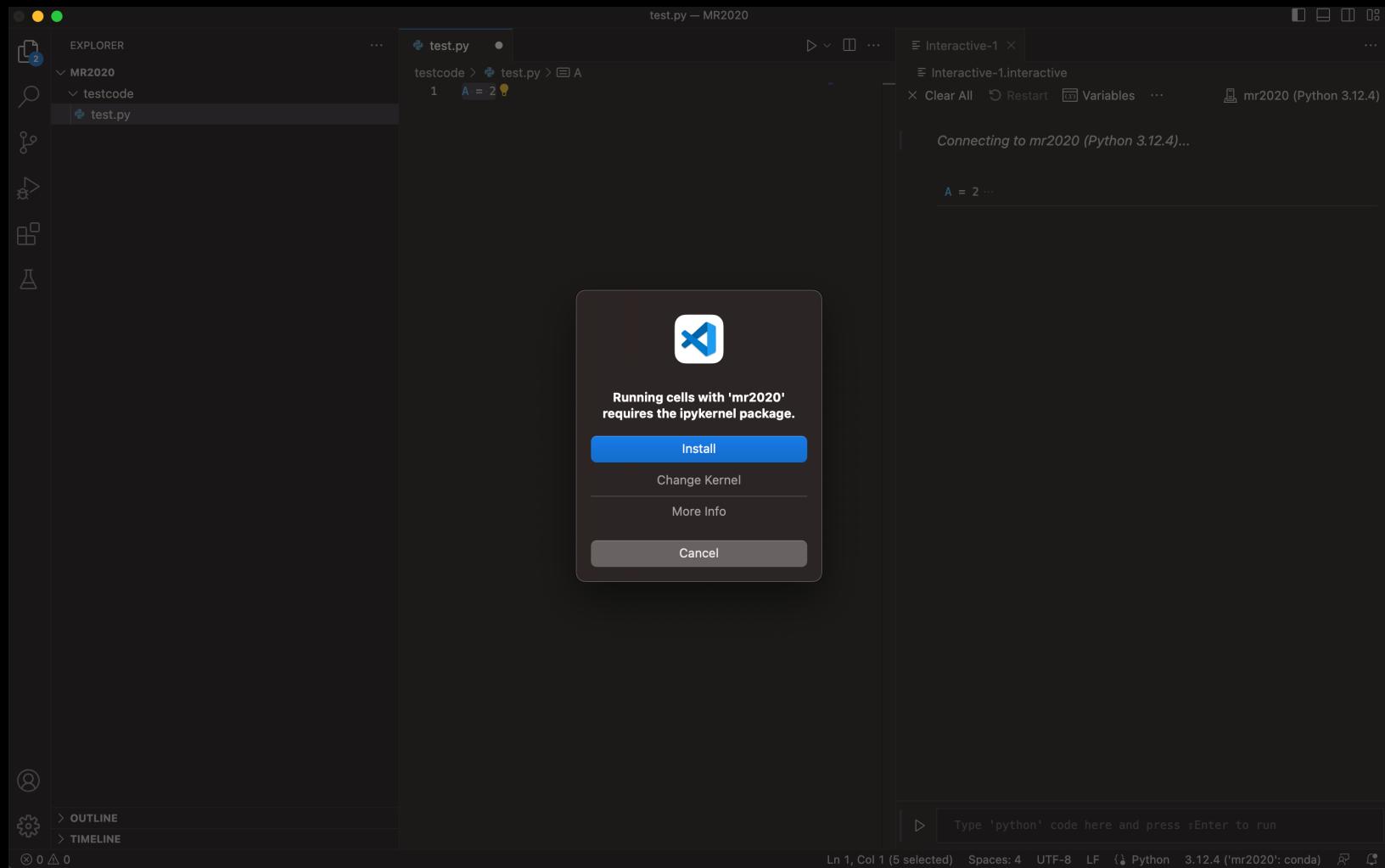


Running code in an interactive window

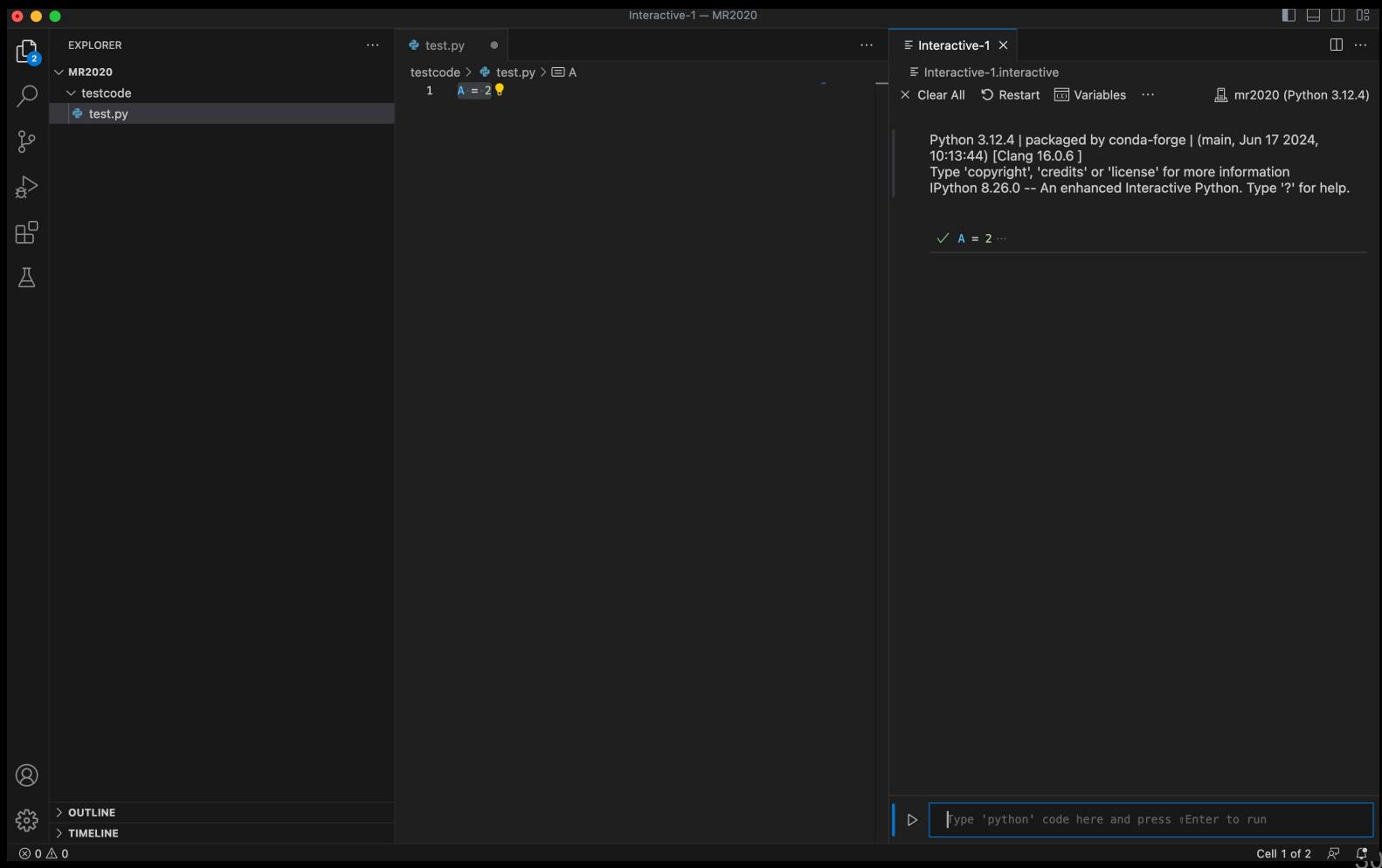
In VSC, highlight the code `A = 2` in `test.py`. Select “Run in Interactive Window” -> “Run Selection/Line in Interactive Window” as shown below.



Click install to install ipykernel into the environment if prompted. If not, continue. If asked if you want to periodically run “git fetch”, click no.



You should get something like the screen below, with an interactive window appearing to the right of the code. You can enter or copy/paste other snippets of code into the box at the bottom of the interactive window and either pressure Command + Enter or press the play button to execute the code. You can close the interactive window by pressing the x on the tab near the top of the screen that says “Interactive-1”.



Get a ChatGPT account

Go to chatgpt.com in a browser. Use your existing login, or if you don't have one, sign up for account by clicking the green button in the bottom left of the screen.

The screenshot shows the ChatGPT 3.5 web interface. At the top left are 'New chat' and 'ChatGPT 3.5' buttons. The main area features a large input field with a placeholder 'How can I help you today?'. Below it are several suggested prompts:

- Create a recipe using ingredients from my kitchen
- Help me pick an outfit that will look good on camera
- Write a text inviting my neighbors to a barbecue
- Make me a personal webpage after asking me three questions

At the bottom left are 'Sign up' and 'Log in' buttons. A message bar at the bottom center says 'Message ChatGPT' with an upward arrow icon. A small note at the bottom states: 'By messaging ChatGPT, you agree to our Terms and have read our Privacy Policy.'