

Recitation 0: Demo Github Classroom

Please follow the directions below EXACTLY to complete this recitation successfully.

1. If you have not done so, head over to <https://github.com> and sign up for an account. Please **USE YOUR DESU EMAIL ACCOUNT** when signing up for the account.
2. At this point, you should have a recent version of Python installed.
3. You will next need to accept the assignment in Github classroom and link it to your Github account. Click on <https://classroom.github.com/a/fqZaNIBP>.
4. Find your name/email in the list and select it. IT IS IMPORTANT THAT YOU SELECT YOUR NAME FROM THE LIST!!!!
5. Accept the assignment and wait a minute before refreshing the page. You should now have a link to your own private repository that contains the assignment.
6. Click the link shown on the page. Should you need the link in the future, simply click on the exercise link provided above to come back to this page.



A dark grey header bar with the GitHub Classroom logo on the left and "GitHub Edu" on the right.



You're ready to go!

You accepted the assignment, **Demo Github Classroom**.

Your assignment repository has been created:

 <https://github.com/desu-pemacs/demo-github-classroom-rasamny>

We've configured the repository associated with this assignment ([update](#)).

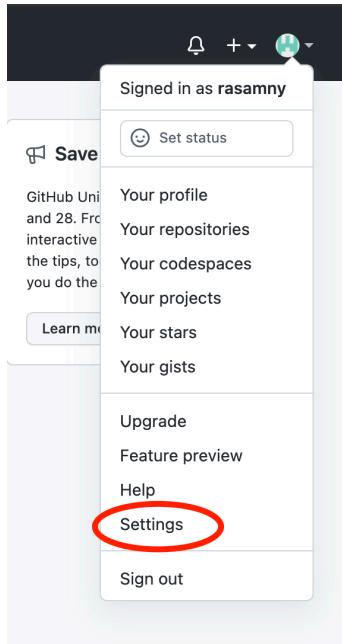
 Your assignment is due by **Sep 22, 2021, 23:59 EDT**

7. You should now see your private repository with a green Code button on the top right as

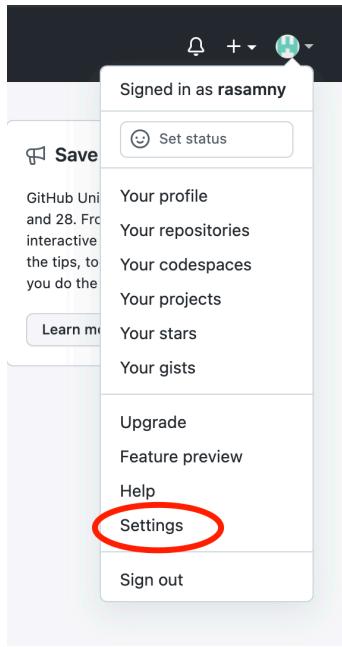
shown in the image below.

The screenshot shows a GitHub repository page for 'github-classroom'. At the top, there are navigation links: Code, Issues, Pull requests, Actions, Projects, Security, and Insights. Below these, a dropdown menu shows 'master', '1 branch', and '0 tags'. To the right are 'Go to file', 'Add file', and a green 'Code' button, which is circled in red. The main area displays a list of files: '.github' (GitHub Classroom Autograding Workflow), 'src' (Initial commit), '.gitignore' (Initial commit), and 'build.gradle' (Initial commit). Each item has a timestamp of '3 days ago'. At the bottom, there's a note to 'Add a README'.

8. You will now need to create a Github authentication token to be used in PyCharm. Click on your avatar on the top right of the screen. You should see a drop-down menu as shown in the image below. Select Settings.



9. On the Settings page, click on the Developer Setting menu item on the left as shown below.



10. On the next page, select the Personal Access Tokens as shown in the image below.

Settings / Developer settings

GitHub Apps OAuth Apps Personal access tokens

GitHub Apps

Want to build something that integrates with and extends GitHub? [Register a new GitHub App](#) to get started developing on the GitHub API. You can also read more about building GitHub Apps in our [developer documentation](#).

New GitHub App

11. Then click on the Generate new token button on the top right as shown in the image below.

Settings / Developer settings

GitHub Apps OAuth Apps Personal access tokens

Personal access tokens

Generate new token Revoke all

Tokens you have generated that can be used to access the [GitHub API](#).

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

12. Enter the word PyCharm in the Note textbox and select the repo, workflow, gist, and read:org checkboxes as shown in the image below. Also, for the expiration, select custom and enter December 31, 2021. This will make your token valid for the entire course.

GitHub Apps

OAuth Apps

Personal access tokens

New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Note

IntelliJ

What's this token for?

Expiration

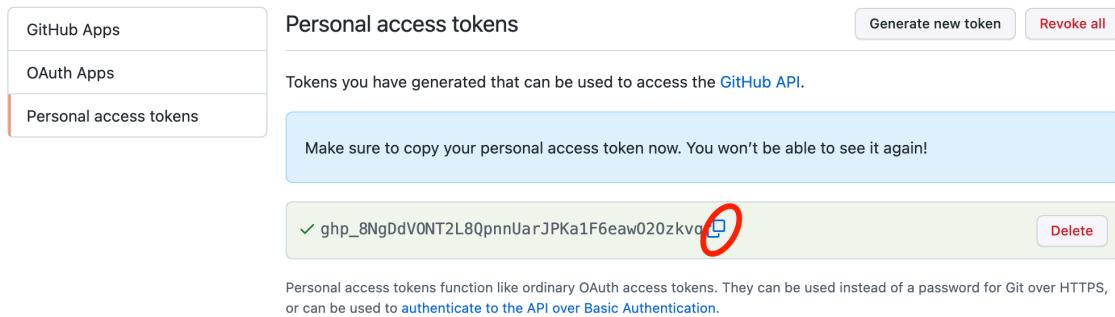
Custom... 12/31/2021

Select scope

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

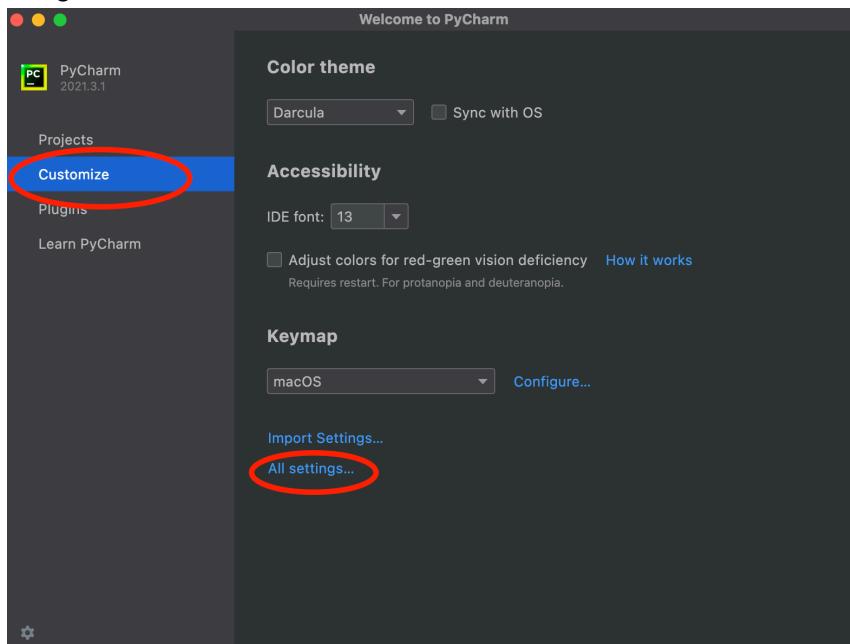
<input checked="" type="checkbox"/> repo	Full control of private repositories
<input type="checkbox"/> repo:status	Access commit status
<input type="checkbox"/> repo_deployment	Access deployment status
<input type="checkbox"/> public_repo	Access public repositories
<input type="checkbox"/> repo:invite	Access repository invitations
<input type="checkbox"/> security_events	Read and write security events
<input checked="" type="checkbox"/> workflow	Update GitHub Action workflows
<input type="checkbox"/> write:packages	Upload packages to GitHub Package Registry
<input type="checkbox"/> read:packages	Download packages from GitHub Package Registry
<input type="checkbox"/> delete:packages	Delete packages from GitHub Package Registry
<input type="checkbox"/> admin:org	Full control of orgs and teams, read and write org projects
<input type="checkbox"/> write:org	Read and write org and team membership, read and write org projects
<input checked="" type="checkbox"/> read:org	Read org and team membership, read org projects
<input type="checkbox"/> admin:public_key	Full control of user public keys
<input type="checkbox"/> admin:public_key	Full control of user public keys
<input type="checkbox"/> write:public_key	Write user public keys
<input type="checkbox"/> read:public_key	Read user public keys
<input type="checkbox"/> admin:repo_hook	Full control of repository hooks
<input type="checkbox"/> write:repo_hook	Write repository hooks
<input type="checkbox"/> read:repo_hook	Read repository hooks
<input type="checkbox"/> admin:org_hook	Full control of organization hooks
<input checked="" type="checkbox"/> gist	Create gists
<input type="checkbox"/> notifications	Access notifications
<input type="checkbox"/> user	Update ALL user data
<input type="checkbox"/> read:user	Read ALL user profile data
<input type="checkbox"/> user:email	Access user email addresses (read-only)
<input type="checkbox"/> user:follow	Follow and unfollow users

13. Click the green Generate Token button at the bottom of the screen.
14. Copy the token by clicking on the copy button as indicated in the image below. You should store the token somewhere because you will not be able to recover it. If you should forget it, you will need to regenerate a new token.

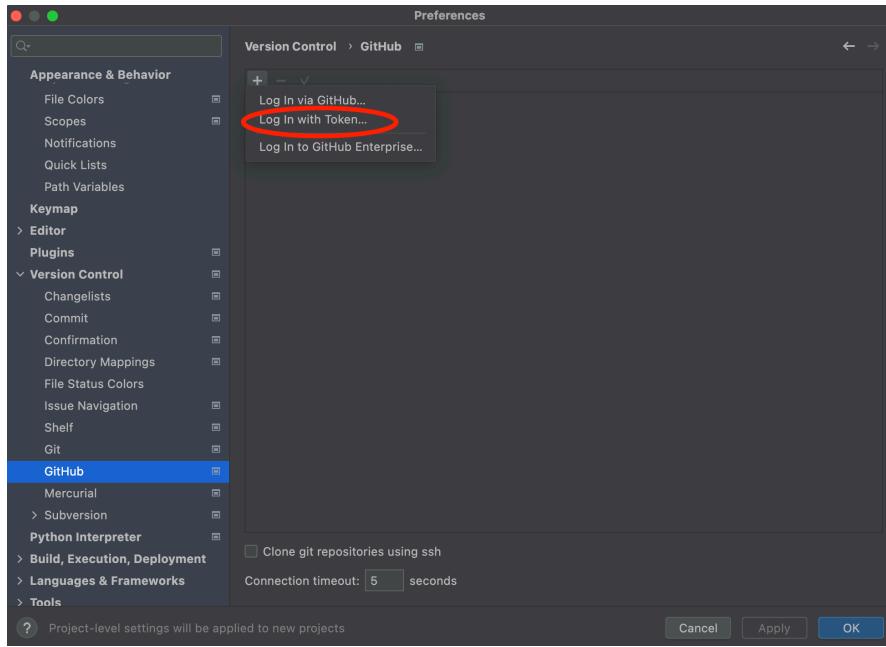


The screenshot shows the GitHub Personal access tokens page. On the left, there's a sidebar with options: GitHub Apps, OAuth Apps, and Personal access tokens (which is selected). At the top right are buttons for "Generate new token" and "Revoke all". Below that, a message says "Tokens you have generated that can be used to access the GitHub API." A note at the bottom of the list says "Make sure to copy your personal access token now. You won't be able to see it again!" A specific token, "ghp_8NgDdV0NT2L8QpnnUarJPKa1F6eaw020zkvo", is listed with a red circle around its "Copy" icon. At the bottom, a note explains that personal access tokens function like ordinary OAuth access tokens and can be used for Git over HTTPS or API authentication.

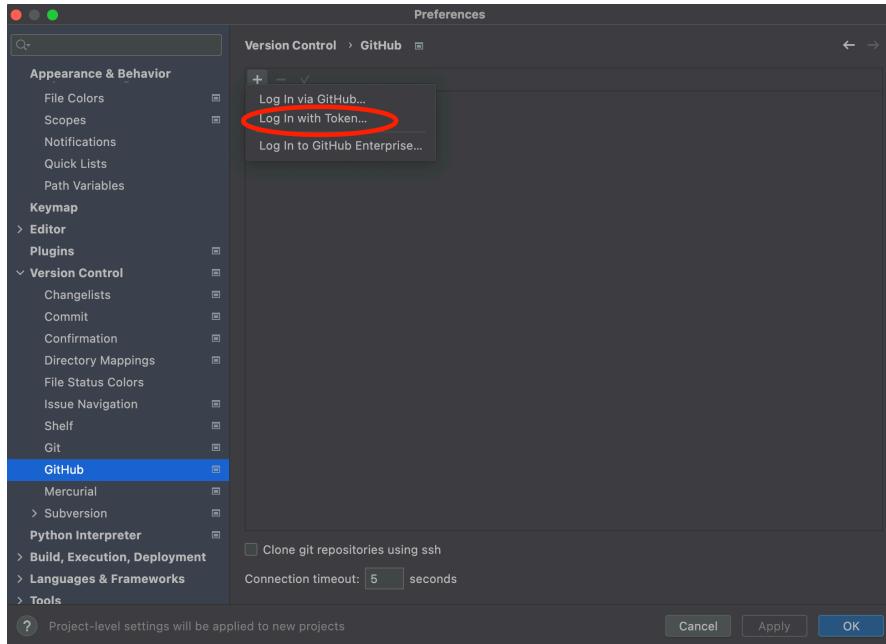
15. Now, open up PyCharm and click on Customize and then All Settings as shown in the image below.



16. Expand the Version Control item on the left of the window and select Github. Then click on the + as shown in the image below.



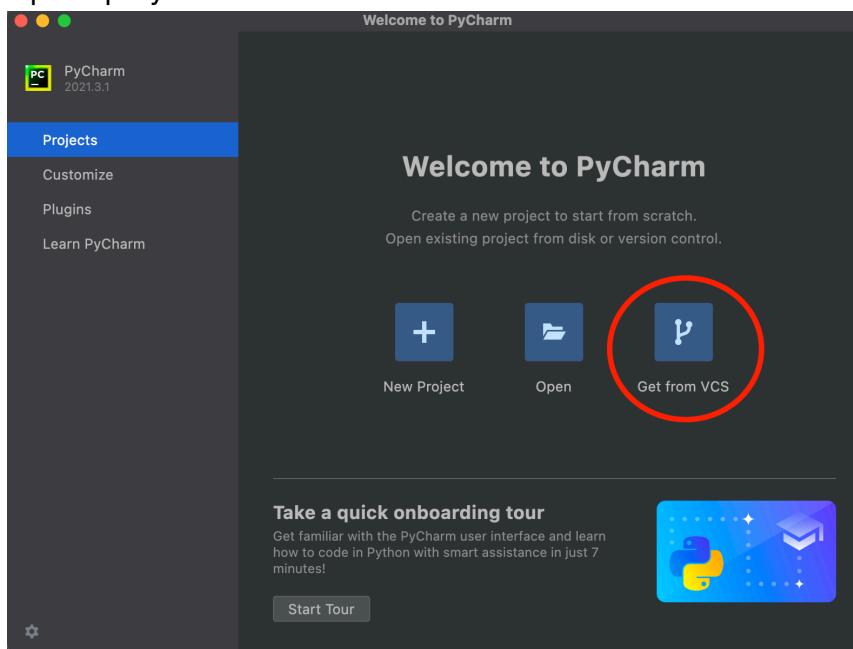
17. Select the Login with Token and paste your token in the Token textbox and click the Add Account button as shown in the image below.



18. Now, go back to the assignment you accepted and click on the link so that you are in the repo for Demo Github Classroom.
19. Click the green code button and then copy the URL by clicking on the copy button as indicated in the image below.

The screenshot shows a GitHub repository page for 'github-classroom'. The 'Code' dropdown menu is open, displaying options like 'Clone', 'HTTPS', 'SSH', 'GitHub CLI', and 'Copy' (which is circled in red). Below the clone options are 'Open with GitHub Desktop' and 'Download ZIP'.

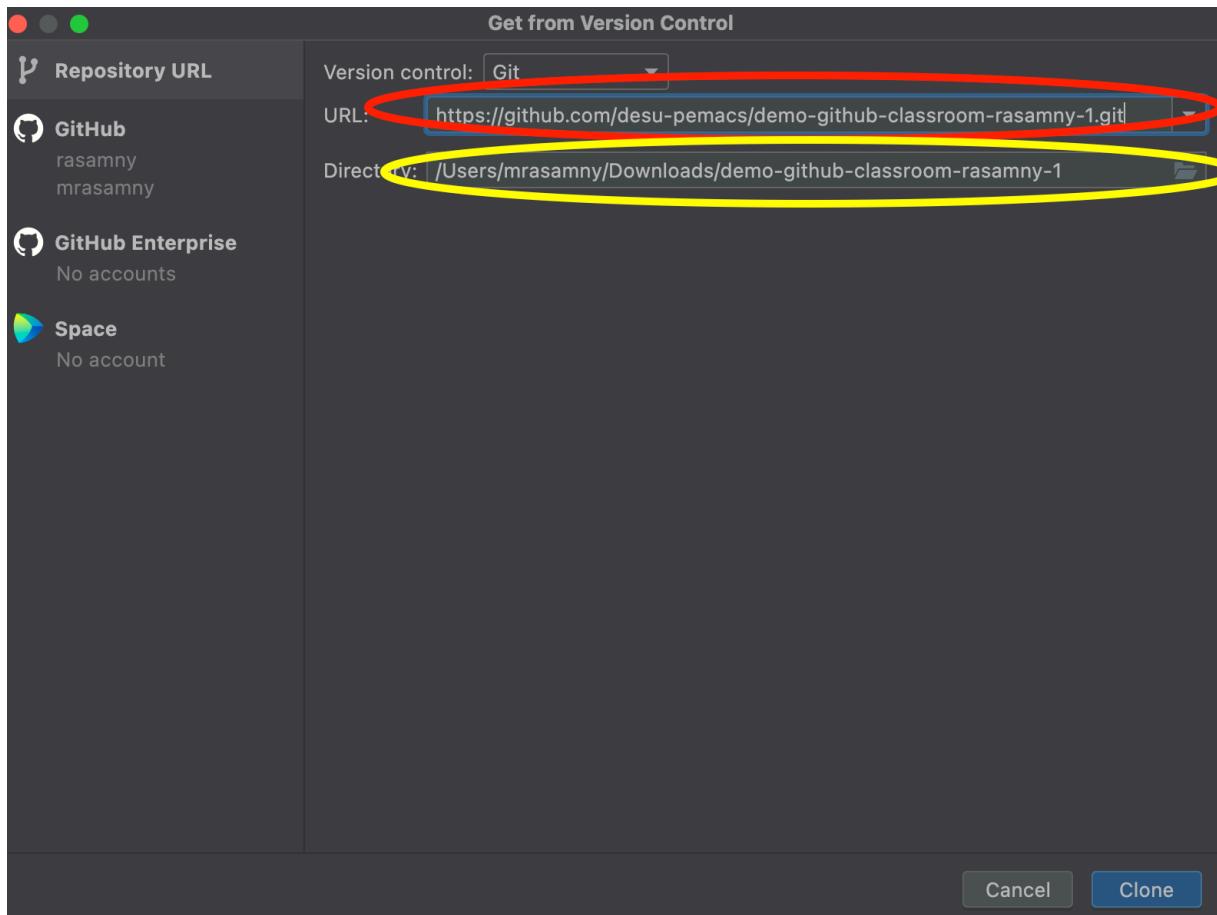
20. Open up PyCharm and use the Get from VCS button to clone the repository.



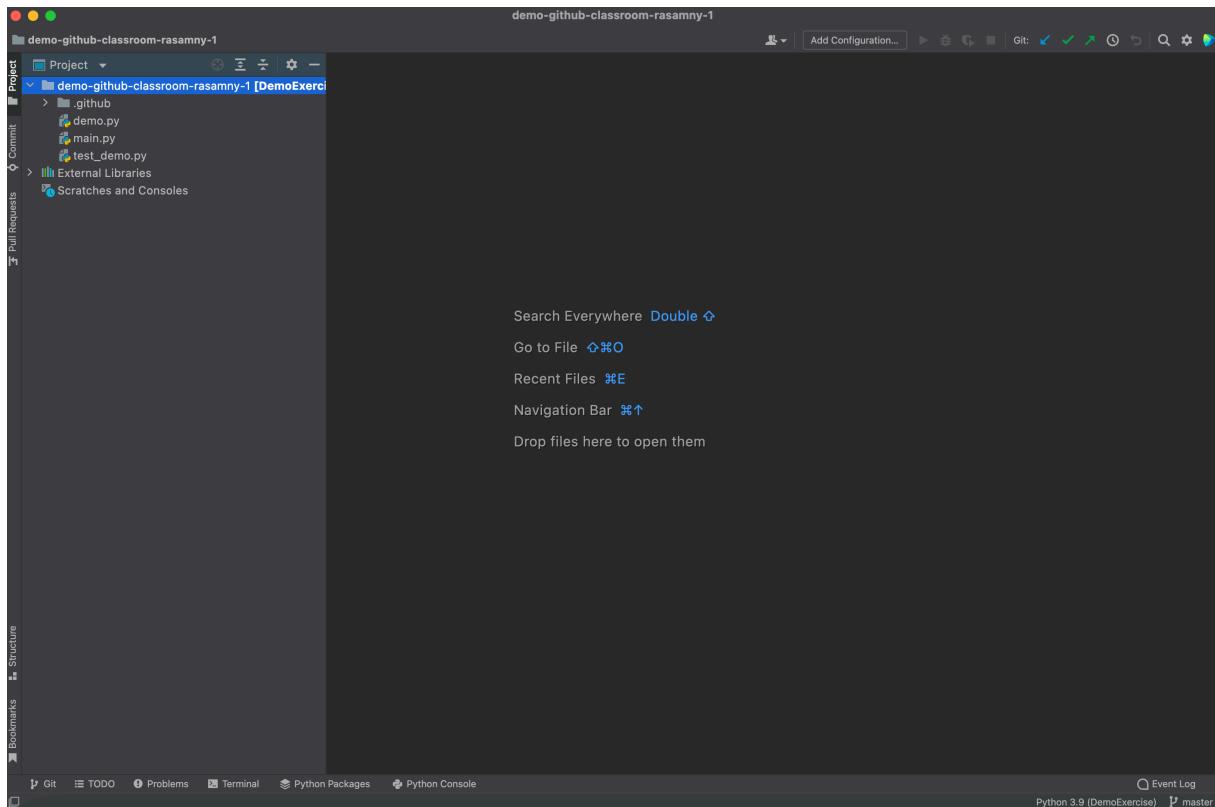
21. Paste the assignment repository URL in the URL field and make sure to select the directory in which the assignment will sit. Pycharm will add the name of repo to the Directory path. If it does not, please add the name of the assignment directory. For example, if the directory is <C:\Users\JohnSmith\Documents>, make sure to add the name of the assignment. In my case, this should be changed to <C:\Users\JohnSmith\Documents\demo-github-classroom-rasamny>

Click the link to download and install Git, if asked.

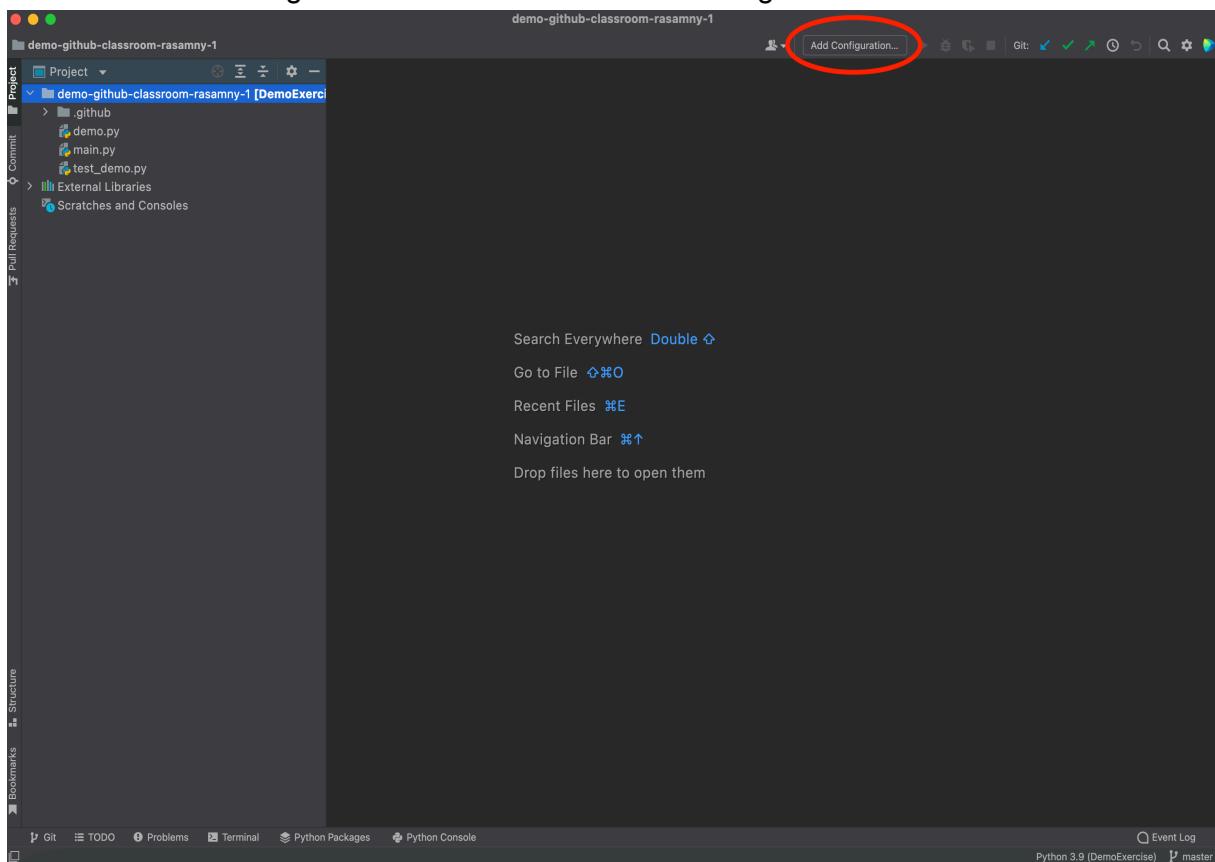
WARNING: If you do not add a folder name, Pycharm will put all the project files in the Documents folder.



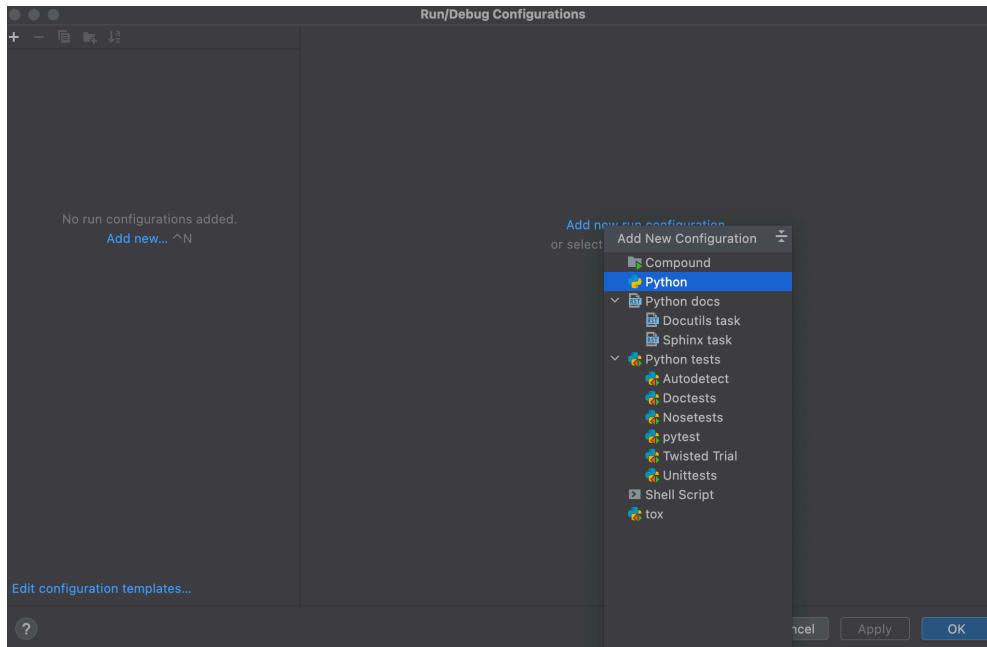
22. Press the Clone button. PyCharm will create the folder for you and place the assignment code in the folder you specified.
23. Your workspace should now look like the image shown below.



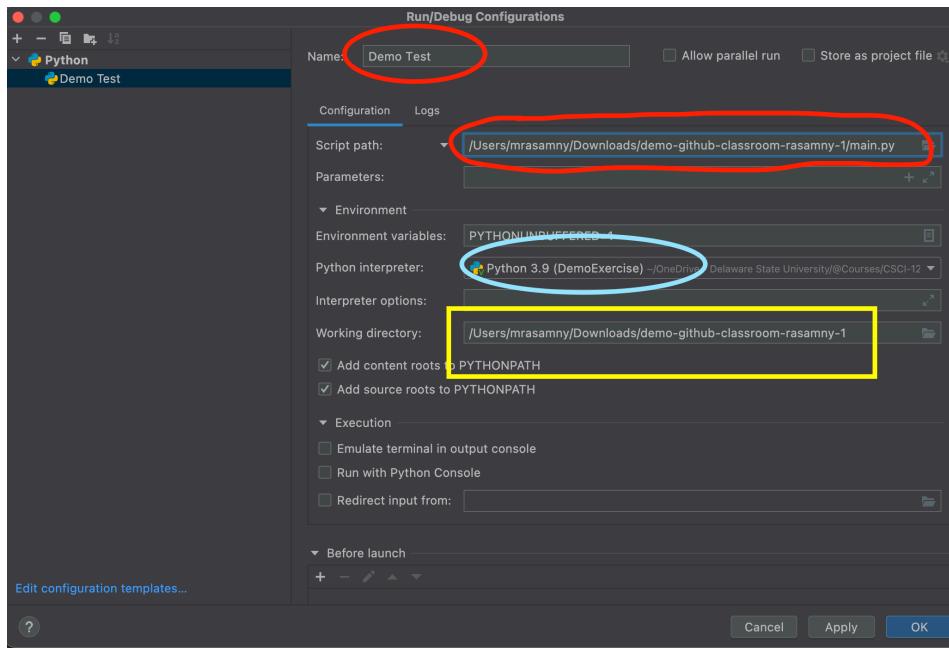
24. Now, to run and test the provided Python code, you will need to create a run configuration. Click on the run configuration button as shown in the image below.



25. Select Add New Configuration and then select Python from the menu as shown in the image below.



26. Give the configuration a name, for example **Demo Test**, and click on the folder at the right of the text box for **Script Path**. Select the *main.py* file.



27. Now press the green play button and notice that the test will fail.

The screenshot shows the PyCharm IDE interface. The top bar displays the project name "demo-github-classroom-rasamny-1" and a "Demo Test" tab. A yellow circle highlights the green play button in the toolbar. The main window shows a failed test run in the "Run" tab. The output shows three failed assertions:

```
E   assert [1, 2, 3, 4] == [1, 2, 3, 4]
E     Right contains 4 more items, first extra item: 1
E     Use -v to get the full diff
test_demo.py:14: AssertionError
=====
short test summary info =====
FAILED test_demo.py::test_remove_1 - assert [] == [1, 2, 4]
FAILED test_demo.py::test_remove_2 - assert [] == [1, 2, 3, 4]
FAILED test_demo.py::test_remove_3 - assert [] == [1, 2, 3, 4]
=====
3 failed in 0.09s =====
```

The status bar at the bottom indicates "Process finished with exit code 0".

28. Open the file `demo.py` and replace the `pass` line with the following code as shown in the figure below. Now run the test again by clicking on the green play button. You should now see that the test succeeded!

The screenshot shows the PyCharm IDE interface. The top bar displays the project name "demo-github-classroom-rasamny-1" and a "Demo Test" tab. A yellow circle highlights the green play button in the toolbar. The main window shows a successful test run in the "Run" tab. The output shows all tests passed:

```
remove() > for item in a_list    if item != item_to_remove
[100%]
=====
3 passed in 0.02s =====
```

The status bar at the bottom indicates "Process finished with exit code 0".

29. Now that the tests have all passed, you now need to commit the changes you made and

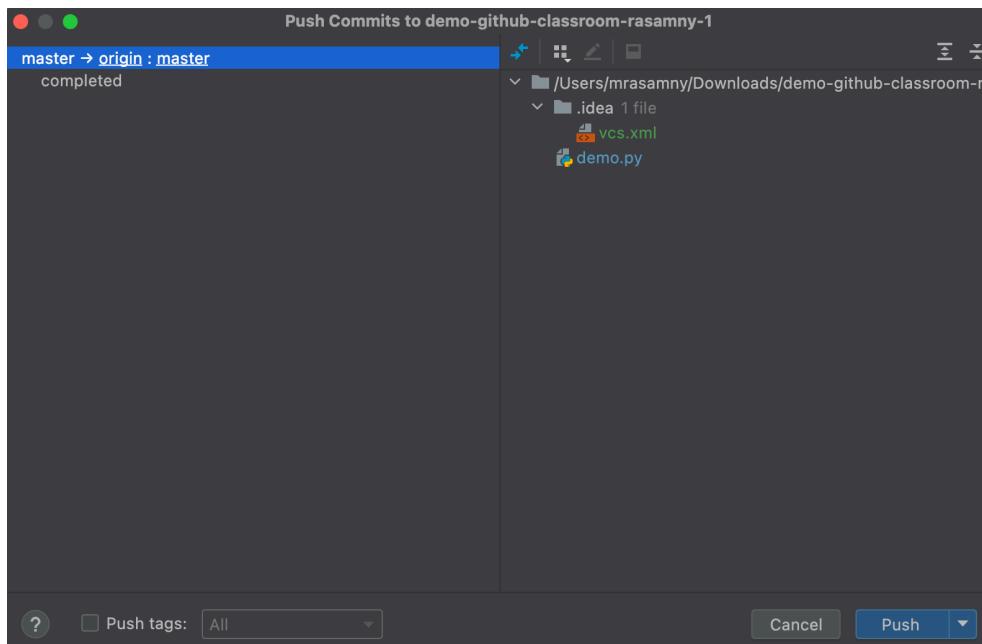
push the changes to your repository on Github. Select the Commit tab as shown in the image below.

The screenshot shows the PyCharm IDE interface. The left sidebar has tabs for 'Project' (selected), 'Commit' (circled in red), and 'Pull Requests'. The main area shows a Python file 'demo.py' with code for a 'remove' function. Below the code editor is a 'Run' tool window showing a successful test run. At the bottom, the navigation bar includes 'Git' (circled in red), 'Run', 'TODO', 'Problems', 'Python Packages', 'Python Console', and 'Terminal'. The status bar at the bottom right indicates the current branch is 'master'.

30. Select all the files that have changed and type a message in th text box as shown in the image, the message should summarize the work done at that point, and click commit and push button.

The screenshot shows the PyCharm IDE interface. On the left, the Project tool window displays a 'Changes' section with two files selected: 'demo.py' and 'vcs.xml'. A yellow circle highlights the 'Commit' button in the toolbar above the changes list. Below the changes list, a message 'completed' is circled in yellow. At the bottom of the screen, a terminal window shows the output of a 'Demo' run, indicating a successful test session with 3 passed tests in 0.02s. The status bar at the bottom right shows the Git log, Python version (3.9), and current branch (master).

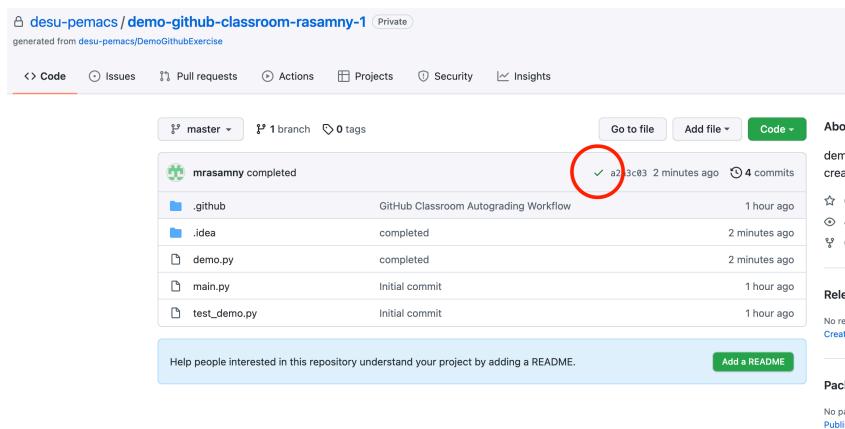
31. Click on the push button in the next window to complete the submission.



32. You should see a message that the files were pushed successfully. You can verify by looking at the Git log. Click the Git tab on the bottom left as shown in the image below.

The screenshot shows the GitHub Desktop application interface. At the top, there's a code editor window for a file named 'demo.py'. Below the code editor is a 'Changes' section with a single commit labeled 'completed'. At the bottom of the application, there's a commit history pane showing several commits. One commit by 'mrasamny' is circled in red and has a green checkmark next to it, indicating it passed the tests.

33. Go back to your repo on Github and make sure that you see a check mark indicating that everything passed. You may need to reload the page to see any changes. See image below.



Congraulations! You now have completed your first recitation!