pycharm

Contents

Note

The instructions here are for setting up PyCharm Community Edition. PyCharm is a powerful Integrated Development Environment (IDE) that provides integration with python, conda, and git. In the practicals, you'll see how you can use it to write, edit, test, and debug your code.

You are **not** required to use PyCharm - there are many other good, free, options out there, including VS Code and Spyder. Many of the steps for the debugging exercise in Week 1 will be the same, broadly-speaking, for other IDEs. If you do use a different IDE, I will do my best to provide help, but just note that it may be some more work to get set up.

Skip to main content

⚠ Warning

Many of the instructions and screenshots shown here are for PyCharm CE 2021.3.3, running on Ubuntu Linux 18.04.6.

As such, there may be some slight differences in appearance or menu options between what is shown here and the version you download.

downloading and installing

To download PyCharm CE, point your browser to https://www.jetbrains.com/pycharm/download/ and select the version for your current operating system (Windows, MacOS, or linux).

⚠ Warning

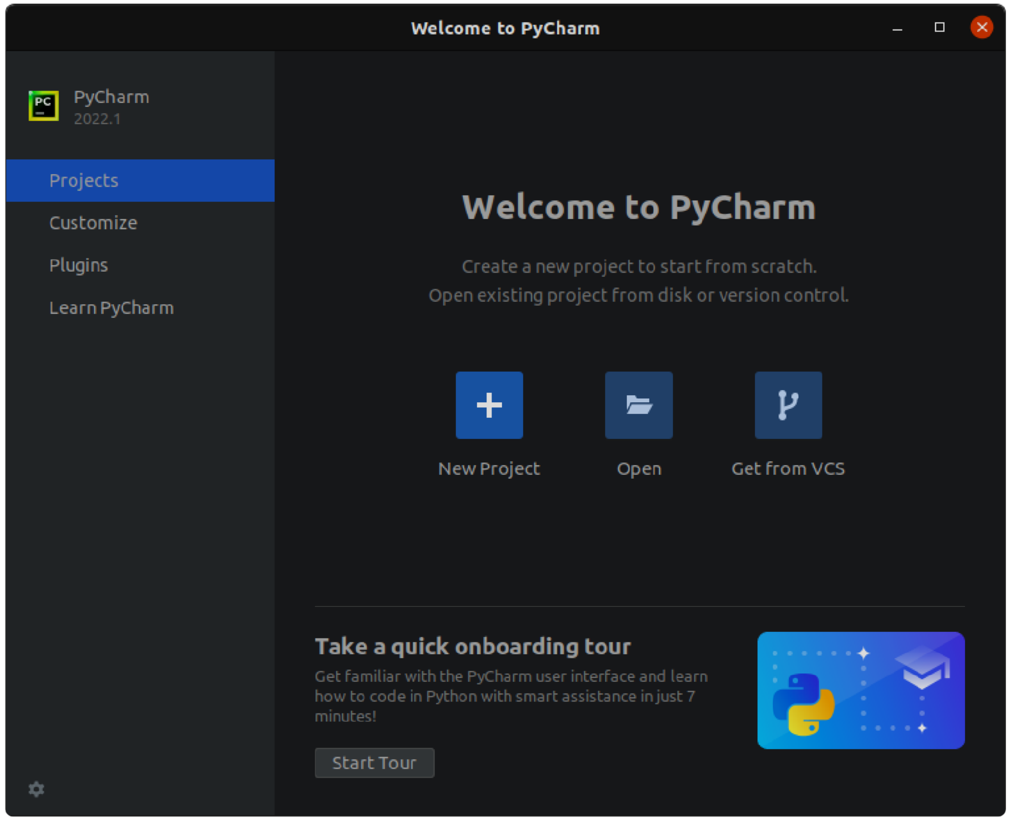
Be sure to download the **Community Edition**, which is free. If you download the other version, you will need to purchase a license once the free trial expires after 30 days.

Once you have downloaded the installer, follow the instructions in the installation wizard. Once the installation has completed, open PyCharm CE.

creating a new project

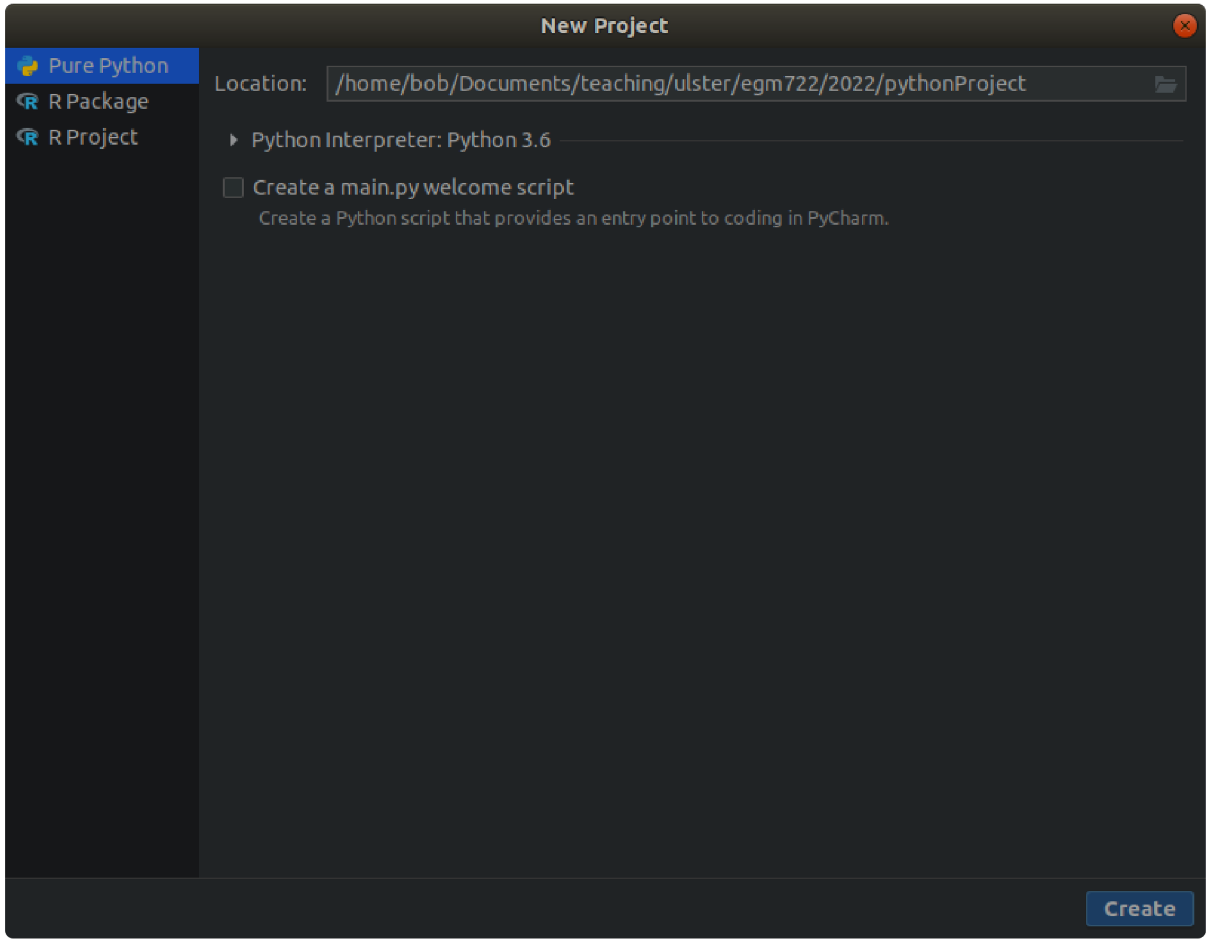
When you first open PyCharm, you should see a screen similar to the following:

Skip to main content


Welcome to PyCharm
PyCharm
2022.1
Projects
Customize
Plugins
Learn PyCharm
Welcome to PyCharm
Create a new project to start from scratch.
Open existing project from disk or version control.
New Project
Open
Get from VCS
Take a quick onboarding tour
Get familiar with the PyCharm user interface and learn how to code in Python with smart assistance in just 7 minutes!
Start Tour

Select **Create New Project**, which will open the following window:

Skip to main content



Location: /home/bob/Documents/teaching/ulster/egm722/2022/pythonProject

Python Interpreter: Python 3.6

Create a main.py welcome script

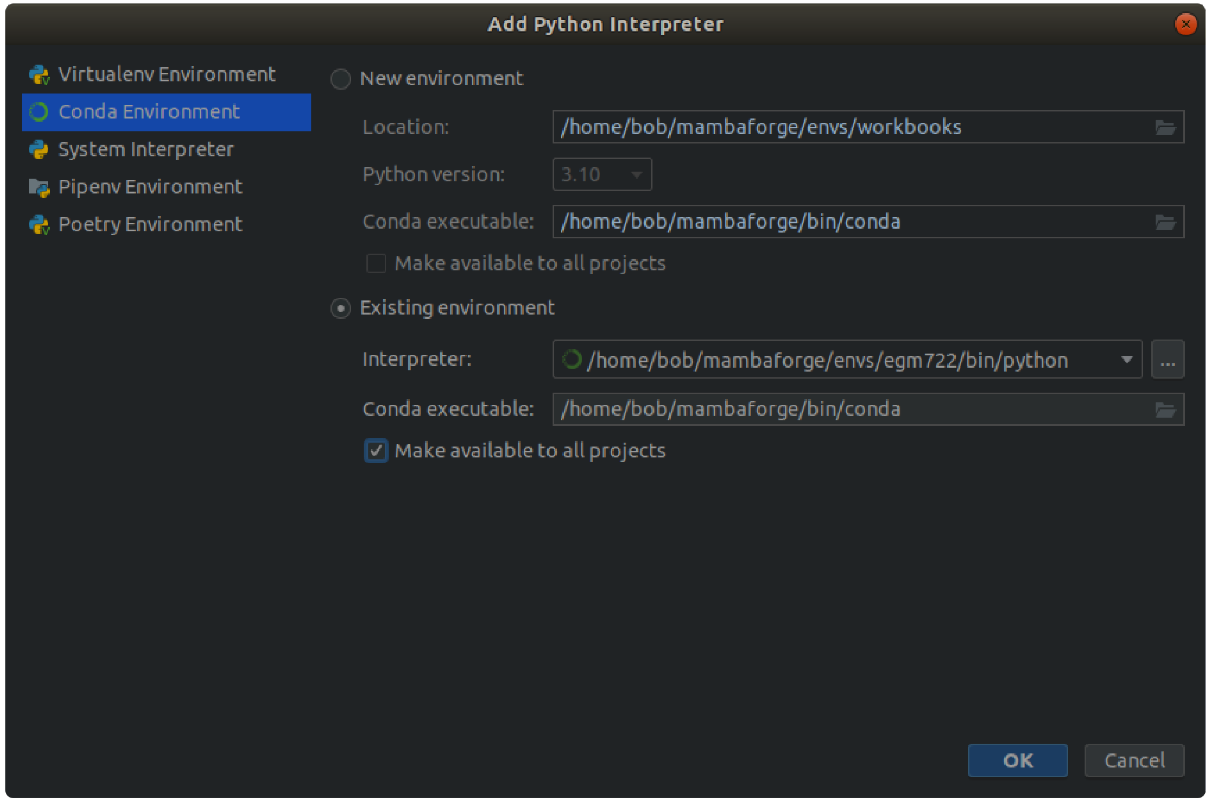
Create a Python script that provides an entry point to coding in PyCharm.

For the location, choose the folder where you have cloned the `egm722` repository by clicking on the folder icon and selecting the folder from the tree (in this example, the folder is located in `/home/bob/Documents/teaching/ulster/egm722/2022/workbooks`).

You should also set up a python interpreter for the project - this will enable you to use some of the autocomplete tools when writing code, run scripts within PyCharm, as well as making use of the debugging tools.

Skip to main content

then **Existing environment** - you should see something like the following:



New environment

Location: /home/bob/mambaforge/envs/workbooks

Python version: 3.10

Conda executable: /home/bob/mambaforge/bin/conda

Make available to all projects

● Existing environment

Interpreter: /home/bob/mambaforge/envs/egm722/bin/python

Conda executable: /home/bob/mambaforge/bin/conda

✓ Make available to all projects

You'll need to provide the path to the python interpreter that is part of your `egm722` conda environment - on most computers, relative to where you've installed Anaconda, this will be[2]:

```
~/Anaconda3/envs/egm722/bin/python
```

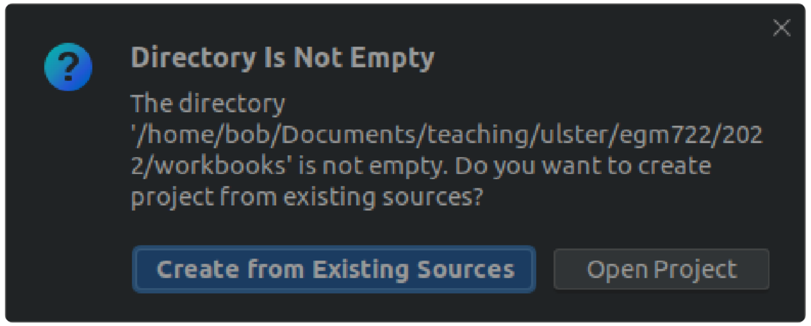
(or, if you're on Windows, `python.exe`). You will also need to supply the path to the `conda` executable (program) that's installed on your computer - again, on most computers, relative to where you've installed Anaconda, this will be:

Skip to main content

```
~/Anaconda3/bin/conda
```

(or, if you're on Windows, `conda.exe`). You can also click the **Make available to all projects** checkbox, which will help you more easily add this interpreter to other new projects. Click **OK** to return to the previous window.

Once you've set the path to the project, and set up the python interpreter, click **Create**. The following window should pop up:



Directory Is Not Empty

The directory '/home/bob/Documents/teaching/ulster/egm722/2022/workbooks' is not empty. Do you want to create project from existing sources?

Select **Create from Existing Sources**.

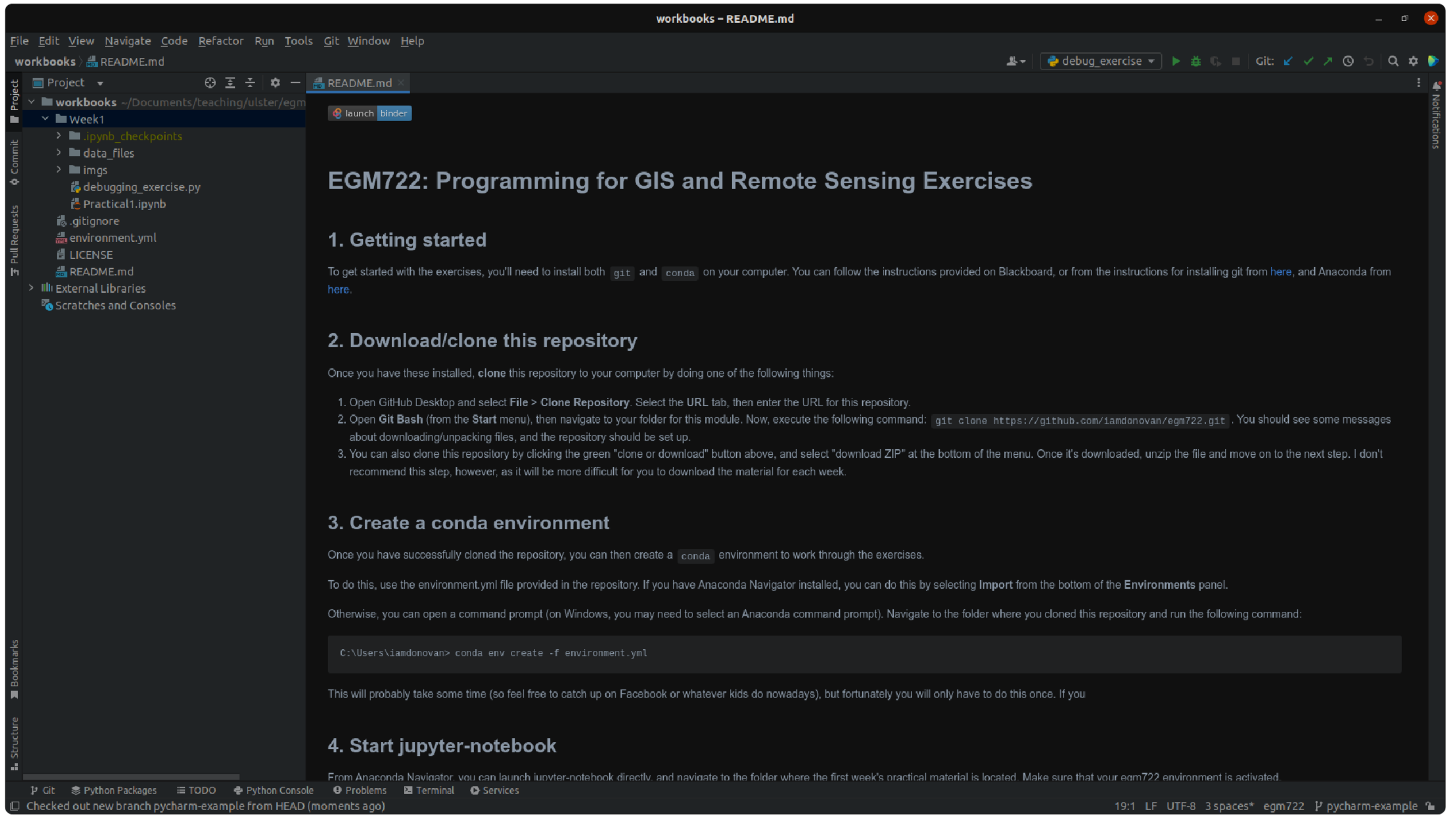
Note

This is the end of the "setup" steps - the rest of the sections of this page provide some more information about working in PyCharm, but they aren't required to actually get started on the practicals.

the pycharm interface

Once the new project has been created, you should see something like this:

Skip to main content



EGM722: Programming for GIS and Remote Sensing Exercises

1. Getting started

To get started with the exercises, you'll need to install both `git` and `conda` on your computer. You can follow the instructions provided on Blackboard, or from the instructions for installing git from here, and Anaconda from here.

2. Download/clone this repository

Once you have these installed, **clone** this repository to your computer by doing one of the following things:

1. Open GitHub Desktop and select **File > Clone Repository**. Select the **URL** tab, then enter the URL for this repository.
2. Open **Git Bash** (from the **Start** menu), then navigate to your folder for this module. Now, execute the following command: `git clone https://github.com/iamdonovan/egm722.git`. You should see some messages about downloading/unpacking files, and the repository should be set up.
3. You can also clone this repository by clicking the green "clone or download" button above, and select "download ZIP" at the bottom of the menu. Once it's downloaded, unzip the file and move on to the next step. I don't recommend this step, however, as it will be more difficult for you to download the material for each week.

3. Create a conda environment

Once you have successfully cloned the repository, you can then create a `conda` environment to work through the exercises.

To do this, use the environment.yml file provided in the repository. If you have Anaconda Navigator installed, you can do this by selecting **Import** from the bottom of the **Environments** panel.

Otherwise, you can open a command prompt (on Windows, you may need to select an Anaconda command prompt). Navigate to the folder where you cloned this repository and run the following command:

```
C:\Users\iamdonovan> conda env create -f environment.yml
```

This will probably take some time (so feel free to catch up on Facebook or whatever kids do nowadays), but fortunately you will only have to do this once. If you

4. Start jupyter-notebook

From Anaconda Navigator, you can launch jupyter-notebook directly, and navigate to the folder where the first week's practical material is located. Make sure that your egm722 environment is activated.

Checked out new branch pycharm-example from HEAD (moments ago)

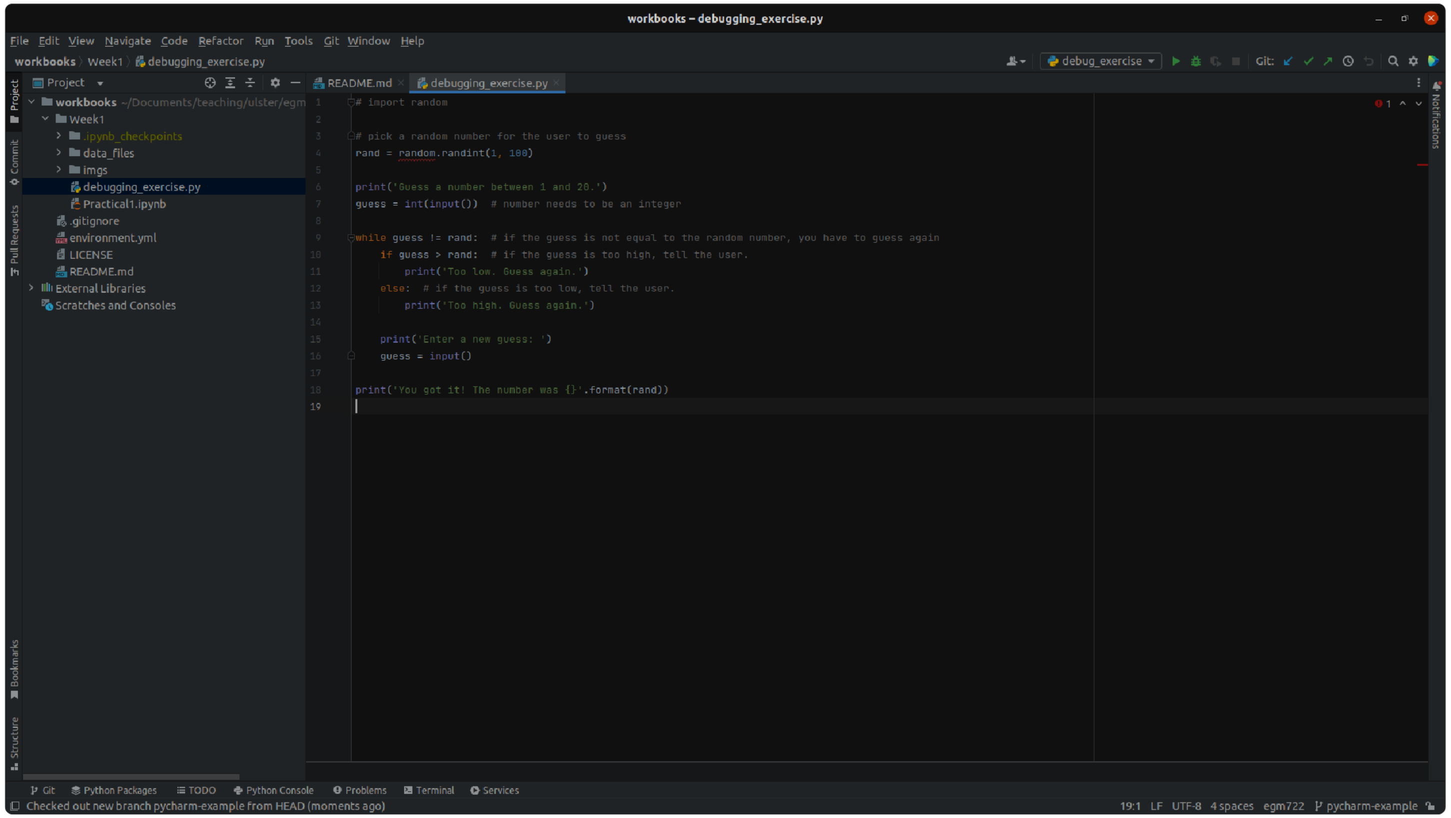
The next few sections will provide some additional information about some of the features of PyCharm.

the file browser

On the left side of the **PyCharm** window, you should see a file browser (note that you may need to click on the **Project** tab to expand the frame).

Here, you can see all of the files and folders included in the project. To open a script or other file, double-click on it in the file browser. For example, here is `debugging_exercise.py`:

Skip to main content


workbooks – debugging_exercise.py
File Edit View Navigate Code Refactor Run Tools Git Window Help
workbooks › Week1 › debugging_exercise.py
Project
workbooks ~/Documents/teaching/ulster/egm
Week1
.ipynb_checkpoints
data_files
imgs
debugging_exercise.py
Practical1.ipynb
.gitignore
environment.yml
LICENSE
README.md
External Libraries
Scratches and Consoles
README.md
debugging_exercise.py
debug_exercise
Git Python Packages TODO Python Console Problems Terminal Services
19:1 LF UTF-8 4 spaces egm722 pycharm-example

```python
# import random

# pick a random number for the user to guess
rand = random.randint(1, 100)

print('Guess a number between 1 and 20.')
guess = int(input())  # number needs to be an integer

while guess != rand:  # if the guess is not equal to the random number, you have to guess again
    if guess > rand:  # if the guess is too high, tell the user.
        print('Too low. Guess again.')
    else:  # if the guess is too low, tell the user.
        print('Too high. Guess again.')

    print('Enter a new guess: ')
    guess = input()

print('You got it! The number was {}'.format(rand))
```

Checked out new branch pycharm-example from HEAD (moments ago)

creating a new script

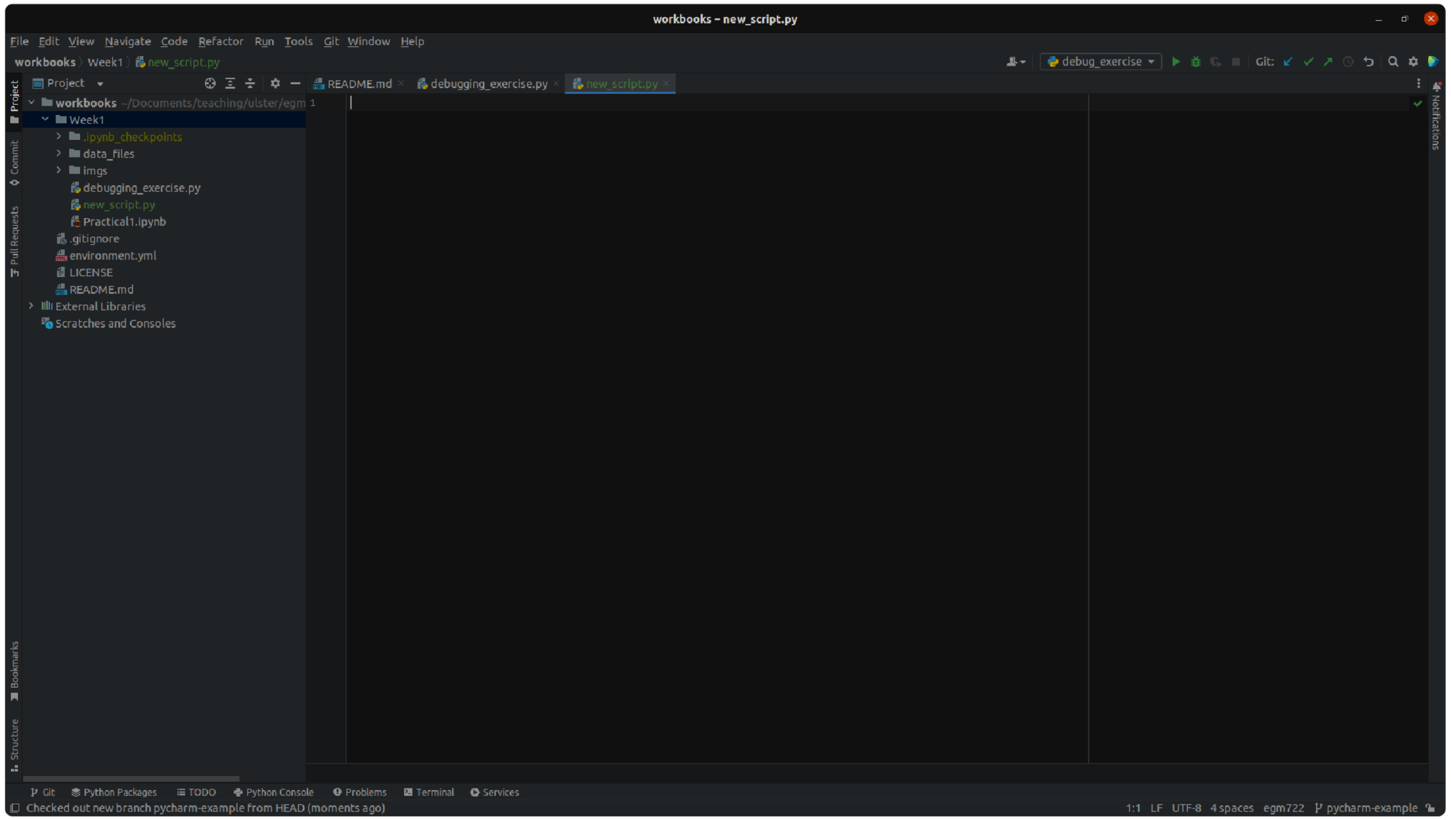
You can also create a new file by right-clicking in the file browser and selecting **New**, followed by the file type. To create a new python script, it would be: right-click > **New** > **Python File**.

Type the name of the new script in the box that appears, and press **Enter**.

⚠ Warning

Skip to main content

PyCharm interfaces with **git** to help you keep track of your projects, so it asks you if you want to add the file to **git**. Click **Add** to do this, and you should see the new file appear in the file browser, and in the main panel:

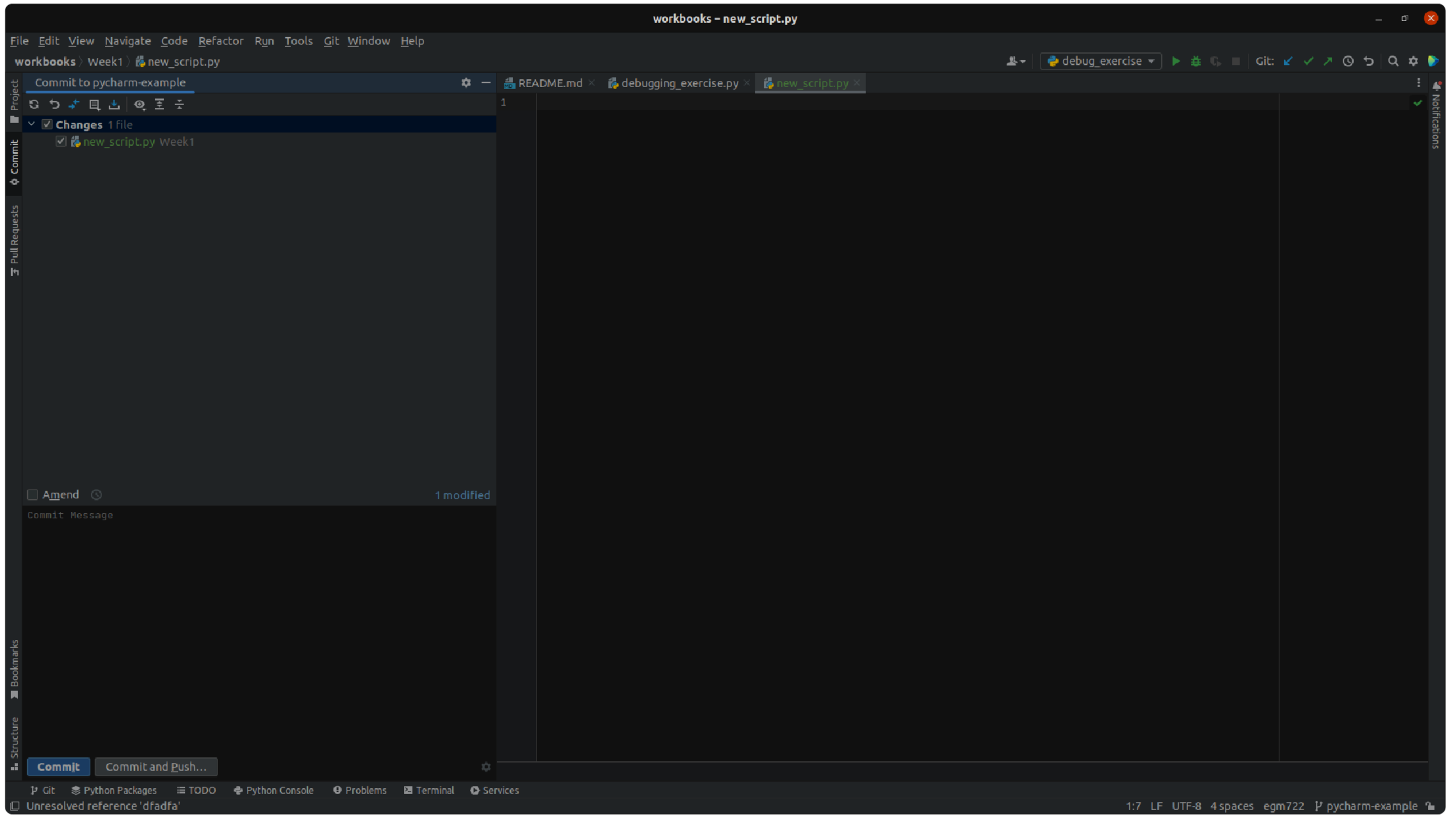


using git

Note that the filename is colored in green, indicating that the file has been *staged* by **git**. Just like with **GitHub Desktop**, we can use PyCharm along with git to keep track of changes to our files.

Under the **Project** tab, you should see a **Commit** tab - click this to bring up the list of current changes:

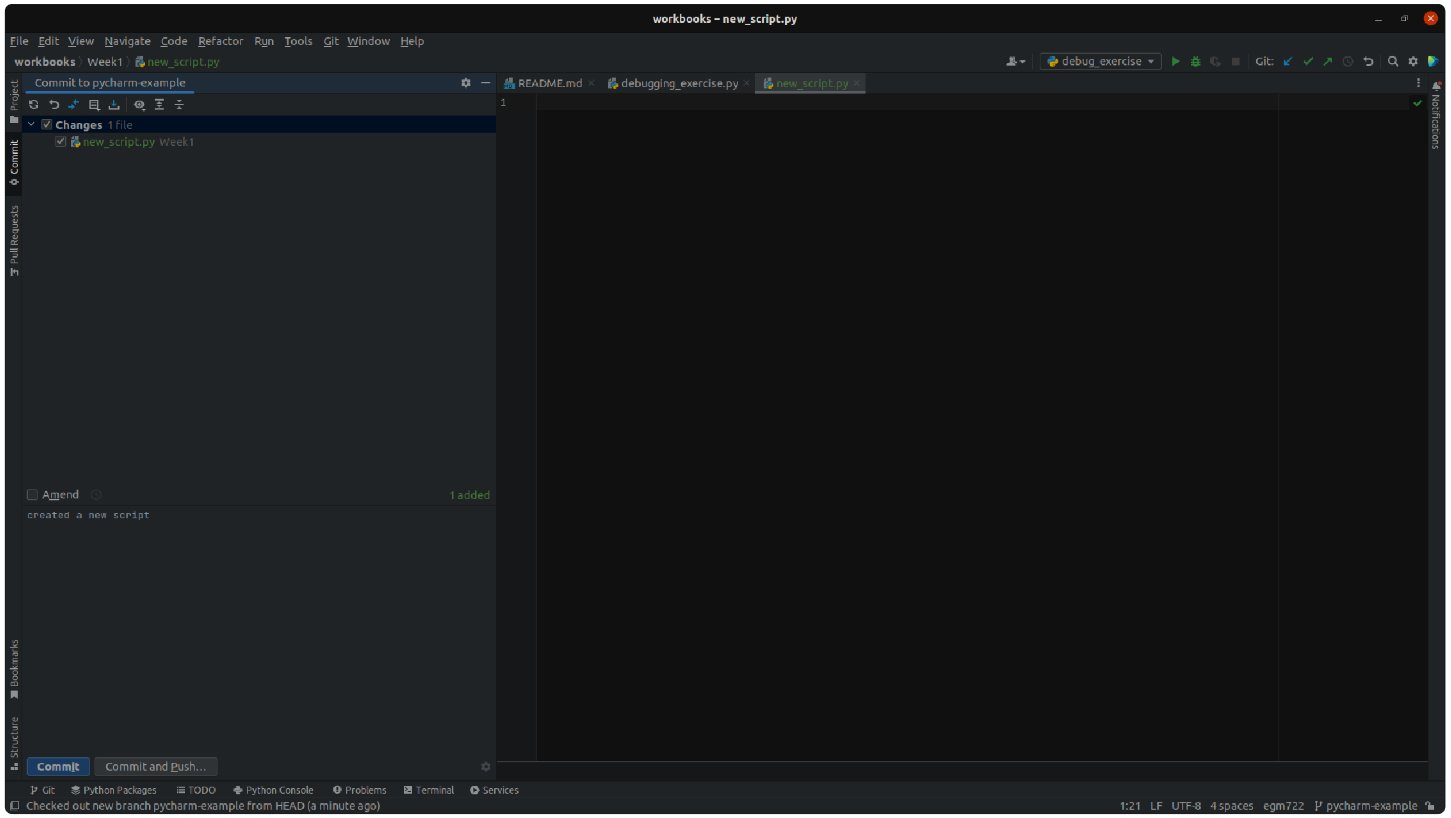
Skip to main content


workbooks – new_script.py
File Edit View Navigate Code Refactor Run Tools Git Window Help
workbooks Week1 new_script.py
debug_exercise
Commit to pycharm-example
README.md debugging_exercise.py new_script.py
Changes 1 file
new_script.py Week1
Amend
1 modified
Commit Message
Commit
Commit and Push...
Git Python Packages TODO Python Console Problems Terminal Services
Unresolved reference 'dfadfa'
1:7 LF UTF-8 4 spaces egm722 pycharm-example

In the list of current changes, we can see that one file has been affected: `new_script.py`.

Before we press **Commit**, we should first write a commit message, which we can do in the lower box of the panel:

Skip to main content


workbooks – new_script.py
File Edit View Navigate Code Refactor Run Tools Git Window Help
workbooks › Week1 › new_script.py
Commit to pycharm-example
README.md
debugging_exercise.py
new_script.py
Changes 1 file
new_script.py Week1
Amend
1 added
Commit
Commit and Push...
Git
Python Packages
TODO
Python Console
Problems
Terminal
Services
1:21 LF UTF-8 4 spaces egm722 pycharm-example

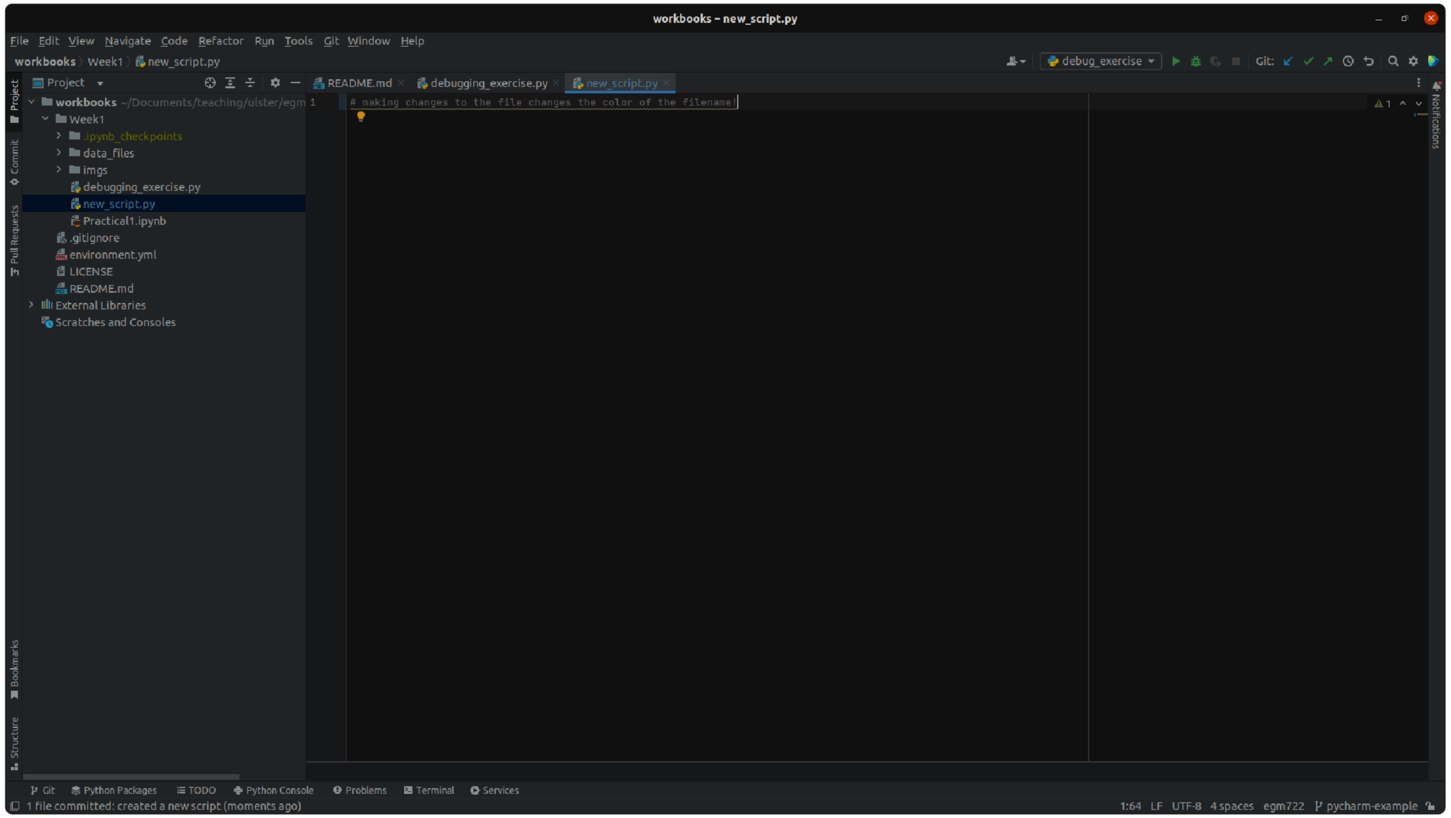
```
created a new script
```

Checked out new branch pycharm-example from HEAD (a minute ago)

Press **Commit**, and you should see the filename change from green to white.

Now, when you make changes to the file, the color will change again, this time to blue:

Skip to main content



```python
# making changes to the file changes the color of the filename!
```

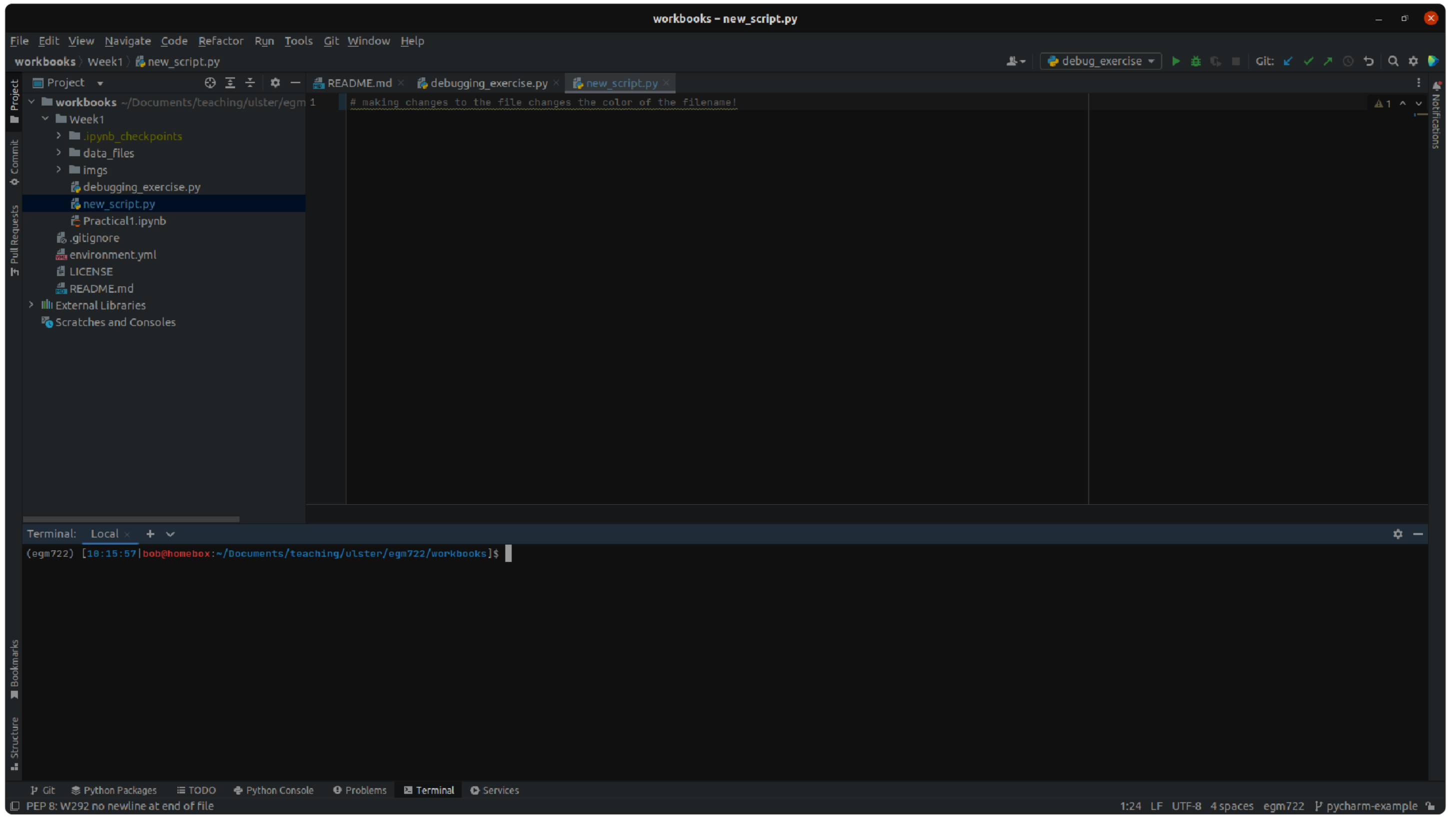
```
1 file committed: created a new script (moments ago)
```

This indicates that we have unstaged and uncommitted changes to the file[3]. We don't need to commit every small change that we make, but it's a idea to do this regularly, and to **push** the changes to the remote repository (so that they're backed up). Don't worry about all of this now - we will come back to these ideas over the course of the module.

terminal interface

Along the bottom of the PyCharm window, you should see a number of tabs, including one that says **Terminal**. Click this to open a **Terminal** (**Command Prompt**):

Skip to main content

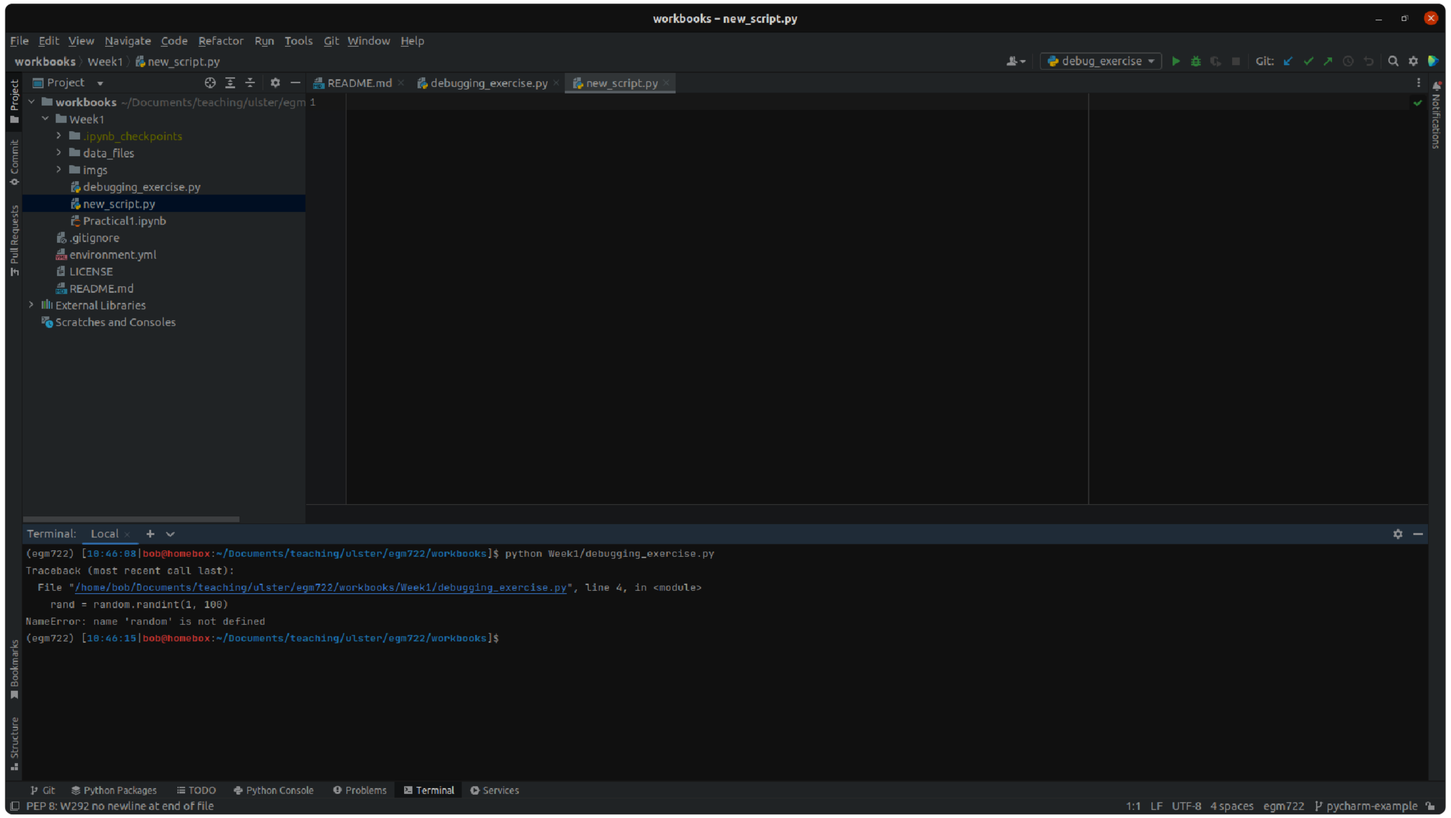


```
# making changes to the file changes the color of the filename!
```

```
(egm722) [10:15:57|bob@homebox:~/Documents/teaching/ulster/egm722/workbooks]$
```

You can use this in the same way that you would the “normal” **Command Prompt**. If you have configured an interpreter for your project, it should open as an **Anaconda Command Prompt** with that environment activated, meaning that you can use it to run scripts in that environment:

Skip to main content

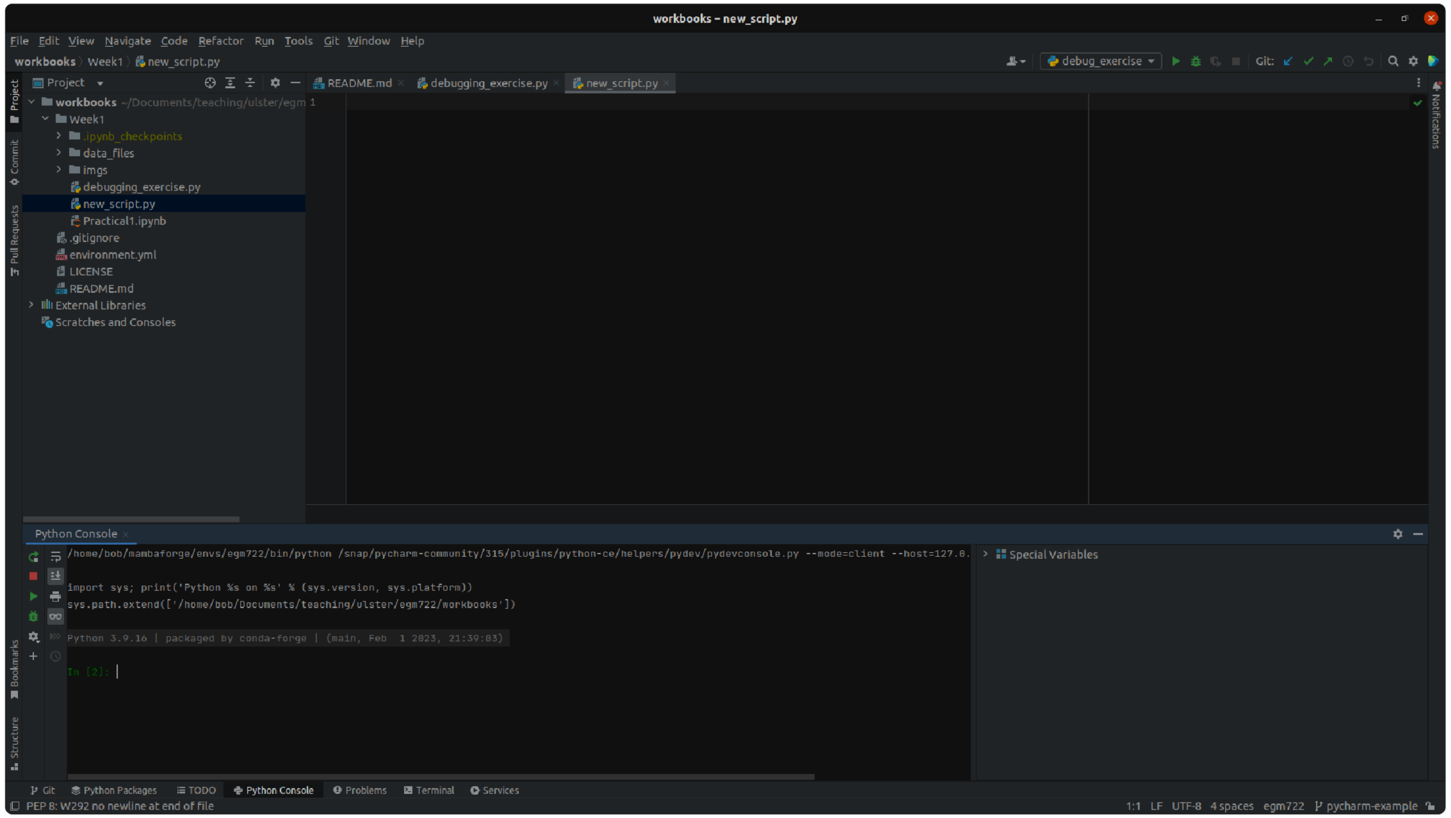


```
(egm722) [10:46:08|bob@homebox:~/Documents/teaching/ulster/egm722/workbooks]$ python Week1/debugging_exercise.py
Traceback (most recent call last):
  File "/home/bob/Documents/teaching/ulster/egm722/workbooks/Week1/debugging_exercise.py", line 4, in <module>
    rand = random.randint(1, 100)
NameError: name 'random' is not defined
(egm722) [10:46:15|bob@homebox:~/Documents/teaching/ulster/egm722/workbooks]$
```

python console

Next to the **Terminal** tab, you should also see a **Python Console** tab:

Skip to main content

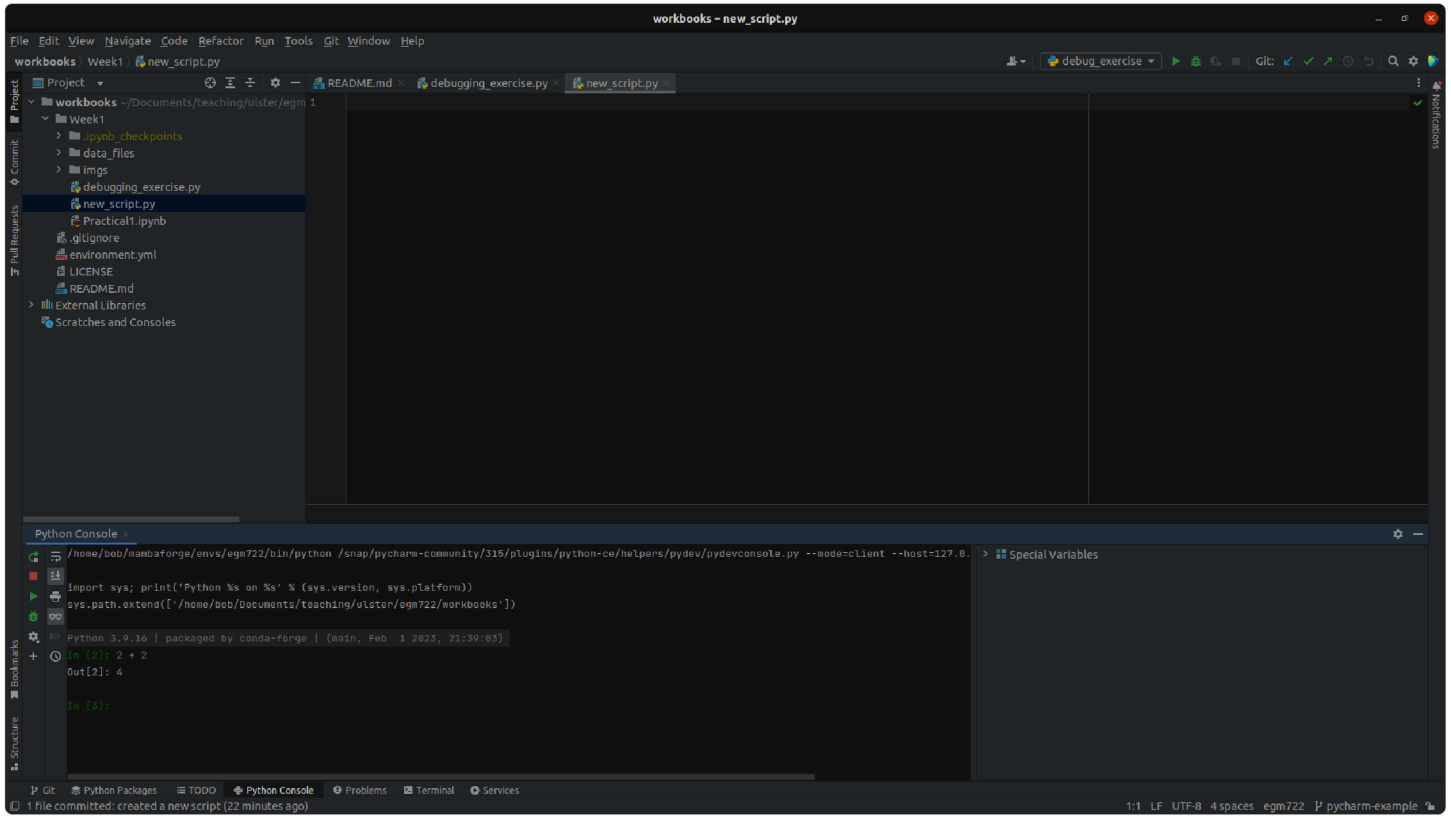


```
/home/bob/mambaforge/envs/egm722/bin/python /snap/pycharm-community/315/plugins/python-ce/helpers/pydev/pydevconsole.py --mode=client --host=127.0.

import sys; print('Python %s on %s' % (sys.version, sys.platform))
sys.path.extend(['/home/bob/Documents/teaching/ulster/egm722/workbooks'])

Python 3.9.16 | packaged by conda-forge | (main, Feb  1 2023, 21:39:03)

In [2]:
```

workbooks – new_script.py
File Edit View Navigate Code Refactor Run Tools Git Window Help
workbooks › Week1 › new_script.py
debug_exercise
Project
README.md
debugging_exercise.py
new_script.py
workbooks ~/Documents/teaching/ulster/egm
Week1
.ipynb_checkpoints
data_files
imgs
debugging_exercise.py
new_script.py
Practical1.ipynb
.gitignore
environment.yml
LICENSE
README.md
External Libraries
Scratches and Consoles
Python Console
Special Variables
Git Python Packages TODO Python Console Problems Terminal Services
PEP 8: W292 no newline at end of file
1:1 LF UTF-8 4 spaces egm722 pycharm-example

This opens an **iPython** (**i**nteractive **Python**) terminal, which you can use in the same way that you would use the cells of a **jupyter** notebook:

Skip to main content



```
/home/bob/mambaforge/envs/egm722/bin/python /snap/pycharm-community/315/plugins/python-ce/helpers/pydev/pydevconsole.py --mode=client --host=127.0.

import sys; print('Python %s on %s' % (sys.version, sys.platform))
sys.path.extend(['/home/bob/Documents/teaching/ulster/egm722/workbooks'])

Python 3.9.16 | packaged by conda-forge | (main, Feb  1 2023, 21:39:03)
In [2]: 2 + 2
Out[2]: 4

In [3]:
```

more resources

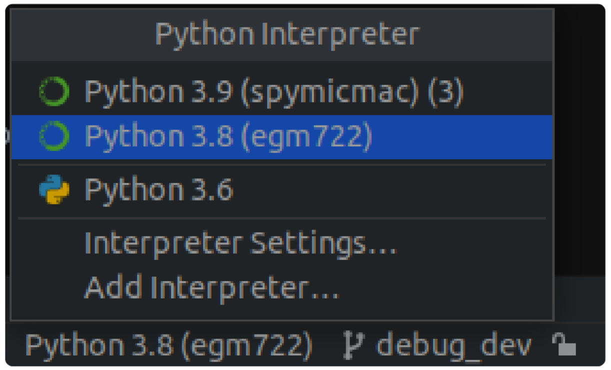
That ends the “tour” of PyCharm - there are many, many more features available, but this should provide enough to get you started. If you’re interested in learning more of the features, here are a few links to get you going:

- Quick start guide [JetBrains]
- Quick Tour [JetBrains]
- Pycharm Tutorial #1 [Tech With Tim]

Skip to main content

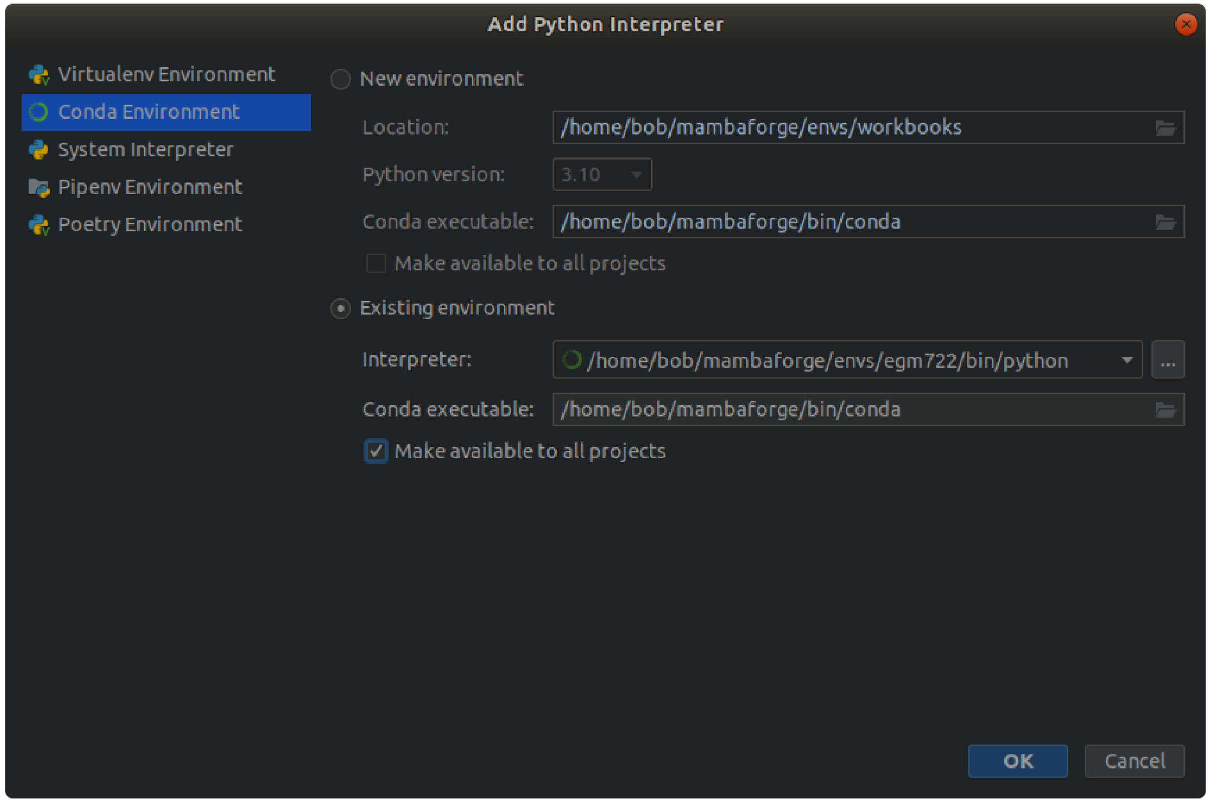
adding an interpreter

If you didn't add an interpreter to your project, don't panic. If you want to add an interpreter to an existing project, or even change the interpreter for the project, there's a way. In the lower right-hand side of the window, you should see something that says "Python 3.8" (or something similar). Click on this to open the interpreter menu:



Select **Add Python Interpreter**. In the window that opens up, select **Conda Environment**, followed by **Existing environment** - you should see something like the following:

Skip to main content


Add Python Interpreter

( ) New environment

Location: /home/bob/mambaforge/envs/workbooks

Python version: 3.10

Conda executable: /home/bob/mambaforge/bin/conda

[ ] Make available to all projects

(●) Existing environment

Interpreter: /home/bob/mambaforge/envs/egm722/bin/python

Conda executable: /home/bob/mambaforge/bin/conda

[✓] Make available to all projects

You'll need to provide the path to the python interpreter that is part of your `egm722` conda environment - on most computers, relative to where you've installed Anaconda, this will be:

```
~/Anaconda3/envs/egm722/bin/python
```

(or, if you're on Windows, `python.exe`). You will also need to supply the path to the `conda` executable (program) that's installed on your computer - again, on most computers, relative to where you've installed Anaconda, this will be:

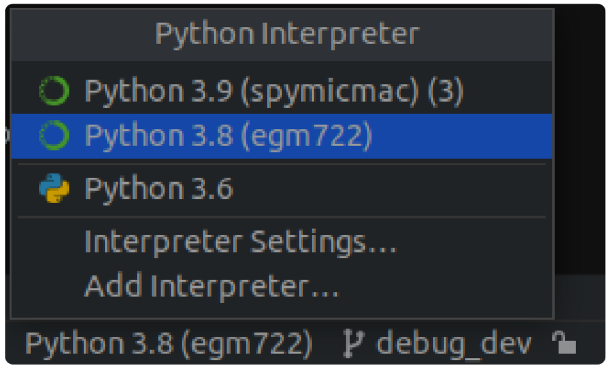
```
~/Anaconda3/bin/conda
```

Skip to main content

(or, if you're on Windows, `conda.exe`). You can also click the **Make available to all projects** checkbox, which will help you more easily add this interpreter to other new projects. Click **OK** to return to the previous window.

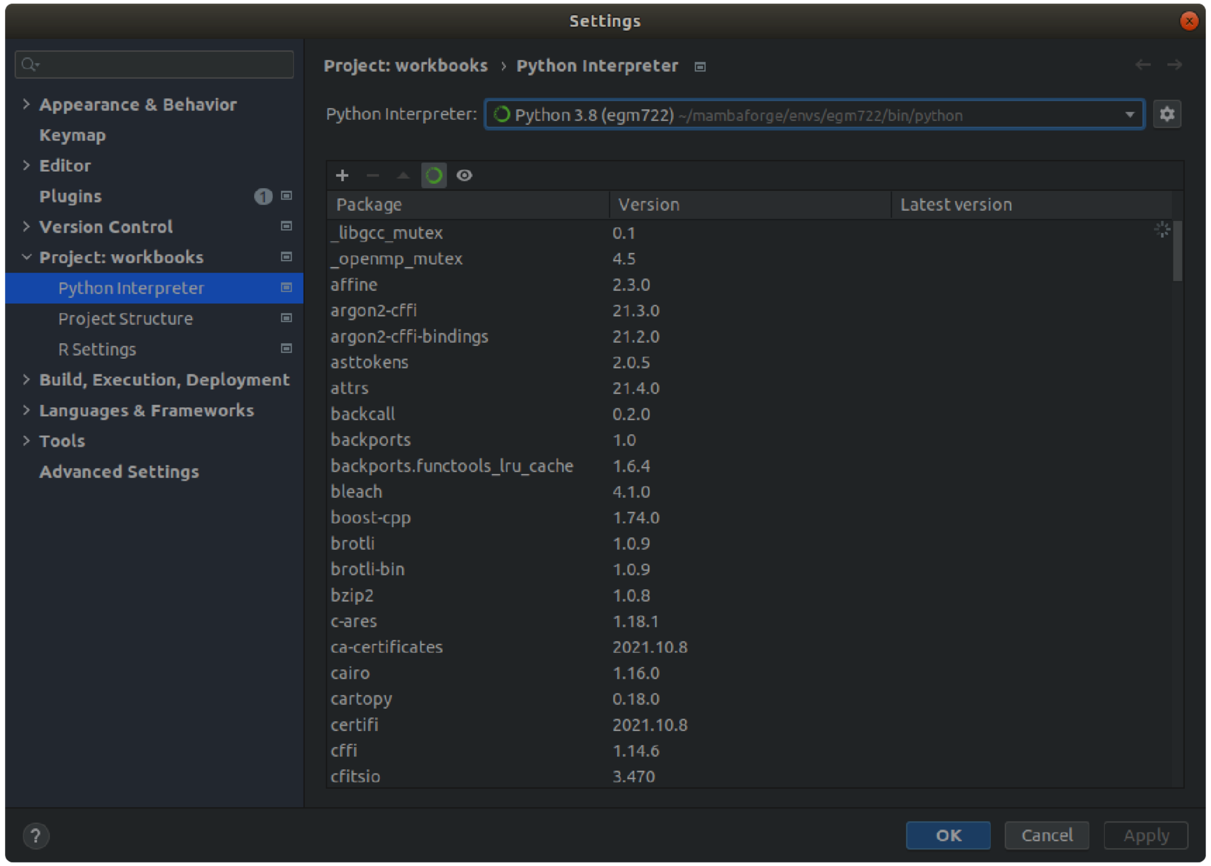
changing the interpreter

To change which python interpreter your project uses, open the Interpreter menu by clicking on it in the lower right-hand side of the window:



Select **Interpreter Settings**, which will open the Interpreter Settings window:

Skip to main content

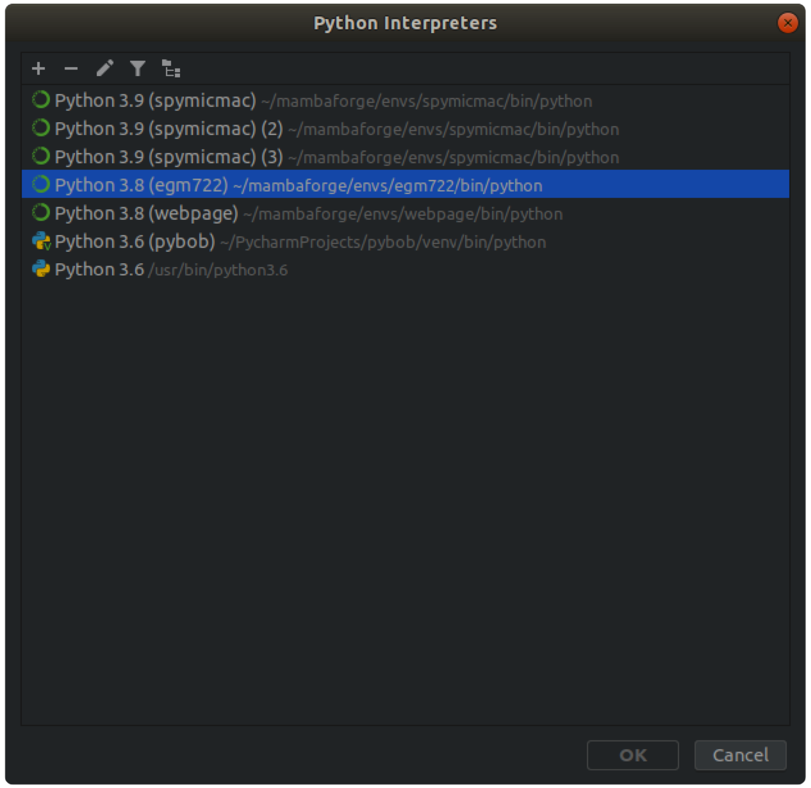


Python Interpreter: Python 3.8 (egm722) ~/mambaforge/envs/egm722/bin/python

Package	Version	Latest version
_libgcc_mutex	0.1	
_openmp_mutex	4.5	
affine	2.3.0	
argon2-cffi	21.3.0	
argon2-cffi-bindings	21.2.0	
asttokens	2.0.5	
attrs	21.4.0	
backcall	0.2.0	
backports	1.0	
backports.functools_lru_cache	1.6.4	
bleach	4.1.0	
boost-cpp	1.74.0	
brotli	1.0.9	
brotli-bin	1.0.9	
bzip2	1.0.8	
c-ares	1.18.1	
ca-certificates	2021.10.8	
cairo	1.16.0	
cartopy	0.18.0	
certifi	2021.10.8	
cffi	1.14.6	
cfitsio	3.470	

To change the interpreter, you can select a different one from the drop menu, or by clicking the gear icon and selecting **Show all**. This will bring up the following window (note that your list will not be the same as this one):

Skip to main content



Python 3.9 (spymicmac) ~/mambaforge/envs/spymicmac/bin/python
Python 3.9 (spymicmac) (2) ~/mambaforge/envs/spymicmac/bin/python
Python 3.9 (spymicmac) (3) ~/mambaforge/envs/spymicmac/bin/python
Python 3.8 (egm722) ~/mambaforge/envs/egm722/bin/python
Python 3.8 (webpage) ~/mambaforge/envs/webpage/bin/python
Python 3.6 (pybob) ~/PycharmProjects/pybob/venv/bin/python
Python 3.6 /usr/bin/python3.6

Select the desired interpreter from this list. If you don't see it there, you'll need to add it using the steps outlined above.

notes and references

[1] Assuming that you have followed the setup steps in order, that is.

[2] The `~` symbol is used to denote the "home" directory for a user. On Windows, this would be `C:\Users\user`. On MacOS, this is `/Users/user`, and on linux it is `/home/user`.

[3] Other colors include red, which indicates files that are not tracked by **git**, and yellow/amber, which indicates files that are

Skip to main content

Previous
configuring jupyter

Next
installing packages with pip