Hands-on Python Foundations References

Arianne Dee

Table of Contents

Contact

Email me at
arianne.dee.studios@gmail.com

Links

Setup links

GitHub repository
https://github.com/ariannedee/python-foundations-3-weeks

Download Python
https://www.python.org/downloads/

Download PyCharm (Community)
https://www.jetbrains.com/pycharm/download/

Download Git instructions
https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

Katacoda scenarios

Playlist
https://learning.oreilly.com/playlists/030e1bff-911b-4338-8e3b-4a0a0faa6783/

Supporting videos

Introduction to Python
https://learning.oreilly.com/videos/introduction-to-python/9780135707333/

Next Level Python
https://learning.oreilly.com/videos/next-level-python/9780136904083/

Documentation

Python documentation
https://docs.python.org/3.8/library/index.html

Requests
Make HTTP requests
https://docs.python-requests.org/en/latest/

Public APIs
https://github.com/public-apis/public-apis

Environs
Use environment variables and .env files for secrets
https://github.com/sloria/environs

Pytest
Test your code
https://docs.pytest.org/en/7.0.x/

Python Anywhere
Run your code in the cloud
https://www.pythonanywhere.com/

Tutorials

Python application structure
https://realpython.com/python-application-layouts/#command-line-application-layouts

Cron jobs
https://www.tutorialspoint.com/unix_commands/crontab.htm
https://www.jcchouinard.com/python-automation-with-cron-on-mac/

Task Scheduler
https://datatofish.com/python-script-windows-scheduler/

Sending emails
https://realpython.com/python-send-email/

Video references

Week 1

Running Python – Intro to Python lesson 1.1
First code – Intro to Python lesson 1.2
Variables – Intro to Python lesson 2.2
Types – Intro to Python lesson 2.1
Strings – Intro to Python lesson 2.7
Functions – Intro to Python lesson 2.6
Conditions – Intro to Python lesson 3.2
Conditionals (if else) – Intro to Python lesson 3.3
While loops – Intro to Python lesson 4.1
For loops – Intro to Python lesson 4.4

Week 2

Lists – Intro to Python lesson 4.3
Dictionaries – Next Level Python lesson 1.3
Sets, tuples – Intro to Python lesson 5.2
Exceptions – Next Level Python lesson 1.4
Reading files – Next Level Python lesson 2.1
Writing files – Next Level Python lesson 2.2
CSV files – Next Level Python lesson 2.3
Requests – Next Level Python lesson 7.1
API requests – Next Level Python lesson 7.5
Command line – Next Level Python lesson 3.1

Week 3

Pip – Next Level Python lesson 3.2
Virtual environments – Next Level Python lesson 3.3
Set up project – Next Level Python lesson 3.5
Clone a project – Next Level Python lesson 3.6
Modules and import – Next Level Python lesson 5.1
__init__.py files – Next Level Python lesson 5.2
Namespaces and scope – Next Level Python lesson 5.3
Run code in the cloud – Next Level Python lesson 9.1
Classes: No video yet, Live training called Object-Oriented Programming in Python

Supplementary videos

More concepts

Dates and times – Next Level Python lesson 1.5

Regular expressions – Next Level Python lesson 1.6

HTML overview – Next Level Python lesson 7.2

Scraping websites – Next Level Python lesson 7.3

Git and GitHub – Next Level Python lesson 3.4

Debugging – Next Level Python lesson 6.1

Testing – Next Level Python lesson 6.2

Projects

Basics review – Next Level Python lesson 1.2

Web scraper project – Next Level Python lesson 8

Intro to data analysis – Intro to Python lesson 6

Intro to web apps – Intro to Python lesson 7

Create an Anvil web app – Next Level Python lesson 9.2

Command line: Common actions

Action	Mac/Linux	Windows
List the folders and files in current location	ls	dir
Display your current location	pwd	cd
Change directory/folder in the file system.	cd <*pathname*>	cd <*pathname*>
Move one level up (one folder) from current location	cd ..	cd ..
Copy a file to another folder	cp	copy
Move a file to another folder	mv	move
Create a new directory (folder)	mkdir	mkdir or md
Delete a file or directory	rm	del
Display the contents of a file	cat <filename>	type <filename>
Clear the window	clear	cls
Show the manual for a command	man <command>	help <command>

Command line: Using pip

1. Open a command line application

Mac/Linux: Terminal

Windows: PowerShell

In **IDE**: PyCharm, VSCode, etc

2. Figure out the correct pip command to use

```
$ pip --version
```

Tells you the pip version, but also the Python version it's for.

You might have to use **`pip3`** or **`pip3.8`** to target the correct version of Python.

If you are using a virtual environment, make sure you've activated it first.

3. Enter the command "`pip install {package_name}`"

Use the pip command from the previous step to install the package.

```
$ pip3.8 install requests
```

4. Check that installation completed successfully

```
/lib/python3.6/site-packages (from requests) (3.0.4)
Installing collected packages: requests
Successfully installed requests-2.23.0
```

5. Run your program to make sure it worked

If it didn't work, you'll get an error:

> ModuleNotFoundError: No module named 'requests'

Make sure you're running your script with the correct Python version (e.g. python3.8)

If you don't have the pip command

You can use **`python -m pip`** instead to run it as a Python module, or follow the instructions at https://pip.pypa.io/en/stable/installation/.

Command line: Using virtual environments

`$ python3.6 -m venv <folder_name>`
Can use any python version > 3.3
Folder name is usually just env

`$ source venv/bin/activate` (Unix) or **`> venv\Scripts\Activate.ps1`** (Win)
Activates virtual env

`$ which python` (Unix) or **`> Get-Command python`** (Win)
Should be **venv/bin/python** now

`$ which pip`
Should be **venv/bin/pip** now

`$ pip install <package-name>`
Installs packages into:
- **venv/lib/python/site-packages** (Unix) or
- **venv/Lib/site-packages/** (Win)

`$ deactivate`
Deactivates virtual environment

Command Line: Using git

Starting a repository

From scratch

$ git init

$ git remote add origin <repo_url>

From an existing repository

$ git clone <repo_url>

Making changes

Making a commit

$ git add <file, folder or .> (. will add all changed files to staging)

$ git commit -m “Commit message”

Pushing changes to remote repository (e.g. GitHub)

$ git push origin main (first time)

$ git push (subsequent times)

Updating repository

Pulling changes from remote repository

$ git pull origin main (first time)

$ git pull (subsequent times)

Getting information

Get status of changes (unstaged and staged)

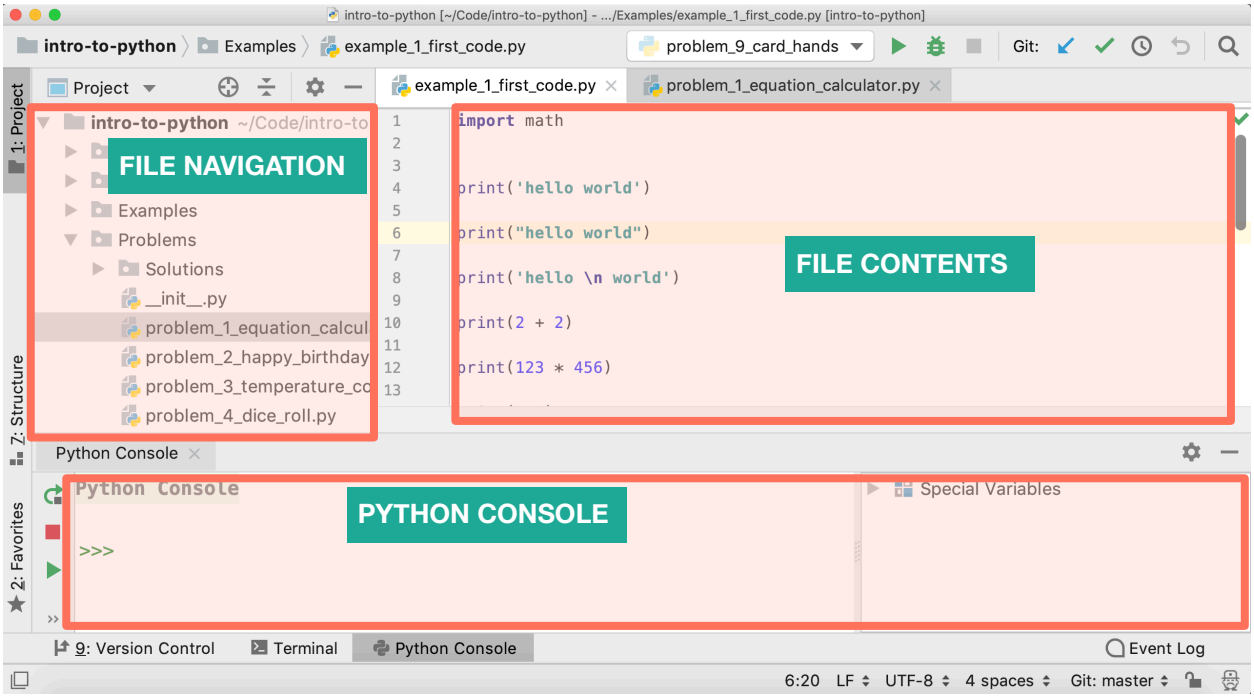
$ git status

Get history of commits

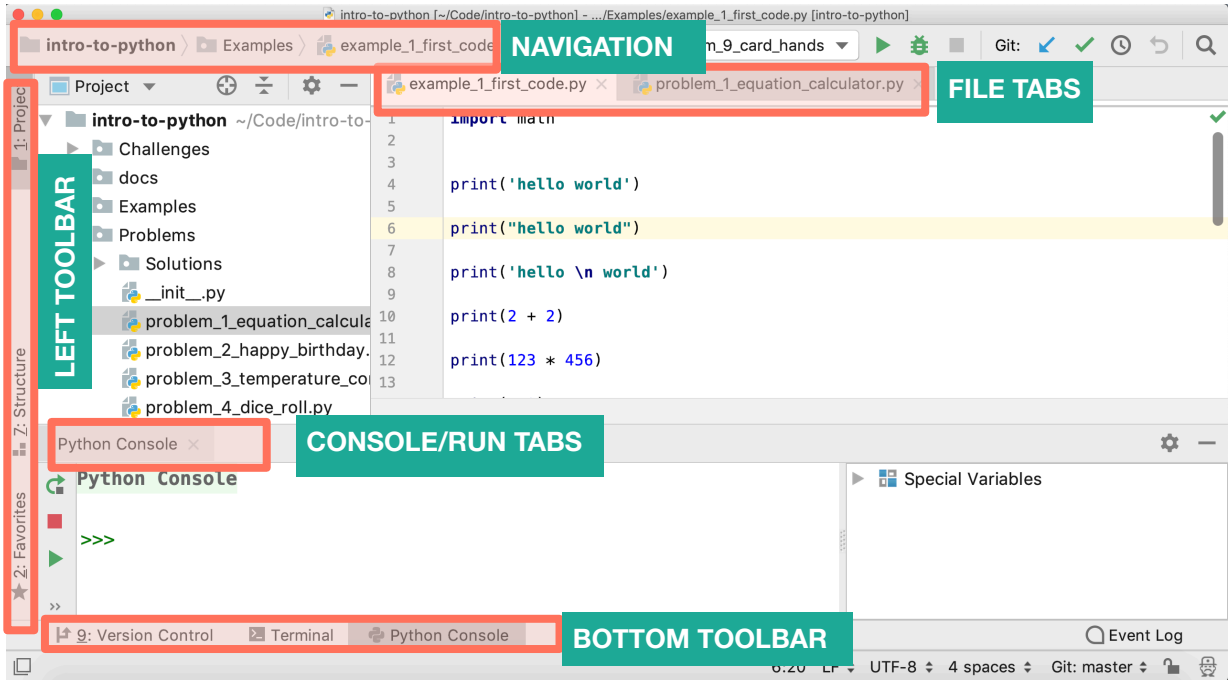
$ git log

PyCharm: Layout

Sections



Toolbars



PyCharm: Keyboard shortcuts

References

https://www.jetbrains.com/help/pycharm/mastering-keyboard-shortcuts.html
Select operating system/key-map at top

https://resources.jetbrains.com/storage/products/pycharm/docs/PyCharm_ReferenceCard.pdf
Some differences between PC/Mac/PyCharm versions

Edit/browse shortcuts

Go to Settings > Keymap

Common shortcuts

A. PyCharm

Ctrl+Shift+A	Search for keyboard shortcut
Ctrl+,	Open settings
Alt+Enter	Show error dialogue

B. Most programs

Ctrl+C	Copy	**Ctrl+V**	Paste
Ctrl+Z	Undo	**Ctrl+Shift+Z**	Redo
Ctrl+F	Find in file	**Ctrl+R**	Replace in file

C. Selecting

Shift+Arrows	Select multiple characters/lines	**Tab+Up**	Widen selection

D. New lines

Ctrl+Back	Delete entire line		
Shift+Enter	Start new line below	**Ctrl+Enter**	Start new line above

E. Navigation

Ctrl+Shift+O	Find file		
Ctrl+Shift+]	Go to next tab	**Ctrl+Shift+[**	Go to previous tab
Ctrl+Alt+←	Go to previous location		
Ctrl+B	Go to definition		

F. Change

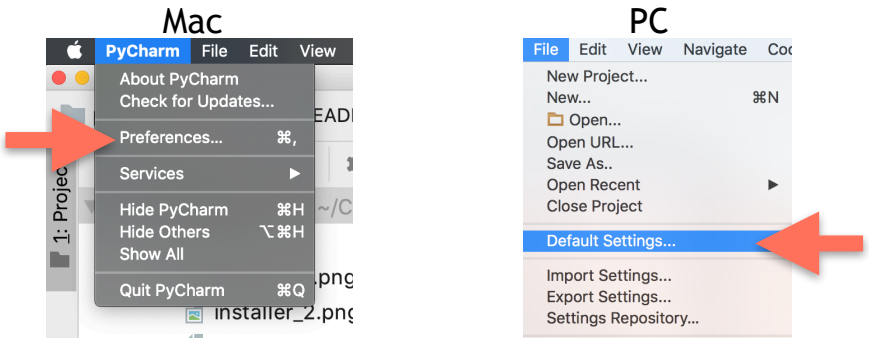
Ctrl+/	Comment / uncomment line		
Ctrl+D	Duplicate selection		
Ctrl+Shift+↑	Move line up	**Ctrl+Shift+↓**	Move line down
Ctrl+Alt+L	Reformat file		

G. Refactor

Shift+F6	Rename		
F6	Move		
Ctrl+Alt+V	Extract variable		
Ctrl+Shift+F	Find in project	**Ctrl+Shift+R**	Replace in project

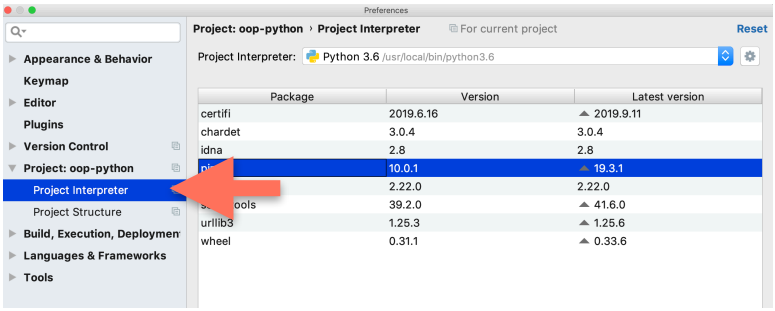
PyCharm: Setting your Python version

1. Open settings/preferences

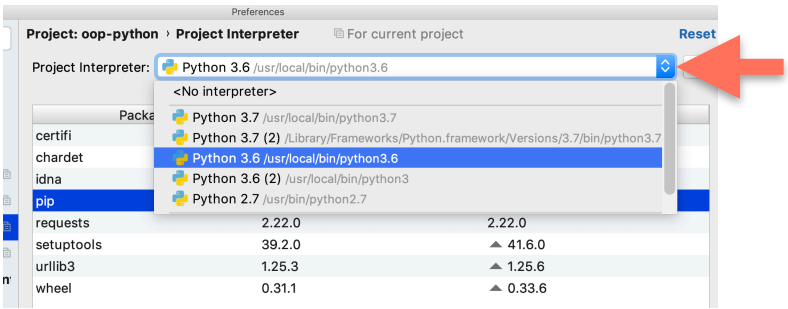


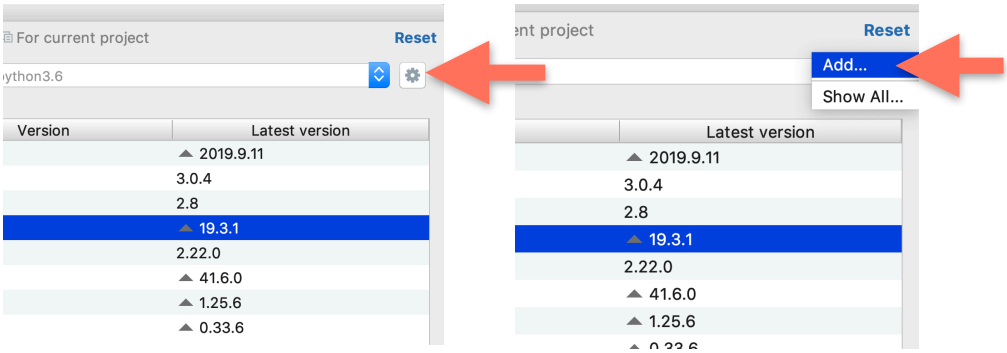
2. Navigate to Project Interpreter

3a. Select existing interpreter

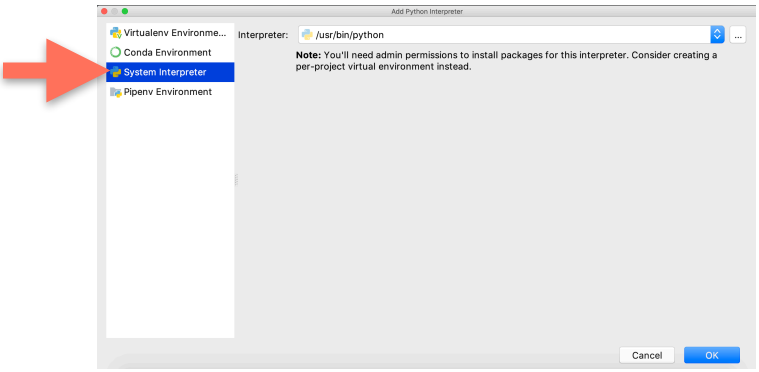


3b. Add new interpreter

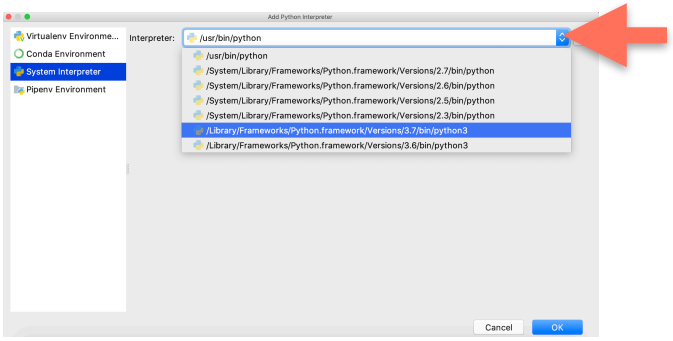


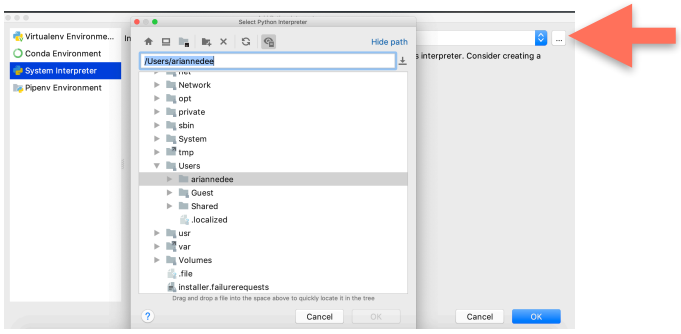


4. Add a system interpreter



5. Select existing system interpreter or find it





If you're not sure where to find it

Windows (look for python.exe)

- C:\Python38
- C:\Program Files\Python38
- C:\Users\username\AppData\Local\Programs\Python\Python38-XX

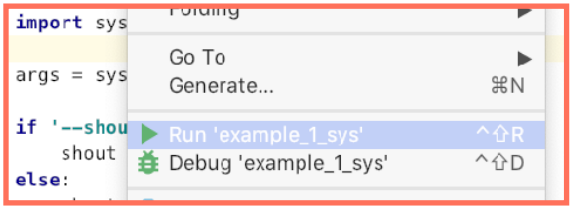
Mac (look for python3.8 or python3)

- /usr/local/bin/
- /Library/Frameworks/Python.framework/Versions/3.8/bin/
- /usr/local/Cellar/python/3.8.X_X/bin/
- /Users/username/anaconda/bin/
- /anaconda3/bin/

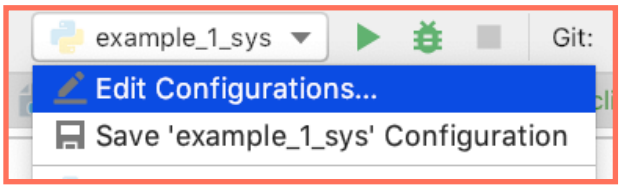
Linux (look for python3.8 or python3)

- /usr/bin/
- /usr/local/bin

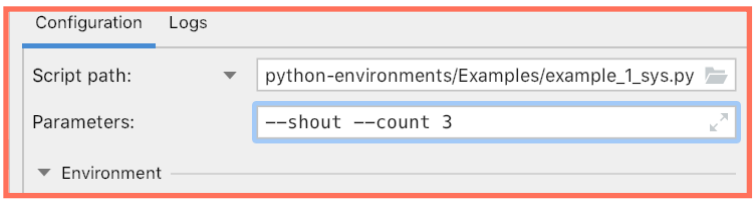
PyCharm: Set “Run” configurations



1. Right click and run the file



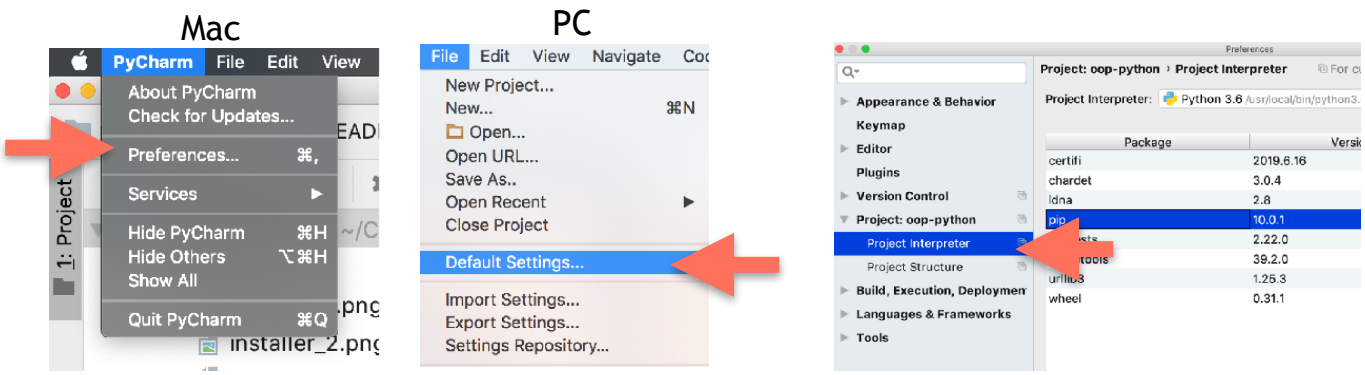
2. In top bar, click dropdown and select Edit Configurations…



3. Add arguments to the “Parameters” field

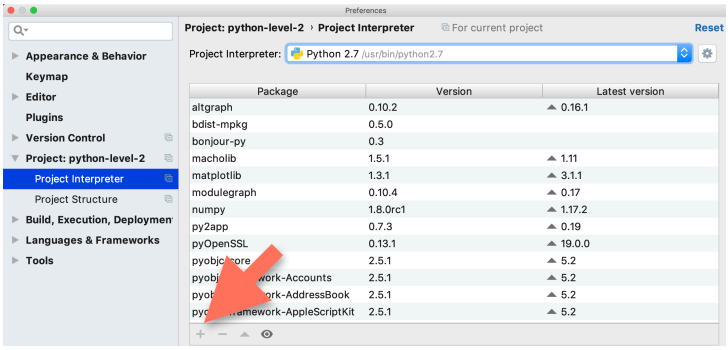
PyCharm: Install packages

1. Open settings/preferences and go to Project Interpreter

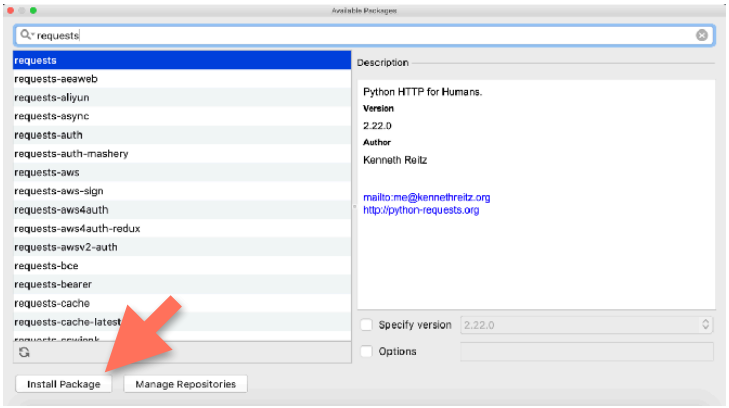


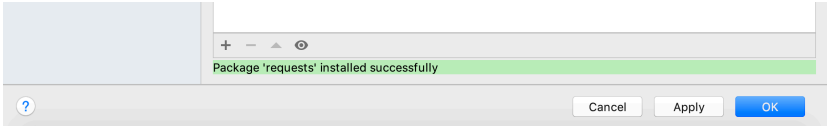
2. Navigate to add package view

3. Search for package, select, and install



4. Verify it worked





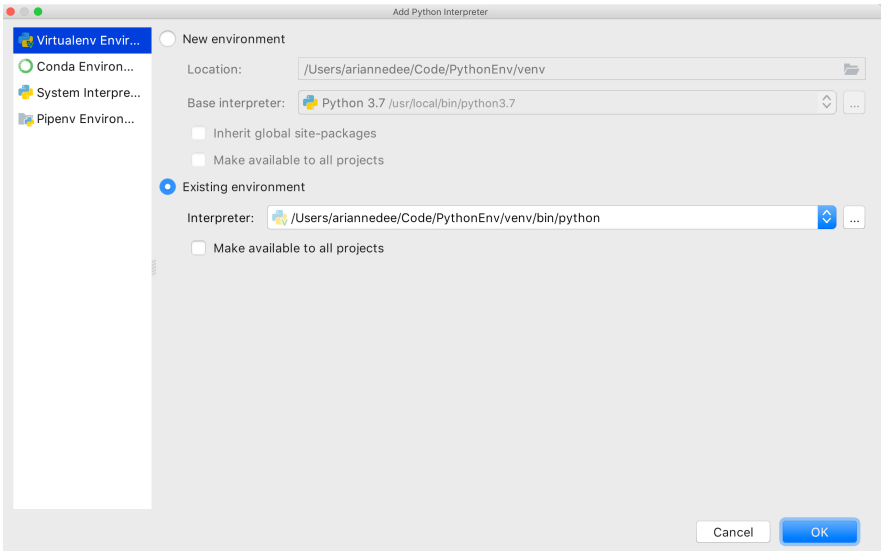
PyCharm: Configure a virtual environment

Follow steps 1 - 3b of PyCharm: Setting your Python version on page 14.

4. Select your existing environment

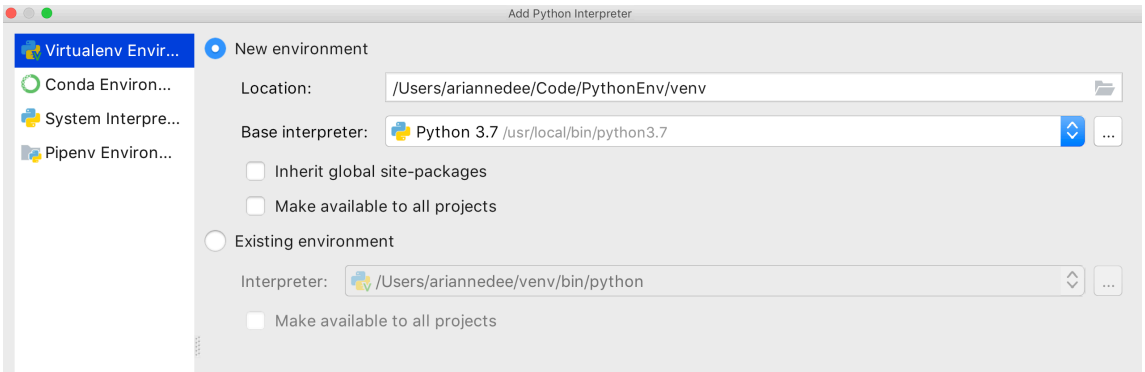
If you created it in the command line with **`$ python3.8 -m venv venv`**

Select **venv/bin/python** (Unix) or **venv\Scripts\python.exe** (Win)



5. Or create a new one

6. Close any open terminals and re-open them so that the



virtual environment is activated

PyCharm: Git integration