

REMINDER: videos on linear algebra

The Essence of Linear Algebra series ([3blue1brown](#)) linked below is an excellent resource for reviewing (or learning) basic concepts of linear algebra. The most important concepts to review are vectors (Chapter 1-2), linear transformations and matrices (Chapter 3), matrix multiplication (Chapter 4), non-square matrices as transformations between dimensions (Chapter 8), and dot products (Chapter 9).

Please review these over the next few weeks.

https://www.youtube.com/playlist?list=PLZHQBObOWTQDPD3MizzM2xVFitgF8hE_ab

REMINDER: help sessions

Mondays 3:50-5:00
this room (or WH 113)

except for this week, which will be Wed after class

REMINDER: Piazza

- piazza.com/vanderbilt/fall2022/psy42196219

(part of) HOMEWORK 1:

Create a Post and Respond to a Post on Piazza
(under the "welcome" tag) - share something about yourself (your interests and hobbies, or your future plans, or what you hope to learn in the class, or really anything you're willing to share)

I shared a (long) post with a bit about myself - after you create a post for yourself, you can respond to the one I wrote or respond to another post

Installing and Configuring Python

installing Python

- if you have never installed Python before, go to <https://www.anaconda.com>, download, and install



Data science technology for
a better world.

Anaconda offers the easiest way to perform Python/R data science and machine learning on a single machine. Start working with thousands of open-source packages and libraries today.

[Download](#)

For MacOS

Python 3.9 • 64-Bit Graphical Installer • 591 MB

[Get Additional Installers](#)



if you already have Python

- if you have Python, check that you have at least version 3.8.x - within Python, run the following code snippet:

```
import sys
```

```
print(sys.version_info)
```

- if your version of Python is old, the simplest thing to do is download and install Anaconda (latest version)

if you have 3.9.x you might also have 3.8.x installed

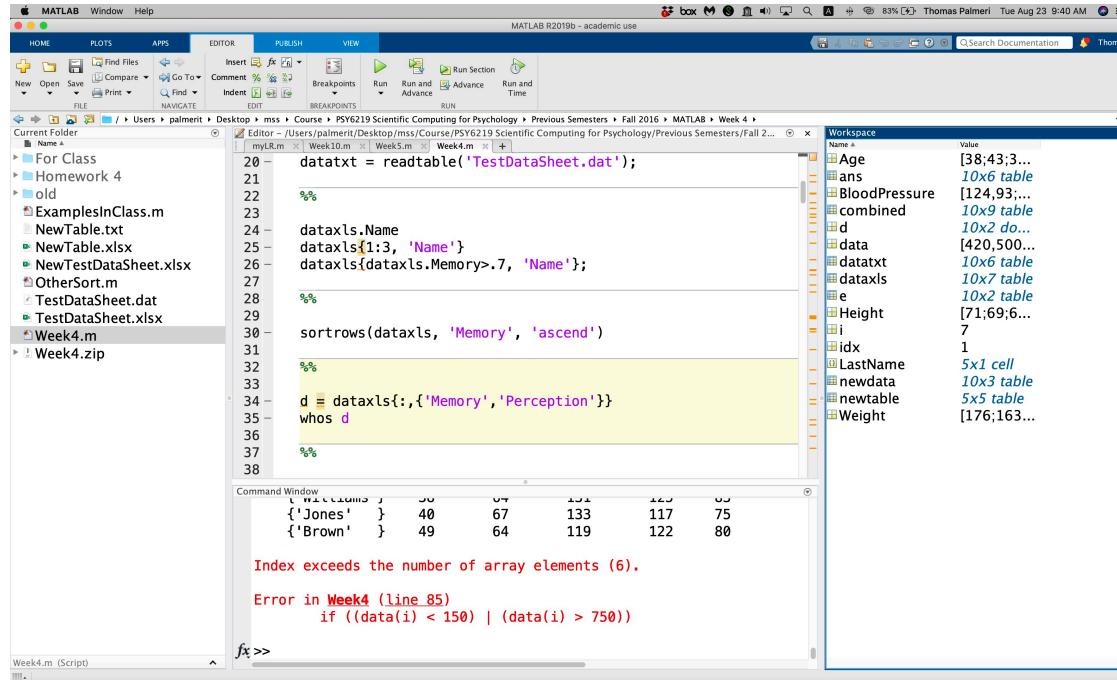
if you already have Python

- you don't have to use the Anaconda build of Python, or download the full (bloated) Anaconda, but that is the (simplest) approach we describe in class

some might need to use a different installation for various reasons (Jason can help if it is necessary to get Python working on your computer, or you can explore other options on your own)

navigating the pieces of your installation

how Python differs from Matlab



Mathworks is the company that owns Matlab. When you buy Matlab, you get the programming language, any toolboxes you purchase along with base Matlab, and the integrated development environment (IDE).

install "Matlab" and you get the language, the toolboxes, and the IDE.

what does a Python installation look like?

Python the language

```
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image

def myfilter(x, alpha=1., beta=2., gam=.5, lam=.1):
    F = 1 - np.exp(-(x/alpha)**beta)
    psi = gam + (1. - lam - gam)*F
    return (psi)

alpha = 100
beta = 3
gam = 0.48
lam = 0.48

x = np.arange(0, 255, .1)
psi = myfilter(x, alpha, beta, gam, lam)
plt.plot(x, psi, 'r-');
plt.ylim((0,1))
```



Python is open source, freely usable and distributable, even for commercial use - Python's license is administered by the Python Software Foundation

you can download Python (for free) directly from [python.org](https://www.python.org) (all releases going back to Python 1.1)

what does a Python installation look like?

Python the language

```
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image

def myfilter(x, alpha=1., beta=2., gam=.5, lam=.1):
    F = 1 - np.exp(-(x/alpha)**beta)
    psi = gam + (1. - lam - gam)*F
    return (psi)

alpha = 100
beta = 3
gam = 0.48
lam = 0.48

x = np.arange(0, 255, .1)
psi = myfilter(x, alpha, beta, gam, lam)
plt.plot(x, psi, 'r-');
plt.ylim((0,1))
```



Python is open source, freely usable and distributable, even for commercial use - Python's license is administered by the Python Software Foundation

you can download Python (for free) directly from [python.org](https://www.python.org) (all releases going back to Python 1.1)

with the most basic Python installation, you edit Python .py files (text files) using a basic text editor, or a text editor designed for editing code (Emacs, Vim, BBedit, and many others)

run Python code from the command line / terminal

```
Last login: Sun Aug 21 15:41:34 on ttys001
```

```
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
Tom-MacBook-Pro-2020:~ palmerit$ python mycode.py
```

what does a Python installation look like?

Python the language

```
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image

def myfilter(x, alpha=1., beta=2., gam=.5, lam=.1):
    F = 1 - np.exp(-(x/alpha)**beta)
    psi = gam + (1. - lam - gam)*F
    return (psi)

alpha = 100
beta = 3
gam = 0.48
lam = 0.48

x = np.arange(0, 255, .1)
psi = myfilter(x, alpha, beta, gam, lam)
plt.plot(x, psi, 'r-');
plt.ylim((0,1))
```

Python packages / modules

(a package is collection of modules)
(like toolboxes in Matlab, libraries in C)

numpy, scipy, matplotlib,
seaborn, pandas, pillow,
scikit-learn, psychopy, keras

what does a Python installation look like?

Python the language

```
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image

def myfilter(x, alpha=1., beta=2., gam=.5, lam=.1):
    F = 1 - np.exp(-(x/alpha)**beta)
    psi = gam + (1. - lam - gam)*F
    return (psi)

alpha = 100
beta = 3
gam = 0.48
lam = 0.48

x = np.arange(0, 255, .1)
psi = myfilter(x, alpha, beta, gam, lam)
plt.plot(x, psi, 'r-');
plt.ylim((0,1))
```

Python packages / modules

(a package is collection of modules)
(like toolboxes in Matlab, libraries in C)

numpy, scipy, matplotlib,
seaborn, pandas, pillow,
scikit-learn, psychopy, keras



you can download packages from pypi.org
using pip from the command line / terminal

what does a Python installation look like?



Anaconda is a company that provides a distribution of Python and 100s of packages (instead of python.org and pypi.org)

Python the language

+

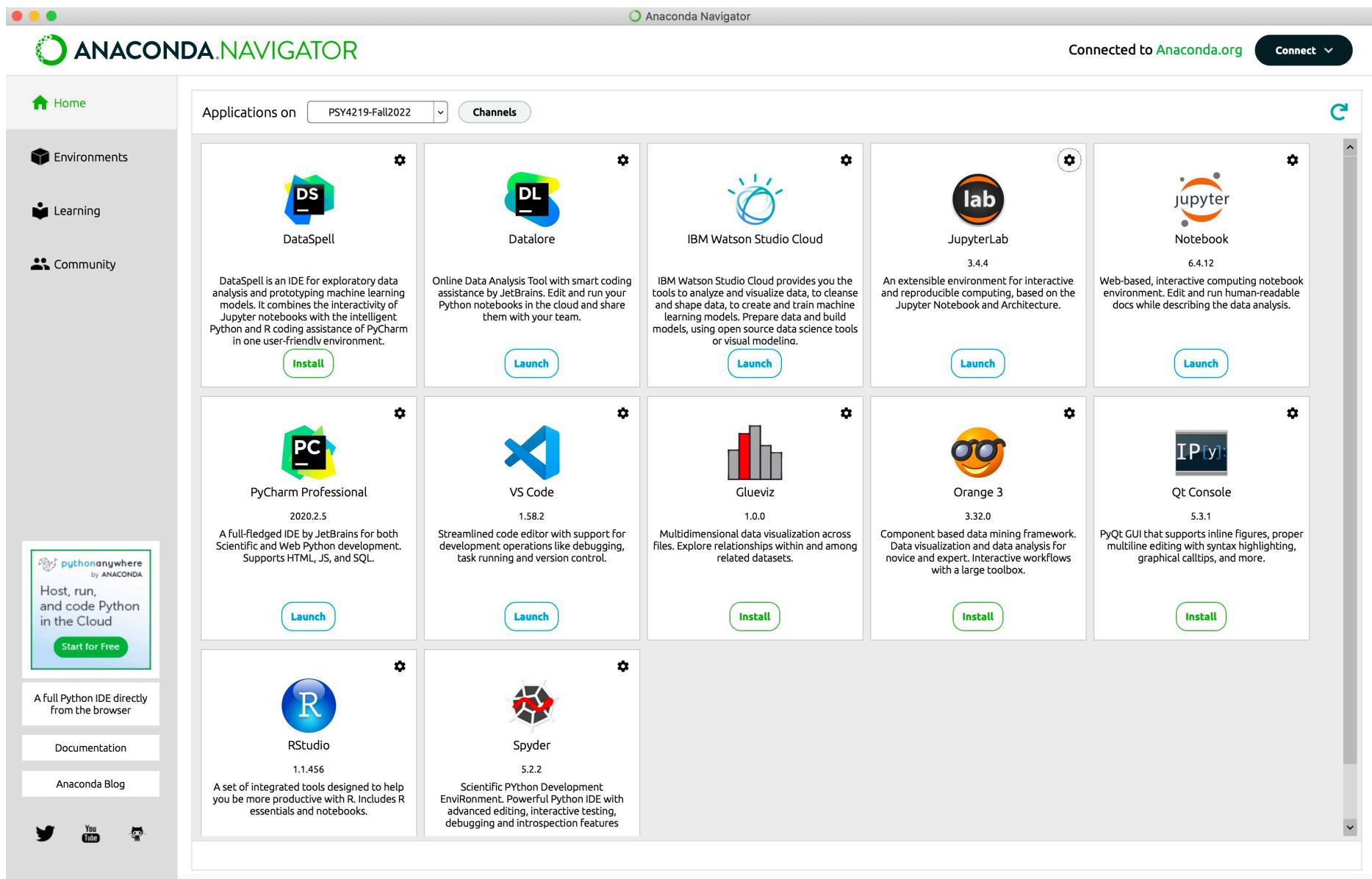
Python packages / modules

\$\$\$ for businesses

free for education

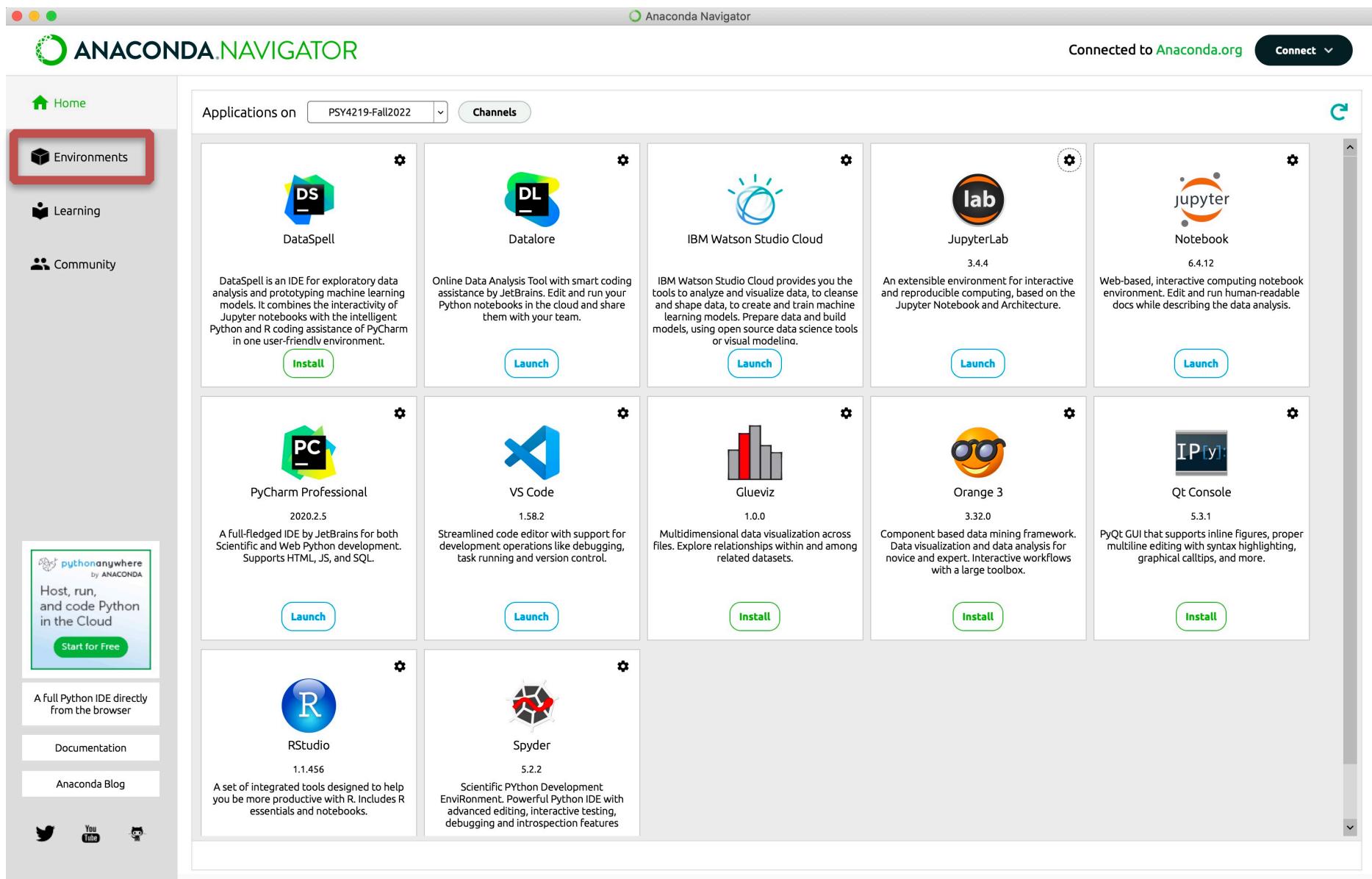
what does a Python installation look like?

the Anaconda installation comes with Anaconda Navigator, which lets you launch applications, and manage packages and environments



what does a Python installation look like?

the Anaconda installation comes with Anaconda Navigator, which lets you launch applications, and manage packages and environments



what does a Python installation look like?

in this class, you need Python 3.8.x to use the psychopy package, and that package depends on particular versions of other packages

in computational neuroscience, you may need a different version of Python to use tensorflow, which in turn depends on other versions of various packages

in your lab research, you use an old analysis package that depends on an old different version of Python and old versions of other packages

what does a Python installation look like?

in this class, you need Python 3.8.x to use the psychopy package, and that package depends on particular versions of other packages

in computational neuroscience, you may need a different version of Python to use tensorflow, which in turn depends on other versions of various packages

in your lab research, you use an old analysis package that depends on an old different version of Python and old versions of other packages

tools to create virtual environment allow you to have all of these different configurations of Python at once

virtualenv and **conda** are two popular tools

Anaconda Navigator creates conda environments and lets you easily manipulate them and use them

what does a Python installation look like?

in this class, you need Python 3.8.x to use the psychopy package, and that package depends on particular versions of other packages

in computational neuroscience, you may need a different version of Python to use tensorflow, which in turn depends on other versions of various packages

in your lab research, you use an old analysis package that depends on an old different version of Python and old versions of other packages

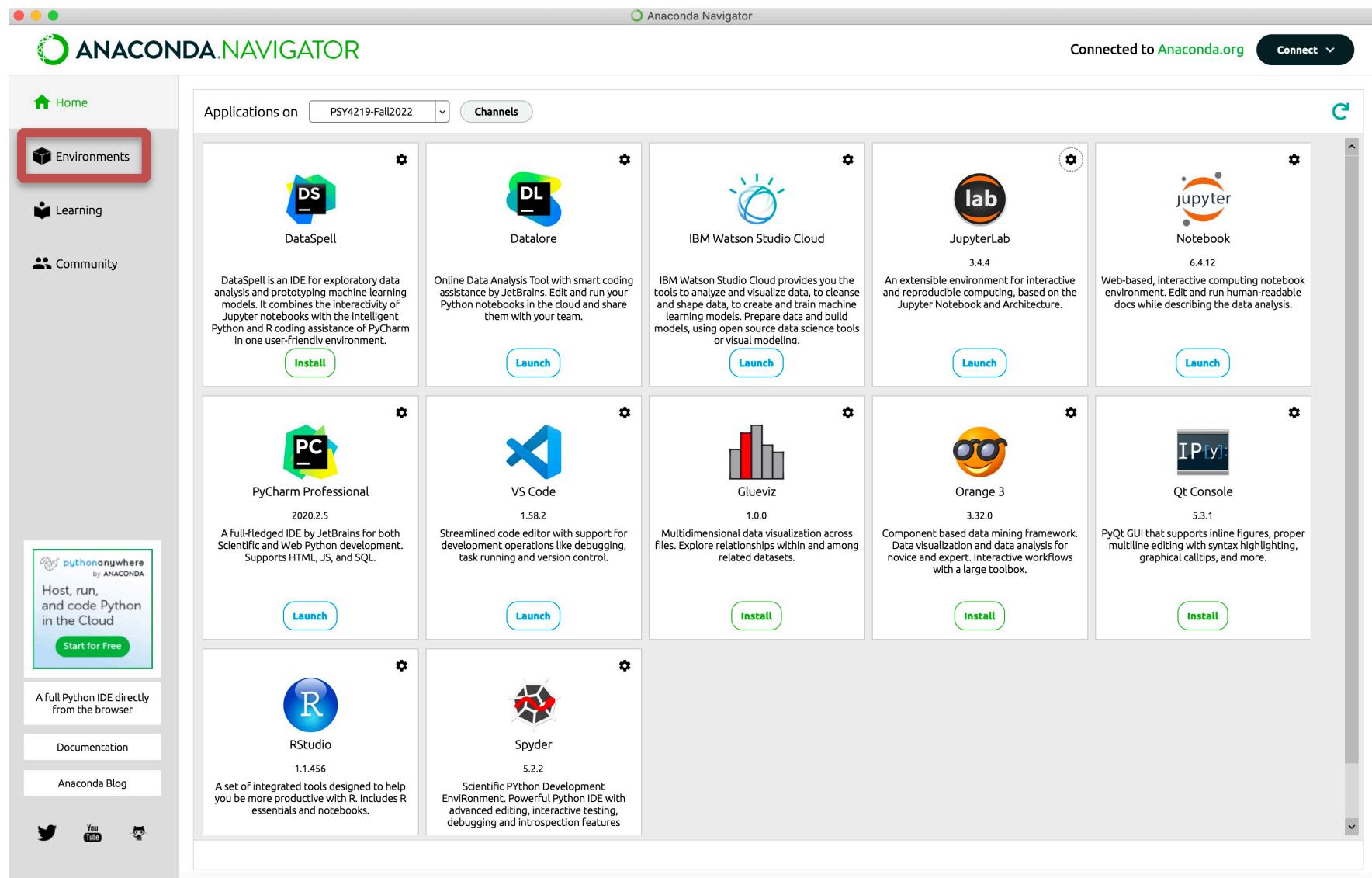
(more extreme versions are docker or singularity containers, which encapsulate a configured version of an entire operating system, not just Python)

tools to create virtual environment allow you to have all of these different configurations of Python at once

virtualenv and **conda** are two popular tools

Anaconda Navigator creates conda environments and lets you easily manipulate them and use them

virtual environments in Anaconda Navigator



<https://docs.anaconda.com/anaconda/navigator/>

virtual environments in Anaconda Navigator

The screenshot shows the Anaconda Navigator interface. On the left is a sidebar with icons for Home, Environments (selected), Learning, and Community. A promotional box for PythonAnywhere is also present. The main area displays a list of installed packages for the 'PSY4219-Fall2022' environment. The packages listed include ca-certificates, cairo, certifi, cffi, charset-normalizer, colorama, configobj, configparser, cryptography, cyclone, debugpy, decorator, defusedxml, dukpy, entrypoints, esprima, esprima-python, et-xmlfile, and jupyter. The 'Create' button at the bottom left is highlighted with a red box.

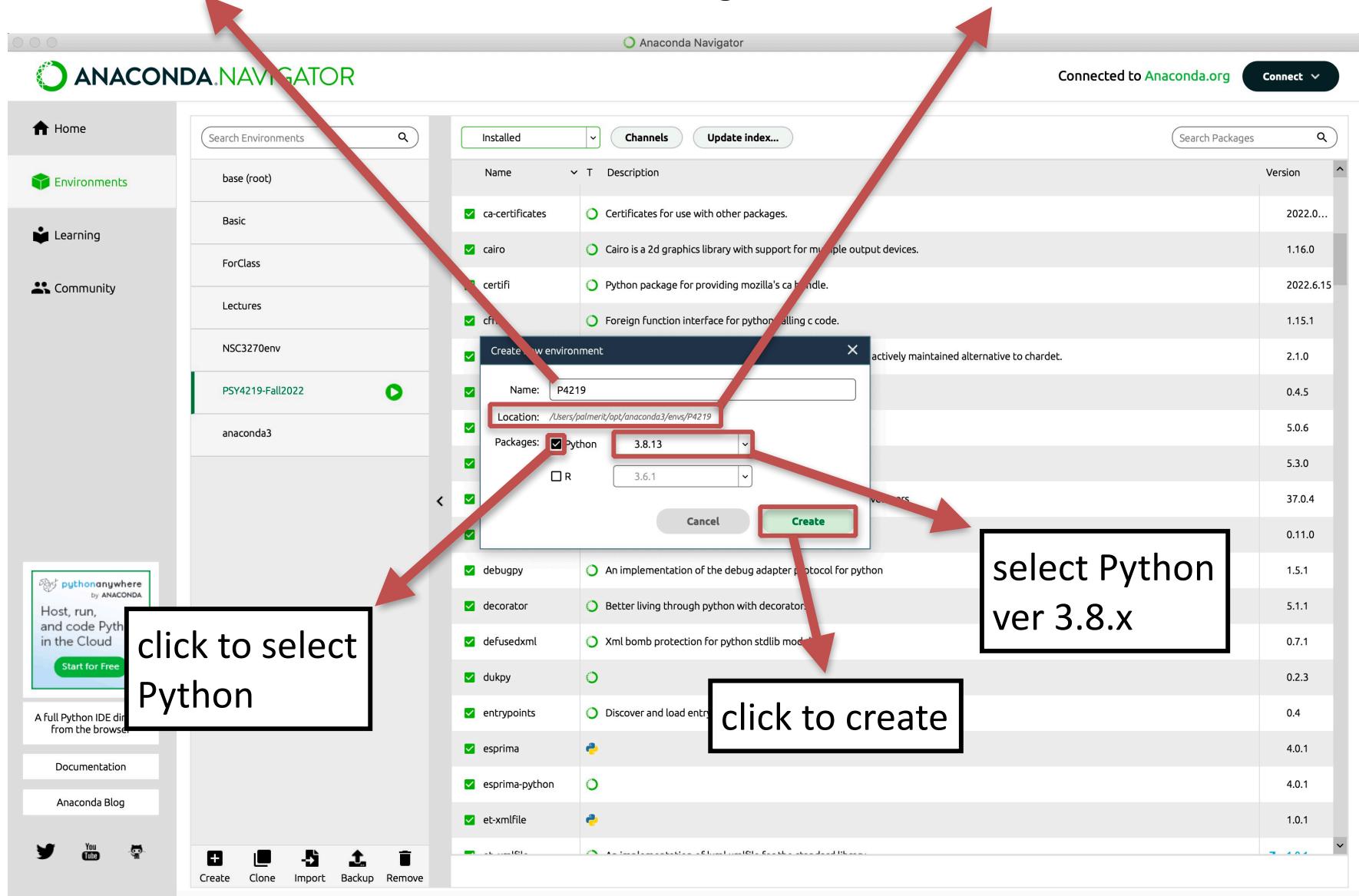
Name	Description	Version
ca-certificates	Certificates for use with other packages.	2022.0...
cairo	Cairo is a 2d graphics library with support for multiple output devices.	1.16.0
certifi	Python package for providing mozilla's ca bundle.	2022.6.15
cffi	Foreign function interface for python calling c code.	1.15.1
charset-normalizer	The real first universal charset detector. open, modern and actively maintained alternative to chardet.	2.1.0
colorama	Cross-platform colored terminal text	0.4.5
configobj	Config file reading, writing and validation.	5.0.6
configparser	Updated configparser from python 3.8 for python 2.6+.	5.3.0
cryptography	Provides cryptographic recipes and primitives to python developers	37.0.4
cyclone	Composable style cycles.	0.11.0
debugpy	An implementation of the debug adapter protocol for python	1.5.1
decorator	Better living through python with decorators.	5.1.1
defusedxml	Xml bomb protection for python stdlib modules	0.7.1
dukpy		0.2.3
entrypoints	Discover and load entry points from installed packages.	0.4
esprima		4.0.1
esprima-python		4.0.1
et-xmlfile		1.0.1

<https://docs.anaconda.com/anaconda/navigator/>

virtual environments in Anaconda Navigator

name of the environment

write down the location of the environment
e.g., /Users/palmerit/opt/anaconda3/envs/P4219



virtual environments in Anaconda Navigator

new environment is created

The screenshot shows the Anaconda Navigator interface. On the left, the 'Environments' sidebar lists several environments: 'base (root)', 'Basic', 'ForClass', 'Lectures', 'NSC 270env', 'P4219' (which is highlighted with a red box), 'PSY4219-Fall2022', and 'anaconda3'. On the right, the main window displays the 'Installed' tab of the package manager, showing a list of 15 packages available. A red arrow points from the text 'new environment is created' to the 'P4219' environment in the sidebar. Another red arrow points from the text 'very few packages installed' to the list of packages in the main window.

Name	Description	Version
ca-certificates	Certificates for use with other packages.	2022.0...
certifi	Python package for providing mozilla's ca bundle.	2022.6.15
libcxx	Llvm c++ standard library	12.0.0
libffi	A portable foreign function interface library.	3.3
ncurses	Library for text-based user interfaces	6.3
openssl	Openssl is an open-source implementation of the ssl and tls protocols	1.1.1q
pip	Pypy recommended tool for installing python packages	22.1.2
python	General purpose programming language	3.8.13
readline	Library for editing command lines as they are typed in	8.1.2
setuptools	Download, build, install, upgrade, and uninstall python packages	63.4.1
sqlite	Implements a self-contained, zero-configuration, sql database engine	3.39.2
tk	A dynamic programming language with gui support. bundles tcl and tk.	8.6.12
wheel	A built-package format for python.	0.37.1
xz	Data compression software with high compression ratio	5.2.5
zlib	Massively spiffy yet delicately unobtrusive compression library	1.2.12

<https://docs.anaconda.com/anaconda/navigator/>

virtual environments in Anaconda Navigator

select "All" to see all available packages

The screenshot shows the Anaconda Navigator interface. On the left is a sidebar with icons for Home, Environments, Learning, and Community. A central panel displays a list of environments: base (root), Basic, ForClass, Lectures, NSC3270env, P4219, PSY4219-Fall2022, and anaconda3. Below these is a section for the P4219 environment, which is currently selected. The main content area shows a table of packages under the 'Installed' tab. A red arrow points to the 'All' option in a dropdown menu that appears over the table header. The table has columns for Name, Description, and Version. The packages listed include libffi, ncurses, openssl, pip, python, readline, setuptools, sqlite, tk, wheel, xz, and zlib.

Name	Description	Version
libffi	A portable foreign function interface library.	3.3
ncurses	Library for text-based user interfaces	6.3
openssl	Openssl is an open-source implementation of the ssl and tls protocols	1.1.1q
pip	Pypy recommended tool for installing python packages	22.1.2
python	General purpose programming language	3.8.13
readline	Library for editing command lines as they are typed in	8.1.2
setuptools	Download, build, install, upgrade, and uninstall python packages	63.4.1
sqlite	Implements a self-contained, zero-configuration, sql database engine	3.39.2
tk	A dynamic programming language with gui support. bundles tcl and tk.	8.6.12
wheel	A built-package format for python.	0.37.1
xz	Data compression software with high compression ratio	5.2.5
zlib	Massively spiffy yet delicately unobtrusive compression library	1.2.12

<https://docs.anaconda.com/anaconda/navigator/>

virtual environments in Anaconda Navigator

search for packages to install

A screenshot of the Anaconda Navigator interface. On the left is a sidebar with icons for Home, Environments, Learning, and Community. A central search bar at the top has 'Search Environments' and a magnifying glass icon. Below it is a dropdown menu set to 'All', a 'Channels' button, and an 'Update index...' button. To the right is a large table of package information. The table has columns for Name, Description, and Version. A red arrow points from the text 'search for packages to install' to the 'Search Packages' input field at the top right of the table area. A black callout box is overlaid on the table, containing the text: 'you will need to install the following:
numpy
scipy
matplotlib
seaborn
pandas
scikit-learn
(maybe more)'.

Name	Description	Version
_anaconda_depends	Simplifies package management and deployment of anaconda	5.3.1
_go_select	The golang select package.	2.3.0
ForClass		0.1.0
Lectures		3.4.3
NSC3270env		0.1
P4219		1.0
PSY4219-Fall2022	on, r, java, scala, c++ and more. runs on single machine,	0.0.40
anaconda3	on, r, java, scala, c++ and more. runs on single machine,	2.0
		0.1
		1.0.0
		0.1
		2.0
		0.1
		1.0.0
		0.1
		2.0
		0.1
		0.0.2
		0.0.3
		2.3.0
abseil-cpp	Abseil common libraries (c++)	202111...
absl-py	Abseil python common libraries, see https://github.com/abseil/abseil-py.	0.9.0
access	Classical and novel measures of spatial accessibility to services	1.1.3
acl-amzn2-aarch64		2.2.51
adal	The adal for python library makes it easy for python application to authenticate to azure active directory (aad) in order to access aad protected web resources.	1.2.7

<https://docs.anaconda.com/anaconda/navigator/>

virtual environments in Anaconda Navigator

numpy (and other things) shows up

e.g., type in numpy

The screenshot shows the Anaconda Navigator interface. On the left, there's a sidebar with icons for Home, Environments (selected), Learning, and Community. Below that is a box for PythonAnywhere. The main area has a search bar at the top with 'Search Environments' and a dropdown set to 'All'. A red arrow points from the text 'numpy (and other things) shows up' to the search bar. To the right of the search bar is a 'Connected to Anaconda.org' status and a 'Connect' button. Below the search bar is a table of packages. A red box highlights the 'numpy' row, and a red arrow points from the text 'e.g., type in numpy' to this highlighted row. The table includes columns for Name, Description, and Version. The 'numpy' row is version 1.9.3 and is described as 'Array processing for numbers, strings, records, and objects.'

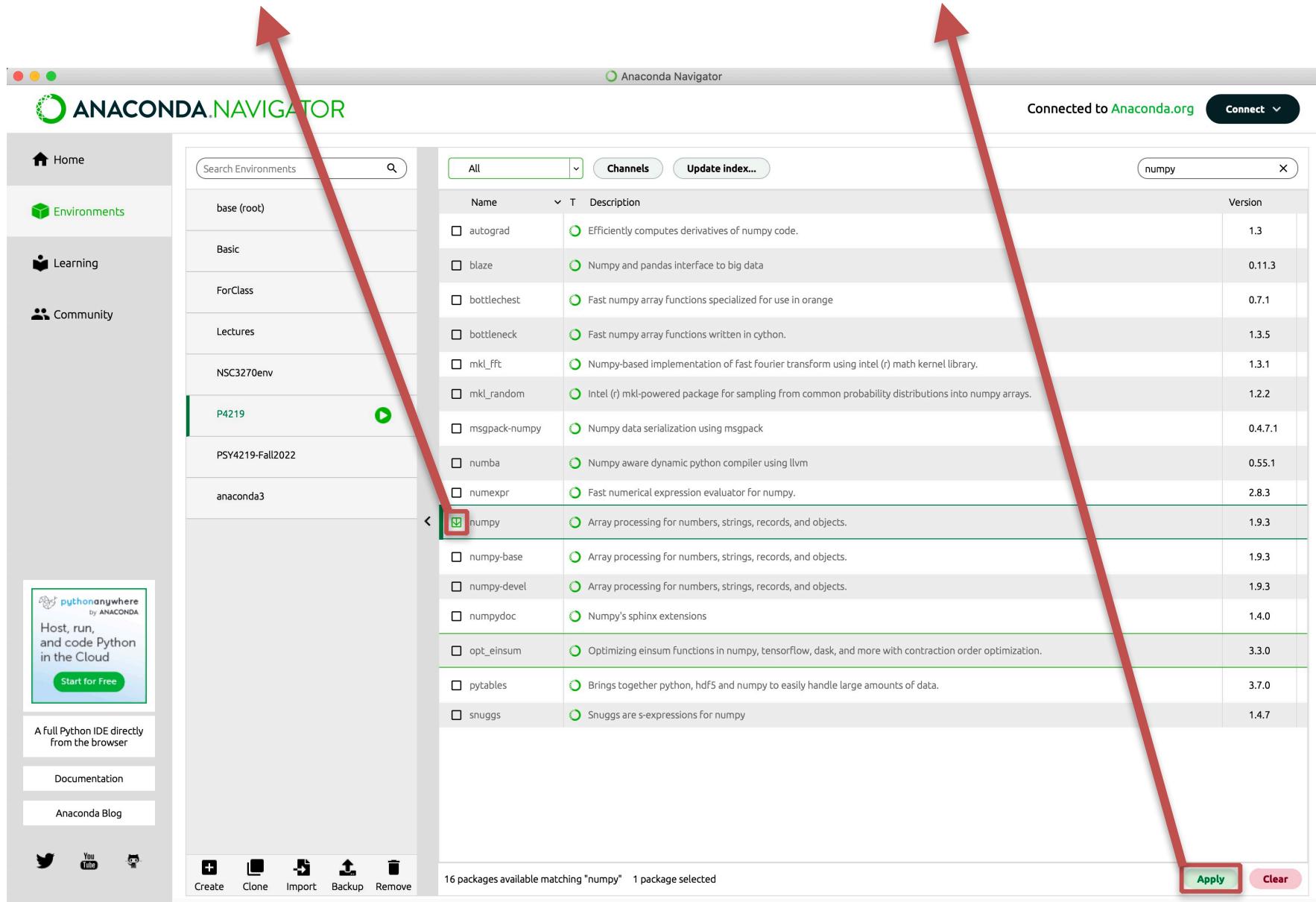
Name	Description	Version
autograd	Efficiently computes derivatives of numpy code.	1.3
blaze	Numpy and pandas interface to big data	0.11.3
bottlechest	Fast numpy array functions specialized for use in orange	0.7.1
bottleneck	Fast numpy array functions written in cython.	1.3.5
mkl_fft	Numpy-based implementation of fast fourier transform using intel (r) math kernel library.	1.3.1
mkl_random	Intel (r) mkl-powered package for sampling from common probability distributions into numpy arrays.	1.2.2
msgpack-numpy	Numpy data serialization using msgpack	0.4.7.1
numba	Numpy aware dynamic python compiler using llvm	0.55.1
numexpr	Fast numerical expression evaluator for numpy.	2.8.3
numpy	Array processing for numbers, strings, records, and objects.	1.9.3
numpy-base	Array processing for numbers, strings, records, and objects.	1.9.3
numpy-devel	Array processing for numbers, strings, records, and objects.	1.9.3
numpydoc	Numpy's sphinx extensions	1.4.0
opt_einsum	Optimizing einsum functions in numpy, tensorflow, dask, and more with contraction order optimization.	3.3.0
pytables	Brings together python, hdf5 and numpy to easily handle large amounts of data.	3.7.0
snuggs	Snuggs are s-expressions for numpy	1.4.7

<https://docs.anaconda.com/anaconda/navigator/>

virtual environments in Anaconda Navigator

click on numpy (and only numpy)

and click "Apply"

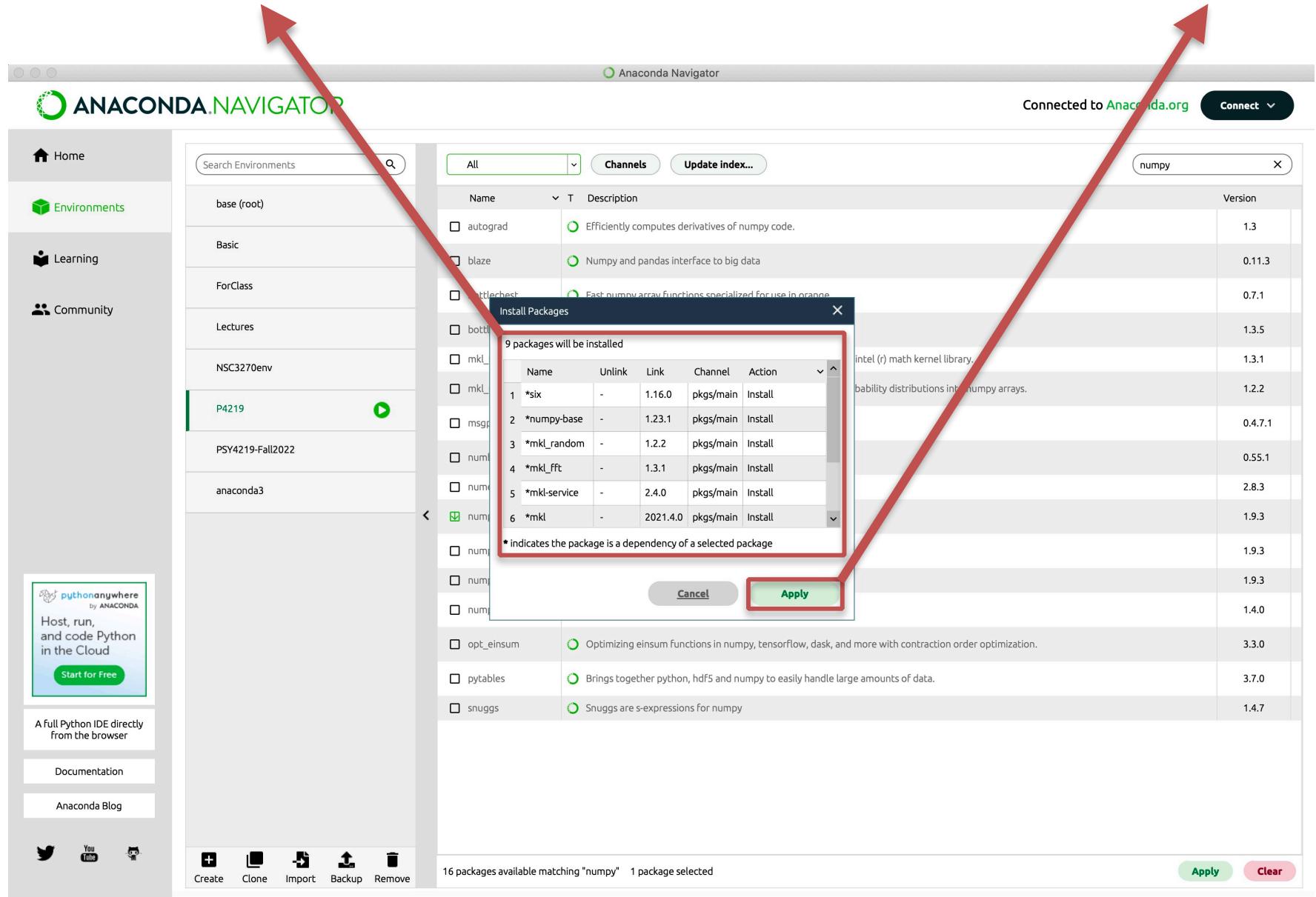


<https://docs.anaconda.com/anaconda/navigator/>

virtual environments in Anaconda Navigator

numpy and its dependencies being installed

click "Apply"



<https://docs.anaconda.com/anaconda/navigator/>

virtual environments in Anaconda Navigator

do the same thing for `scipy`, `matplotlib`, `seaborn`, `pandas`, `scikit-learn`

The screenshot shows the Anaconda Navigator interface. On the left is a sidebar with icons for Home, Environments, Learning, and Community. A callout box for "pythonanywhere by ANACONDA" offers a "Start for Free" button. Below the sidebar are links for "A full Python IDE directly from the browser", "Documentation", and "Anaconda Blog". At the bottom are social media links for Twitter, YouTube, and GitHub, along with buttons for "Create", "Clone", "Import", "Backup", and "Remove". The main area displays a list of packages in the "P4219" environment. The list includes:

Name	Description	Version
_anaconda_depends	Simplifies package management and deployment of anaconda	5.3.1
_go_select	The golang select package.	2.3.0
_ipyw_jlab_nb_ex...	A configuration metapackage for enabling anaconda-bundled jupyter extensions	0.1.0
_libarchive_static....	A static build of libarchive containing only conda-related parts	3.4.3
_libgcc_mutex	Mutex for libgcc and libgcc-ng	0.1
_low_priority	Metapackage to lower the priority of a package using track_features	1.0
_mutex_mxnet	Mutex package to pin a variant of mxnet conda package	0.0.40
_py-xgboost-mutex	Scalable, portable and distributed gradient boosting (gbdt, gbrt or gbm) library, for python, r, java, scala, c++ and more. runs on single machine, hadoop, spark, flink and dataflow	2.0
_pytorch_select	Metapackage for establishing variant priority in pytorch variants.	0.1
_r-mutex	A mutex package to ensure environment exclusivity between anaconda r and mro.	1.0.0
_r-xgboost-mutex	Scalable, portable and distributed gradient boosting (gbdt, gbrt or gbm) library, for python, r, java, scala, c++ and more. runs on single machine, hadoop, spark, flink and dataflow	2.0
_sysroot_linux-64...		3
_tflow_1100_select		0.0.2
_tflow_190_select		0.0.3
_tflow_select		2.3.0
abseil-cpp	Abseil common libraries (c++)	202111...
absl-py	Abseil python common libraries, see https://github.com/abseil/abseil-py .	0.9.0
access	Classical and novel measures of spatial accessibility to services	1.1.3
acl-amzn2-aarch64		2.2.51
adal	The adal for python library makes it easy for python application to authenticate to azure active directory (aad) in order to access aad protected web resources.	1.2.7

At the bottom of the list, it says "9238 packages available".

<https://docs.anaconda.com/anaconda/navigator/>

virtual environments in Anaconda Navigator

DO NOT update packages (unless you know what you are doing)

The screenshot shows the Anaconda Navigator interface. On the left, there's a sidebar with links to Home, Environments (which is selected and highlighted in blue), Learning, and Community. Below these are links for PythonAnywhere (with a 'Start for Free' button), a full Python IDE, Documentation, and the Anaconda Blog. At the bottom of the sidebar are social media icons for Twitter, YouTube, and GitHub, and navigation buttons for Create, Clone, Import, Backup, and Remove.

The main area displays a list of installed packages in the 'PSY4219-Fall2022' environment. The packages are listed in a table with columns for Name, Description, and Version. One specific package, 'argon2-cffi', has its version number '20.1.0' highlighted with a red box. Other packages listed include anyio, aom, appdirs, appnope, arabic-reshaper, arabic_reshaper, argon2-cffi, asttokens, astunparse, attrs, babel, backcall, beautifulsoup4, bleach, blosc, brotli, brotli-bin, brotlipy, bz2, and c-ares.

Name	Description	Version
anyio	High level compatibility layer for multiple asynchronous event loop implementations on python	3.5.0
aom		3.4.0
appdirs	A small python module for determining appropriate platform-specific dirs.	1.4.4
appnope	Disable app nap on os x 10.9	0.1.2
arabic-reshaper		2.1.3
arabic_reshaper		2.1.3
argon2-cffi	The secure argon2 password hashing algorithm.	20.1.0
asttokens	The asttokens module annotates python abstract syntax trees (asts) with the positions of tokens and text in the source code that generated them.	2.0.5
astunparse	An ast unparser for python.	1.6.3
attrs	Attrs is the python package that will bring back the joy of writing classes by relieving you from the drudgery of implementing object protocols (aka dunder methods).	21.4.0
babel	Utilities to internationalize and localize python applications	2.9.1
backcall	Specifications for callback functions passed in to an api	0.2.0
beautifulsoup4	Python library designed for screen-scraping	4.11.1
bleach	Easy, whitelist-based html-sanitizing tool	4.1.0
blosc	A blocking, shuffling and loss-less compression library that can be faster than `memcpy`	1.21.1
brotli	Brotli compression format	1.0.9
brotli-bin	Brotli compression format	1.0.9
brotlipy	Python bindings to the brotli compression library	0.7.0
bz2	High-quality data compressor	1.0.8
c-ares	This is c-ares, an asynchronous resolver library	1.18.1

<https://docs.anaconda.com/anaconda/navigator/>

virtual environments in Anaconda Navigator

next, you need to install psychopy, which is not part of the anaconda distribution
(most specialized discipline-specific packages are not part of anaconda)



in some cases, you would need to install these specialized packages using
conda from the command line / terminal, but psychopy (like some specialized
packages) has a distribution on conda-forge

virtual environments in Anaconda Navigator

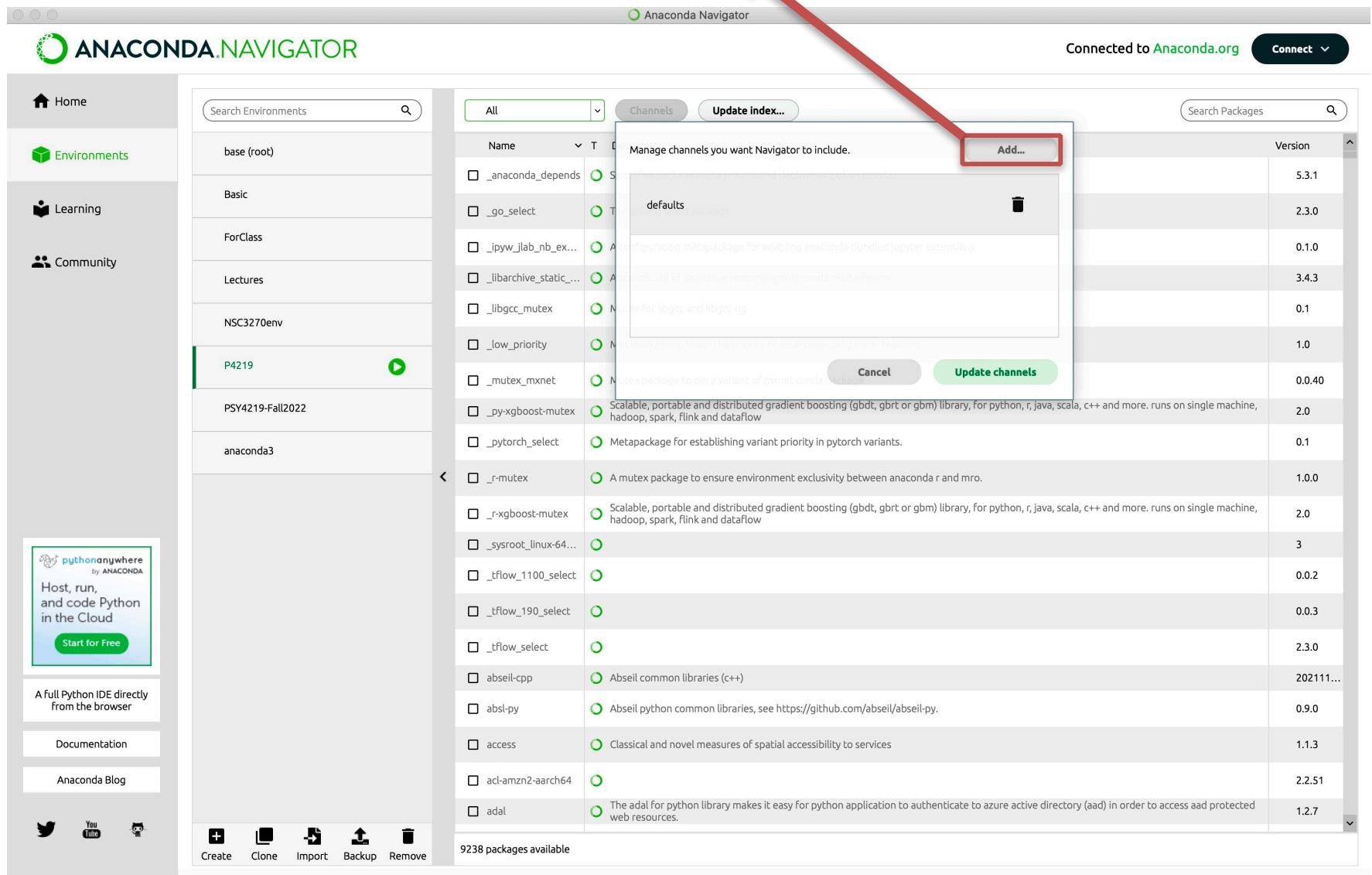
need to see if conda-forge is on your "Channels" (places Anaconda checks for packages)

A screenshot of the Anaconda Navigator interface. On the left is a sidebar with icons for Home, Environments, Learning, and Community. A central panel shows a list of environments: base (root), Basic, ForClass, Lectures, NSC3270env, P4219, PSY4219-Fall2022, and anaconda3. Below these is a large table of packages under the 'Channels' tab. The 'Channels' tab is highlighted with a red arrow pointing to it from the top right. The table has columns for Name, Description, and Version. The 'Description' column contains brief descriptions of each package, often mentioning 'Simplifies package management and deployment of anaconda' or being a 'Metapackage'. The 'Version' column shows the current version of each package. At the bottom of the table, it says '9238 packages available'. The top right of the interface shows 'Connected to Anaconda.org' and a 'Connect' button.

<https://docs.anaconda.com/anaconda/navigator/>

virtual environments in Anaconda Navigator

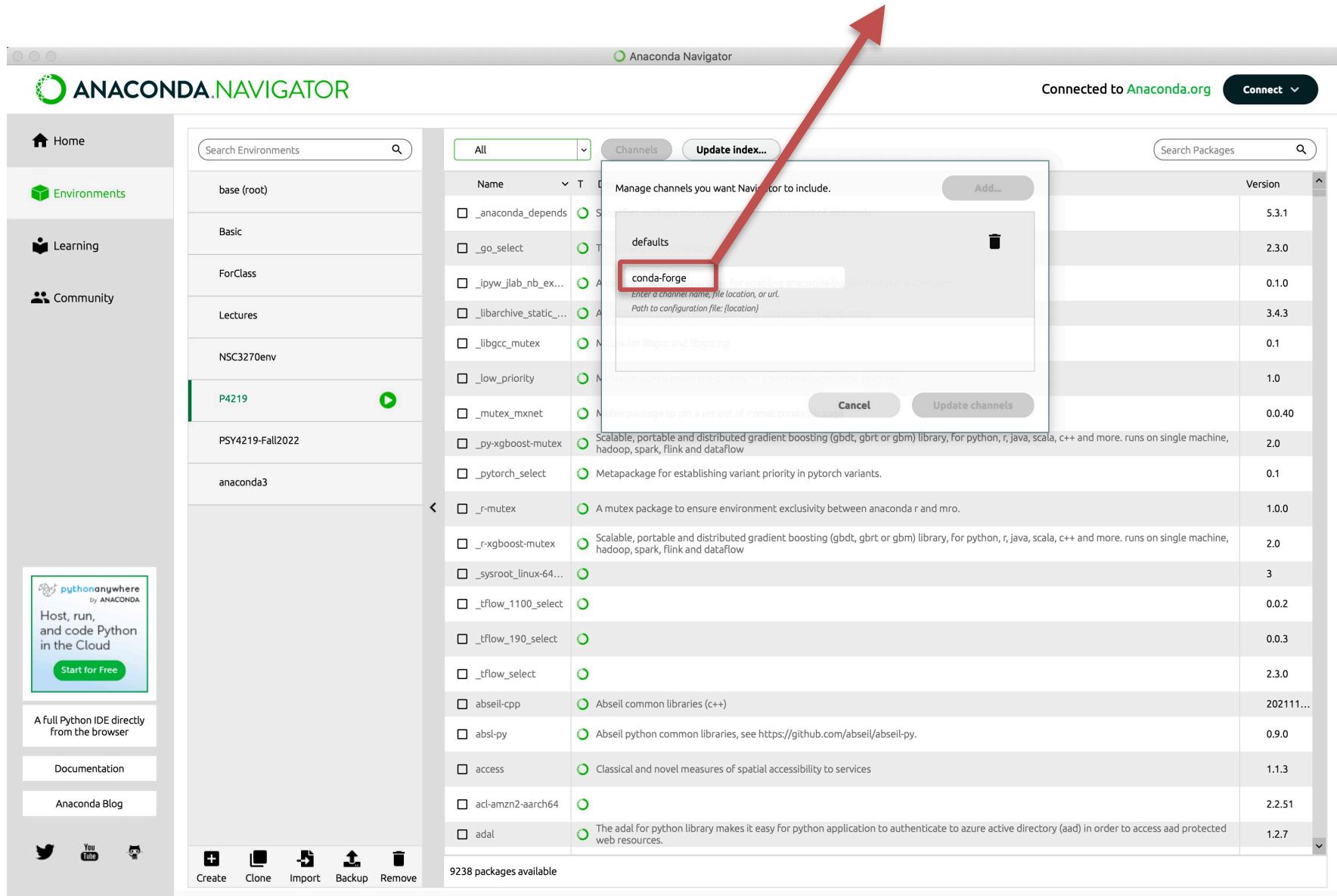
if conda-forge is not listed, click "Add..."



<https://docs.anaconda.com/anaconda/navigator/>

virtual environments in Anaconda Navigator

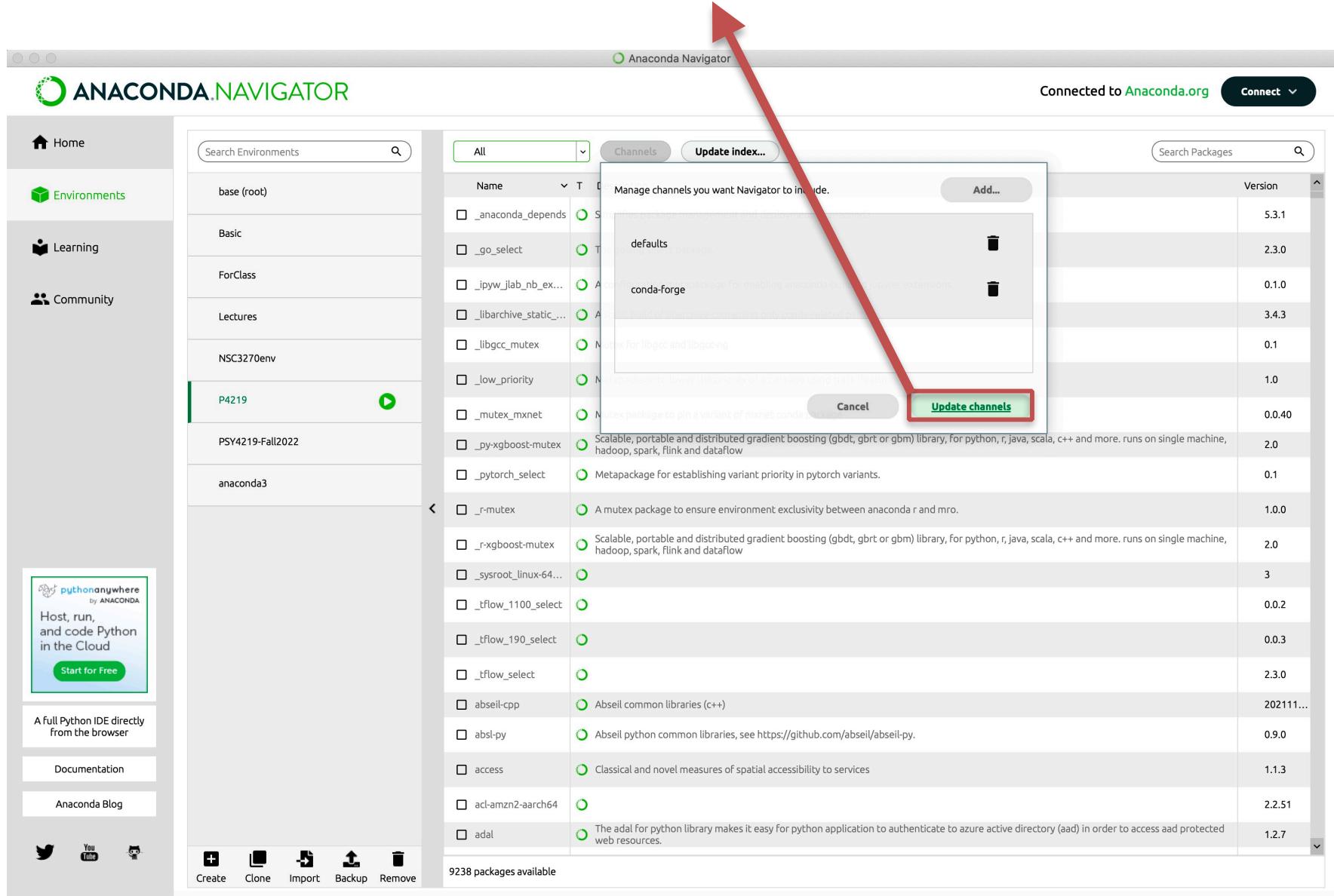
type in conda-forge and hit enter/return



<https://docs.anaconda.com/anaconda/navigator/>

virtual environments in Anaconda Navigator

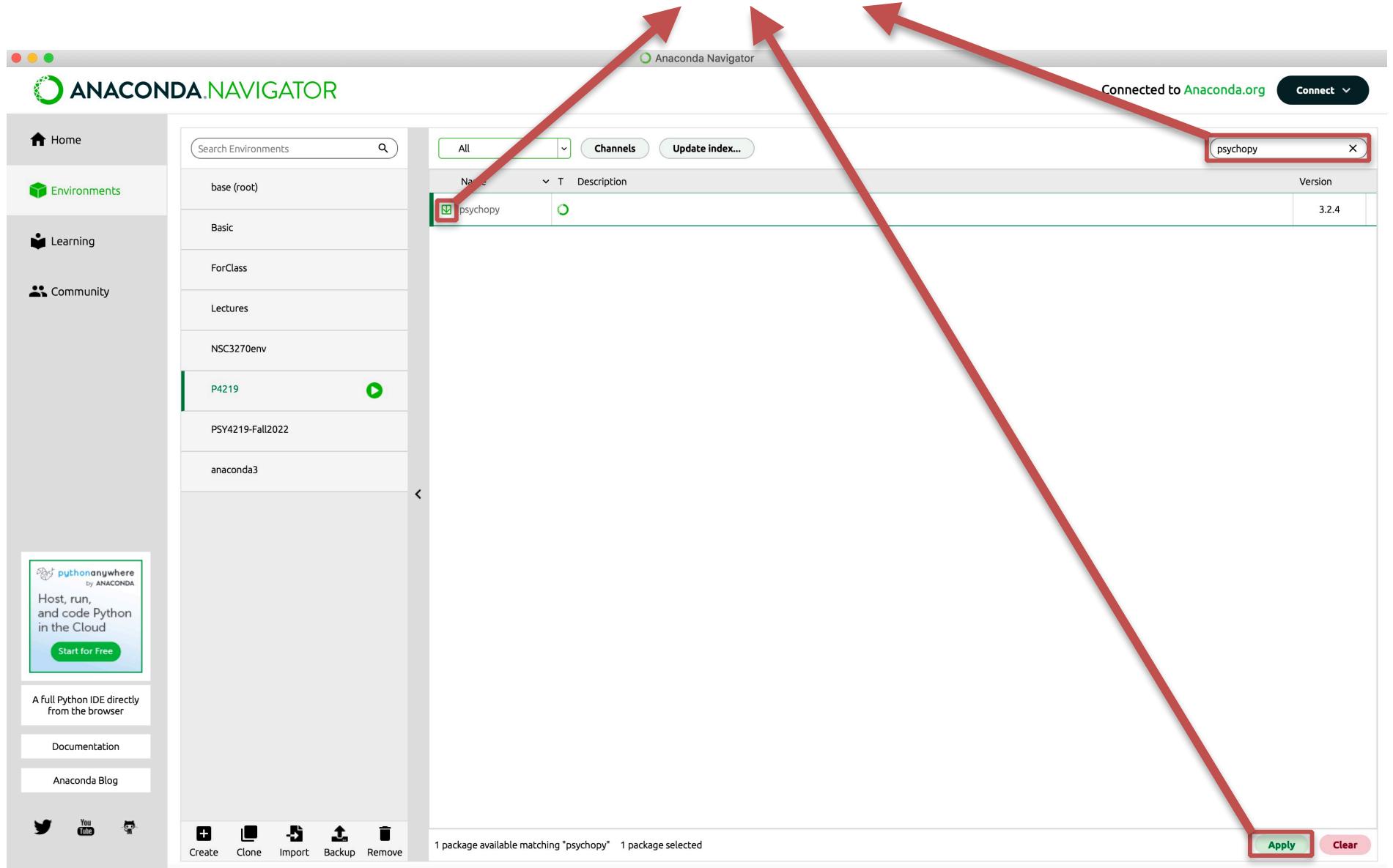
click on "Update channels"



<https://docs.anaconda.com/anaconda/navigator/>

virtual environments in Anaconda Navigator

type in psychopy, click on it, and click "Apply"



<https://docs.anaconda.com/anaconda/navigator/>

how do you use the virtual environment?

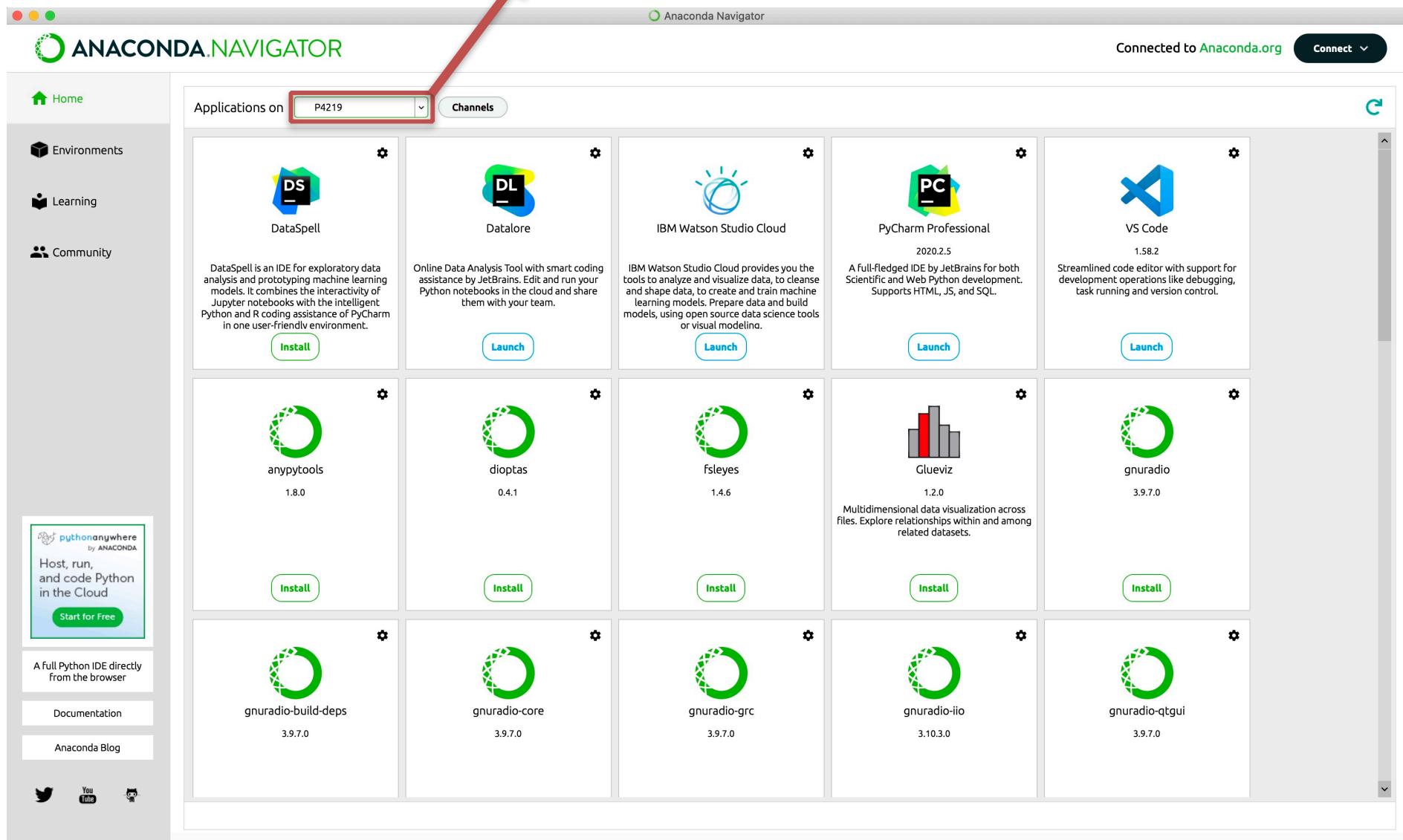
assuming it's selected, click on Home

The screenshot shows the Anaconda Navigator interface. On the left, there's a sidebar with icons for Home, Environments, Learning, and Community. A callout box highlights the 'Home' icon. Below the sidebar, there's a section for PythonAnywhere with a 'Start for Free' button. The main area shows a list of environments on the left: base (root), Basic, ForClass, Lectures, NSC3270env, P4219 (which is highlighted with a red box), PSY4219-Fall2022, and anaconda3. On the right, there's a table of installed packages. The 'Installed' tab is selected, showing 26 packages available and 1 package selected. The selected package is 'blas'. The table includes columns for Name, Description, Version, and Action (indicated by a small circular icon). The packages listed include blas, ca-certificates, certifi, intel-openmp, libcxx, libffi, mkl, mkl-fft, mkl-random, mkl-service, mkl_fft, mkl_random, ncurses, numpy, numpy-base, openssl, pip, python, readline, and setuptools.

Name	Description	Version
blas	Linear algebra package	1.0
ca-certificates	Certificates for use with other packages.	2022.0...
certifi	Python package for providing mozilla's ca bundle.	2022.6.15
intel-openmp	Math library for intel and compatible processors	2021.4.0
libcxx	Llvm c++ standard library	12.0.0
libffi	A portable foreign function interface library.	3.3
mkl	Math library for intel and compatible processors	2021.4.0
mkl-fft		1.3.1
mkl-random		1.2.2
mkl-service	Python hooks for intel(r) math kernel library runtime control settings.	2.4.0
mkl_fft	Numpy-based implementation of fast fourier transform using intel (r) math kernel library.	1.3.1
mkl_random	Intel (r) mkl-powered package for sampling from common probability distributions into numpy arrays.	1.2.2
ncurses	Library for text-based user interfaces	6.3
numpy	Array processing for numbers, strings, records, and objects.	1.23.1
numpy-base	Array processing for numbers, strings, records, and objects.	1.23.1
openssl	Openssl is an open-source implementation of the ssl and tls protocols	1.1.1q
pip	Pypa recommended tool for installing python packages	22.1.2
python	General purpose programming language	3.8.13
readline	Library for editing command lines as they are typed in	8.1.2
setuptools	Download, build, install, upgrade, and uninstall python packages	63.4.1

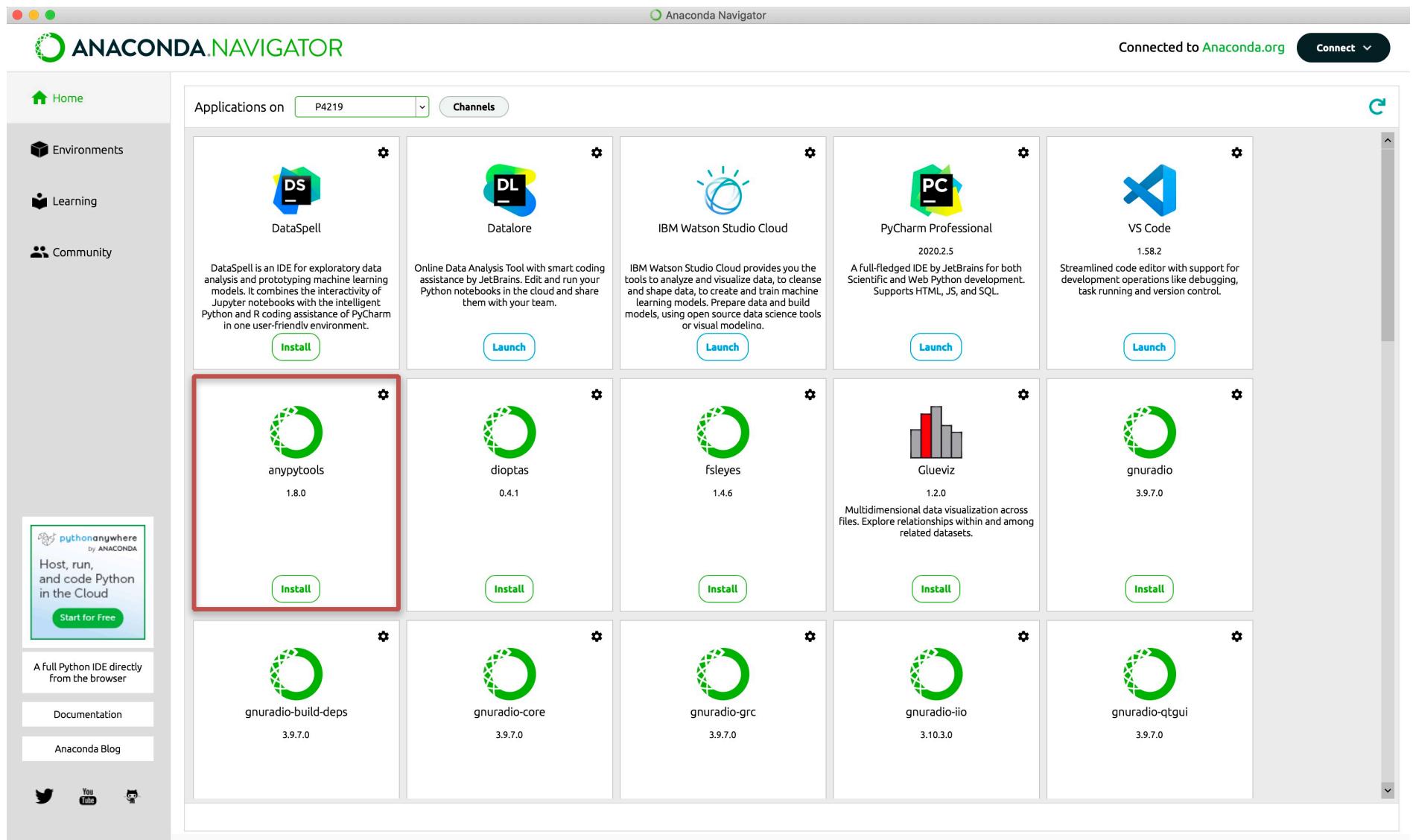
how do you use the virtual environment?

can see here that your new environment is selected (can also select others)



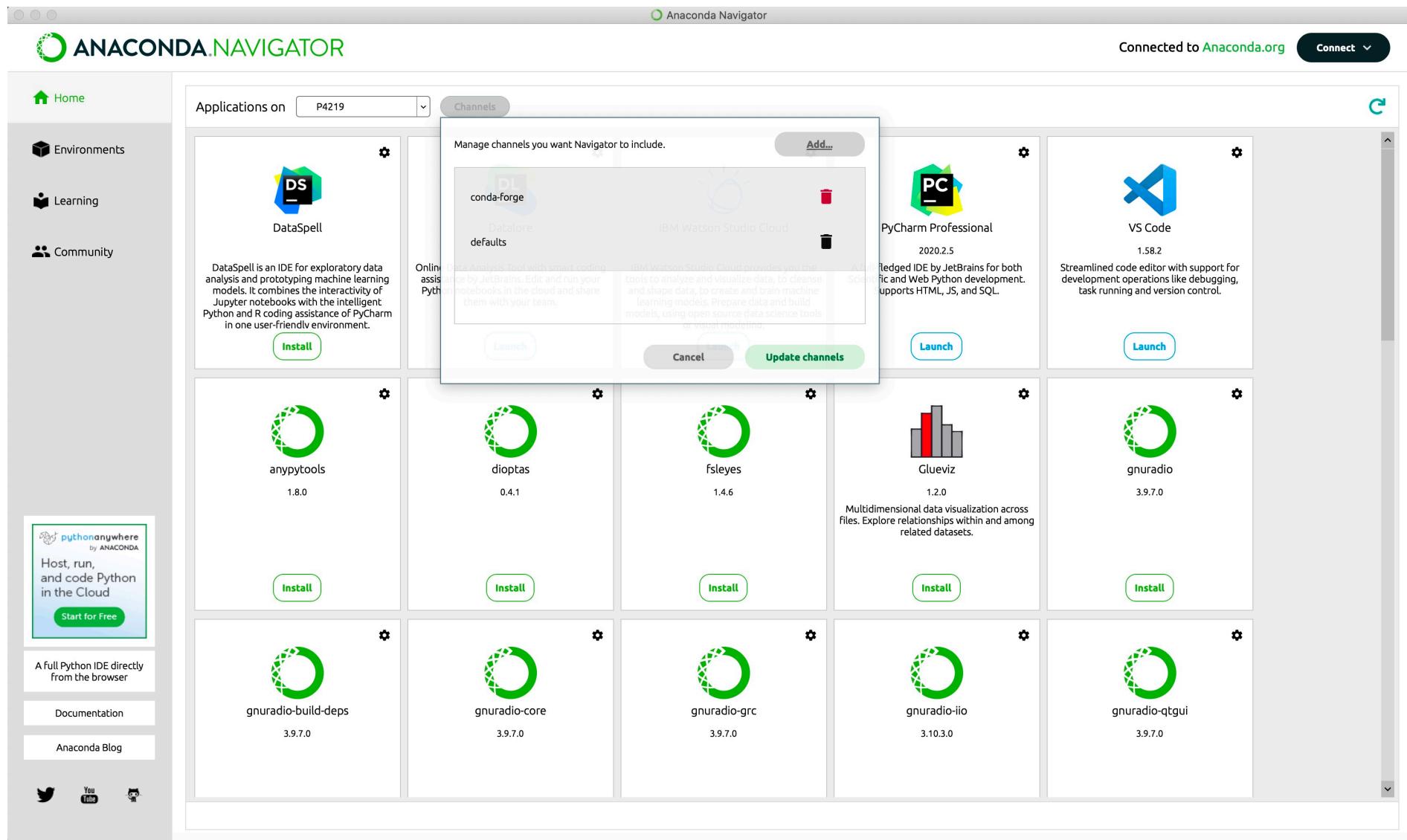
how do you use the virtual environment?

you might also see a bunch of other stuff on the dashboard (from conda-forge)



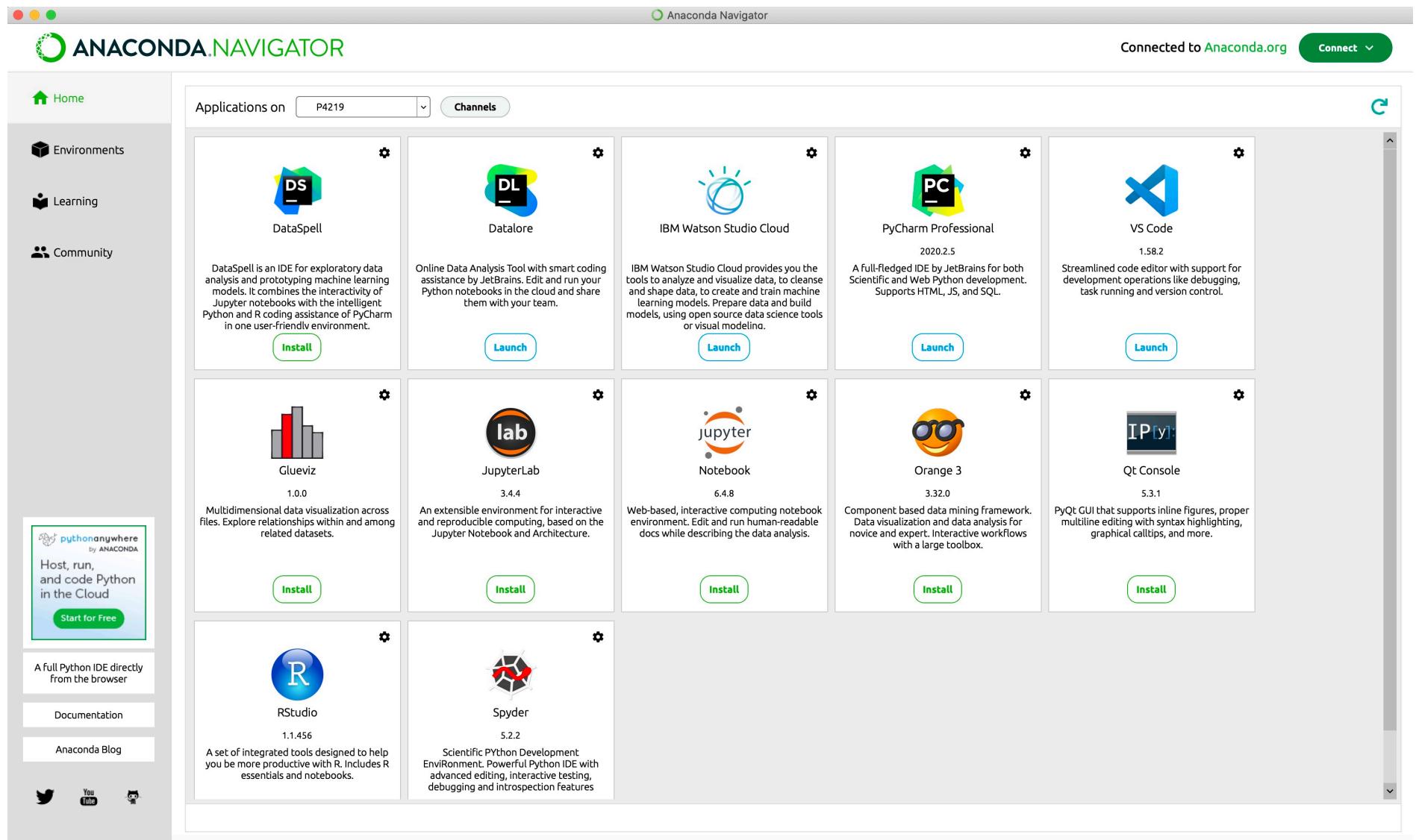
how do you use the virtual environment?

go into channels and delete conda-forge (trash can) and "Update channels"



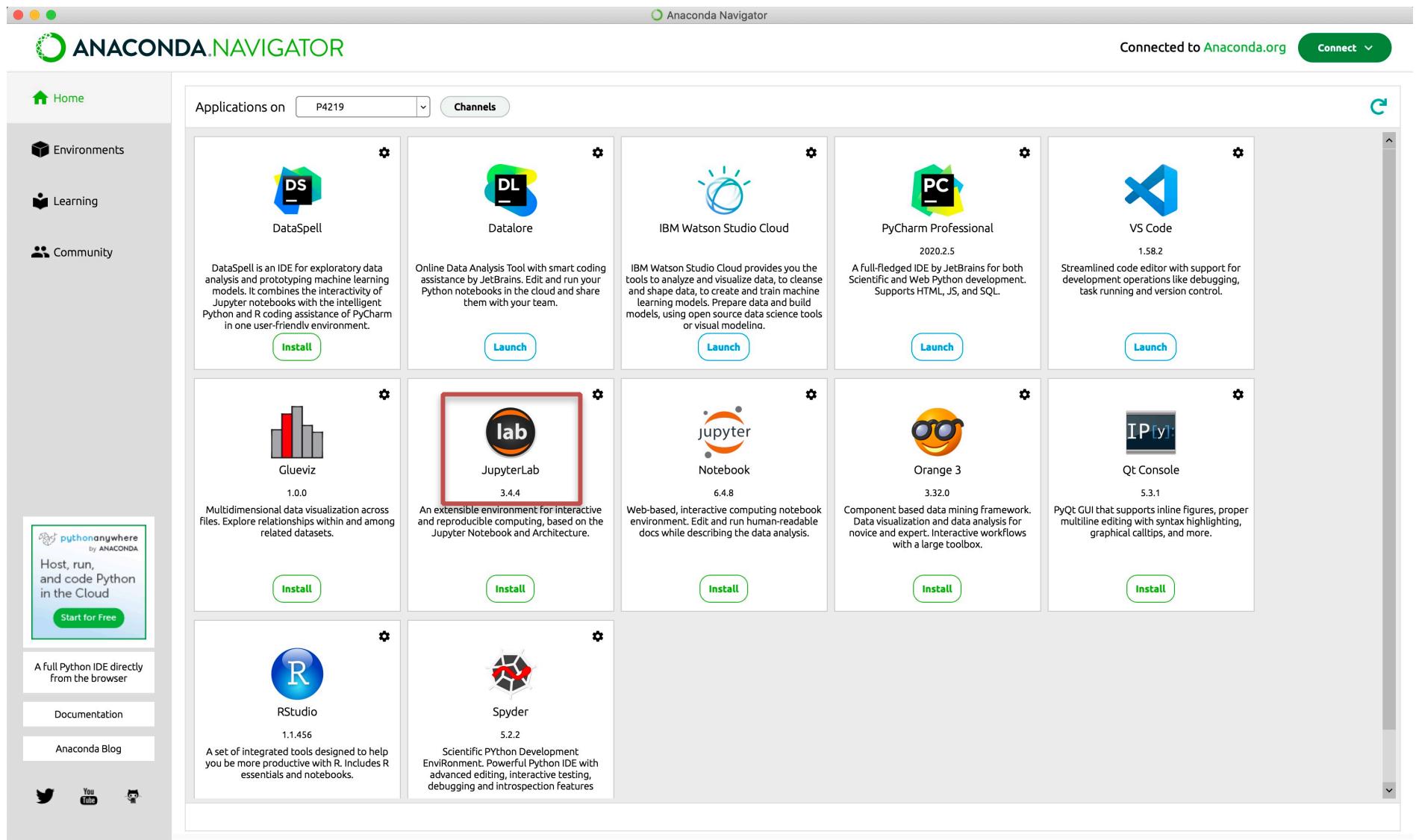
how do you use the virtual environment?

and they'll go away



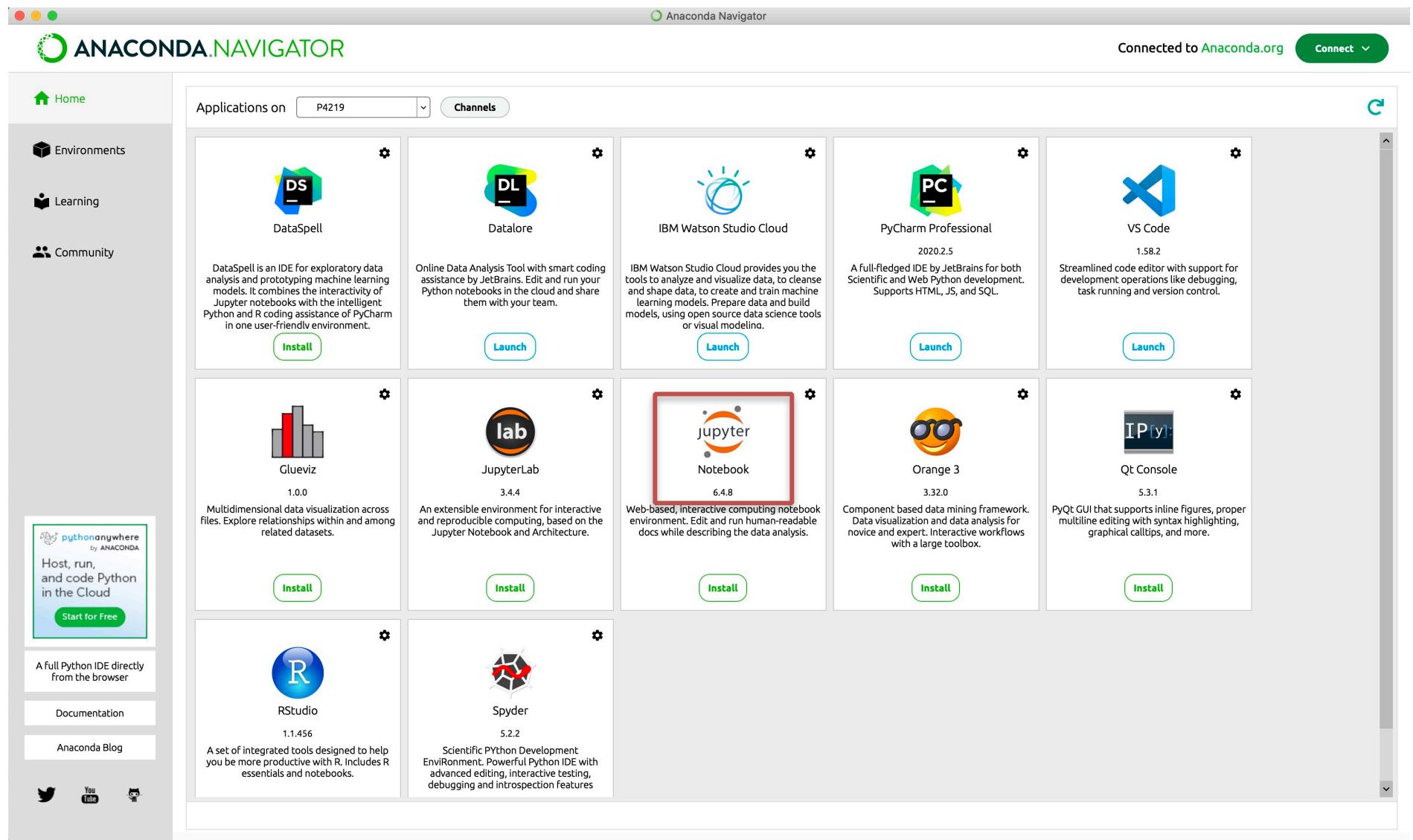
how do you use the virtual environment?

I'll be using JupyterLab



how do you use the virtual environment?

if you prefer Jupyter Notebook you can use that (JupyterLab has a bit more functionality)



how do you use the virtual environment?

you'll need to "Install" (really adding more packages to your new environment)

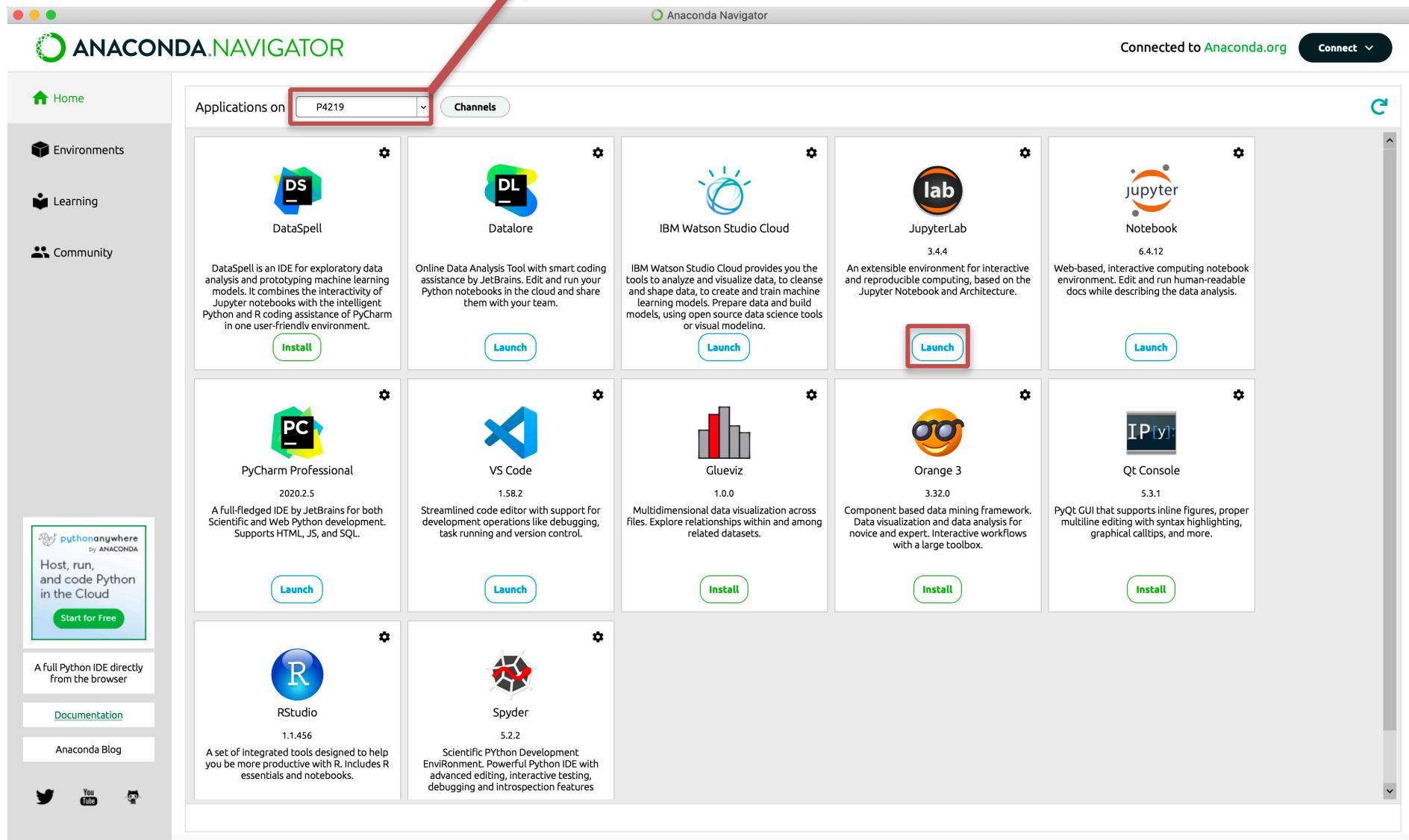
The screenshot shows the Anaconda Navigator interface. On the left is a sidebar with links to Home, Environments, Learning, and Community. A callout box highlights the "pythonanywhere by ANACONDA" link, which says "Host, run, and code Python in the Cloud" and has a "Start for Free" button. Below this are links for "A Full Python IDE directly from the browser", "Documentation", and "Anaconda Blog". At the bottom of the sidebar are social media icons for Twitter, YouTube, and GitHub.

The main area displays a grid of applications:

Application	Description	Version	Action
DataSpell	DataSpell is an IDE for exploratory data analysis and prototyping machine learning models. It combines the interactivity of Jupyter notebooks with the intelligent Python and R coding assistance of PyCharm in one user-friendly environment.	1.0.0	Install
Datalore	Online Data Analysis Tool with smart coding assistance by JetBrains. Edit and run your Python notebooks in the cloud and share them with your team.	3.4.4	Launch
IBM Watson Studio Cloud	IBM Watson Studio Cloud provides you the tools to analyze and visualize data, to cleanse and shape data, to create and train machine learning models. Prepare data and build models, using open source data science tools or visual modeling.	6.4.8	Launch
PyCharm Professional	A full-fledged IDE by JetBrains for both Scientific and Web Python development. Supports HTML, JS, and SQL.	2020.2.5	Launch
VS Code	Streamlined code editor with support for development operations like debugging, task running and version control.	1.58.2	Launch
Glueviz	Multidimensional data visualization across files. Explore relationships within and among related datasets.	1.0.0	Install
JupyterLab	An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.	3.4.4	Install
Notebook	Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.	6.4.8	Install
Orange 3	Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.	3.32.0	Install
Qt Console	PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.	5.3.1	Install
RStudio	A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.	1.1.456	Install
Spyder	Scientific Python Development Environment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features.	5.2.2	Install

launch JupyterLab (or Jupyter Notebook)

using this environment



Jupyter (Lab and Notebook) runs in a browser

we'll talk about using Jupyter in a bit

The screenshot shows a Jupyter Notebook interface running in a web browser. The browser's address bar displays "localhost:8891/lab". The left sidebar contains a file tree showing various files and folders, including "Week9b.ipynb", "Week9a.ipynb", and "Week9b.zip". The main area is divided into two code cells. The top cell, titled "image processing", contains Python code for image filtering and plotting. The bottom cell, titled "Masking", contains Python code for reading an image, extracting its data, and determining its dimensions. The browser's status bar at the bottom indicates "Mode: Command" and "Ln 1, Col 1 Week9b.ipynb".

```
[ ]: import numpy as np
import matplotlib.pyplot as plt
from PIL import Image

def myfilter(x, alpha=1., beta=2., gam=.5, lam=.1):
    F = 1 - np.exp(-(x/alpha)**beta)
    psi = gam + (1. - lam - gam)*F
    return (psi)

alpha = 100
beta = 3
gam = 0.48
lam = 0.48

x = np.arange(0, 255, .1)
psi = myfilter(x, alpha, beta, gam, lam)
plt.plot(x, psi, 'r-');
plt.ylim((0,1))

im = Image.open('Jordingray.bmp')

imdata = np.asarray(im)

imdata2 = myfilter(imdata, alpha, beta, gam, lam)

imdata2 = np.uint8(255*imdata2)

im2 = Image.fromarray(imdata2)

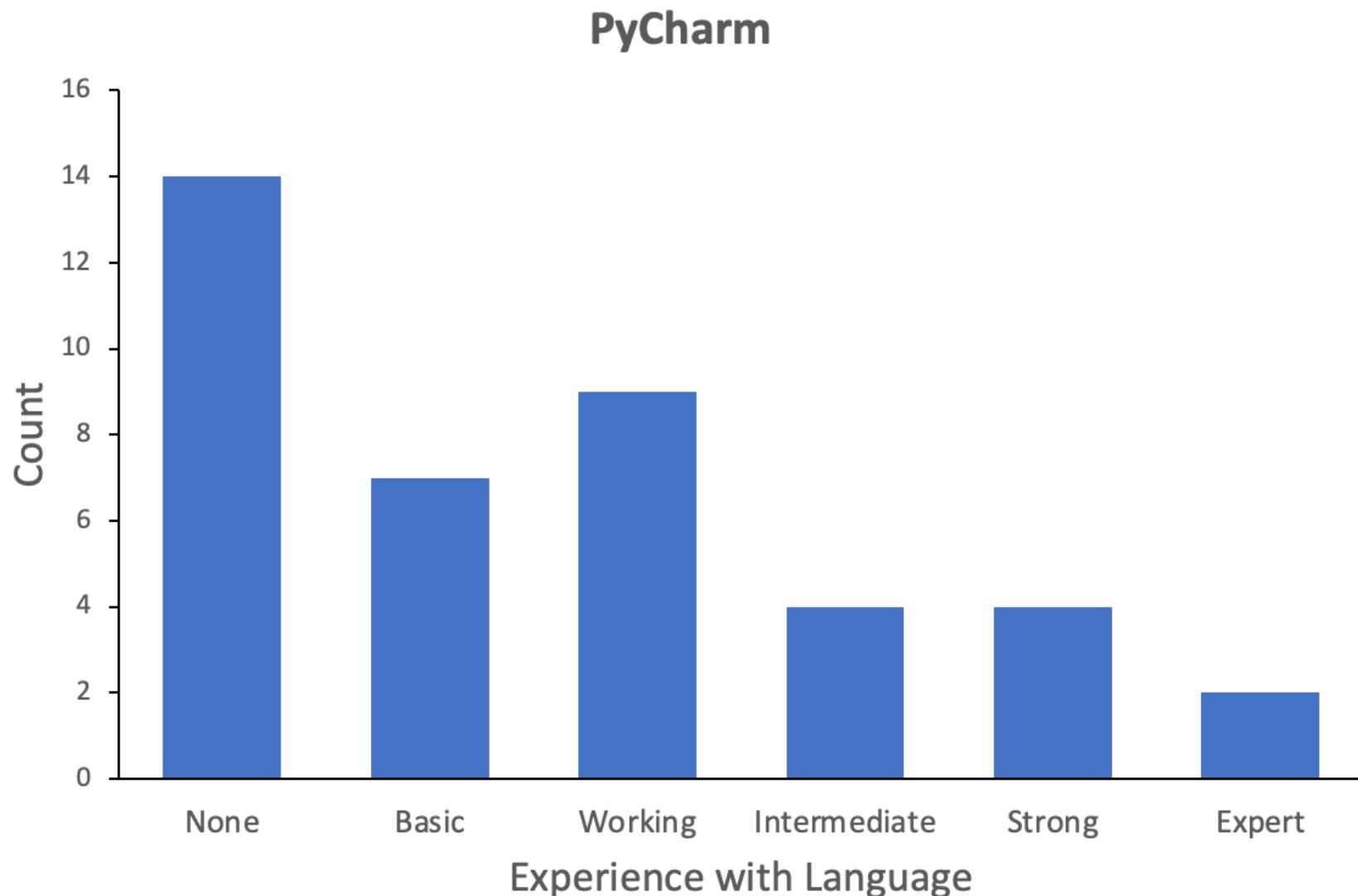
display(im)
display(im2)
```

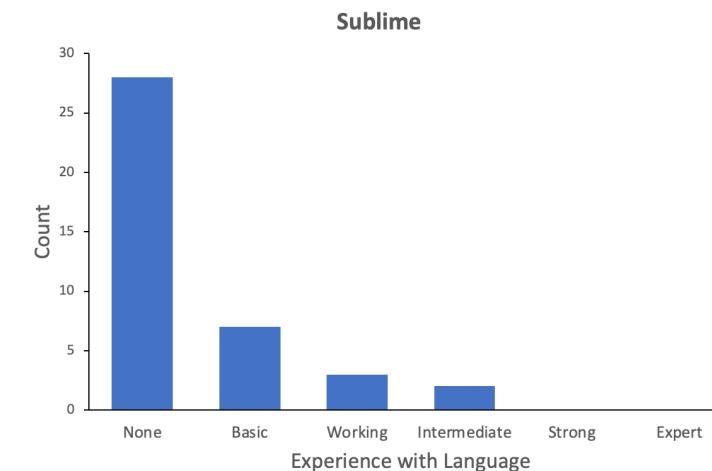
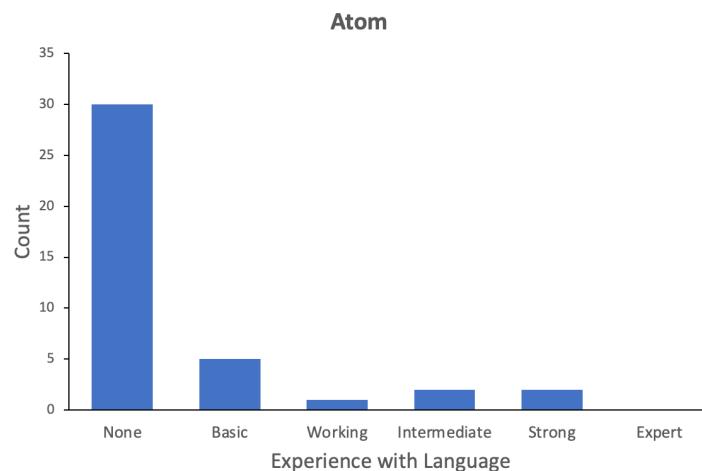
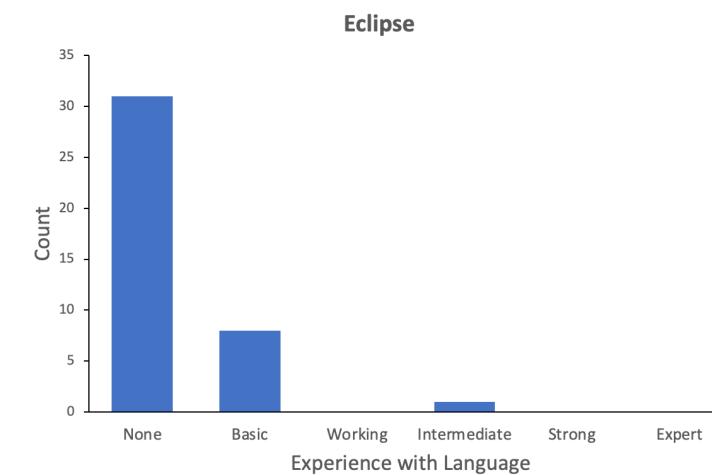
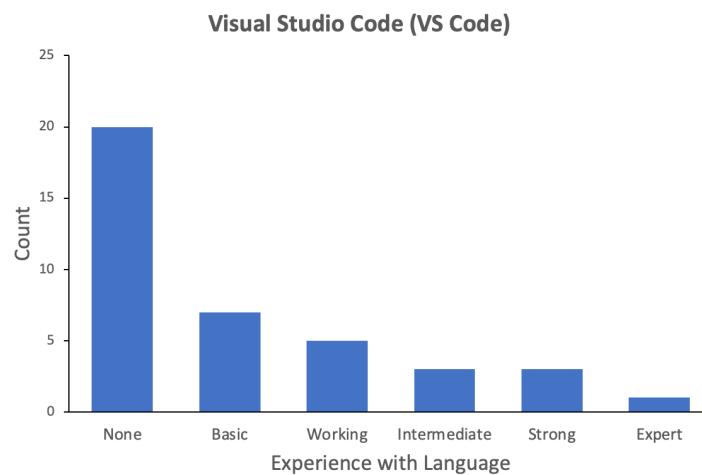
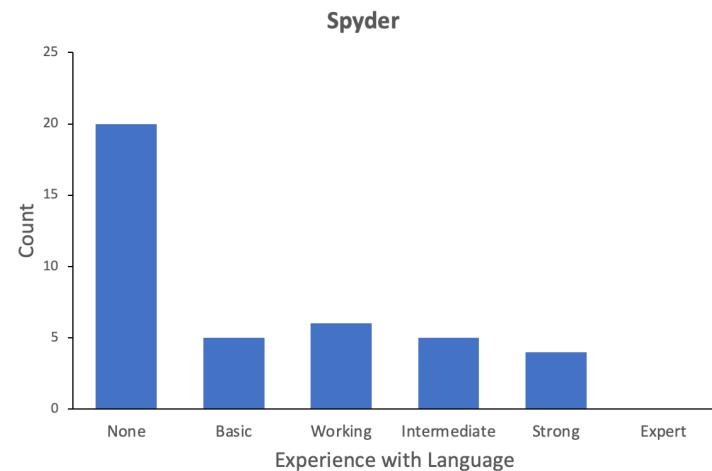
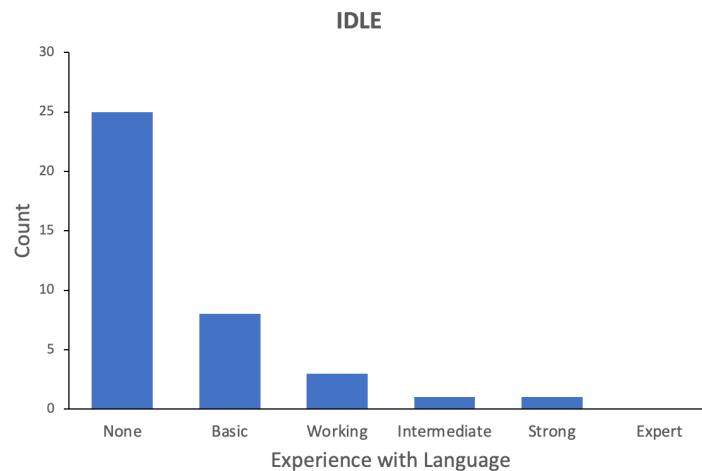
```
[1]: import numpy as np
import matplotlib.pyplot as plt
from PIL import Image

im = Image.open('Jordingray.bmp')

imdata = np.asarray(im)
r,c = imdata.shape
```

Integrated Development Environments





Integrated Development Environments

- edit code (often with hints, autocompletion, instant documentation)
- run code
- debug code (break points, stepping through code, watching variable)
- Python terminal
- more advanced tools

PyCharm

if you want to be able to follow along in PyCharm
(using PyCharm is not required)

while we will not use PyCharm for a while, I would like you to get PyCharm (or another IDE) working now at the start of the semester (to address issues)

the main thing we're trying to do now is to make sure PyCharm uses the virtual environment we just created

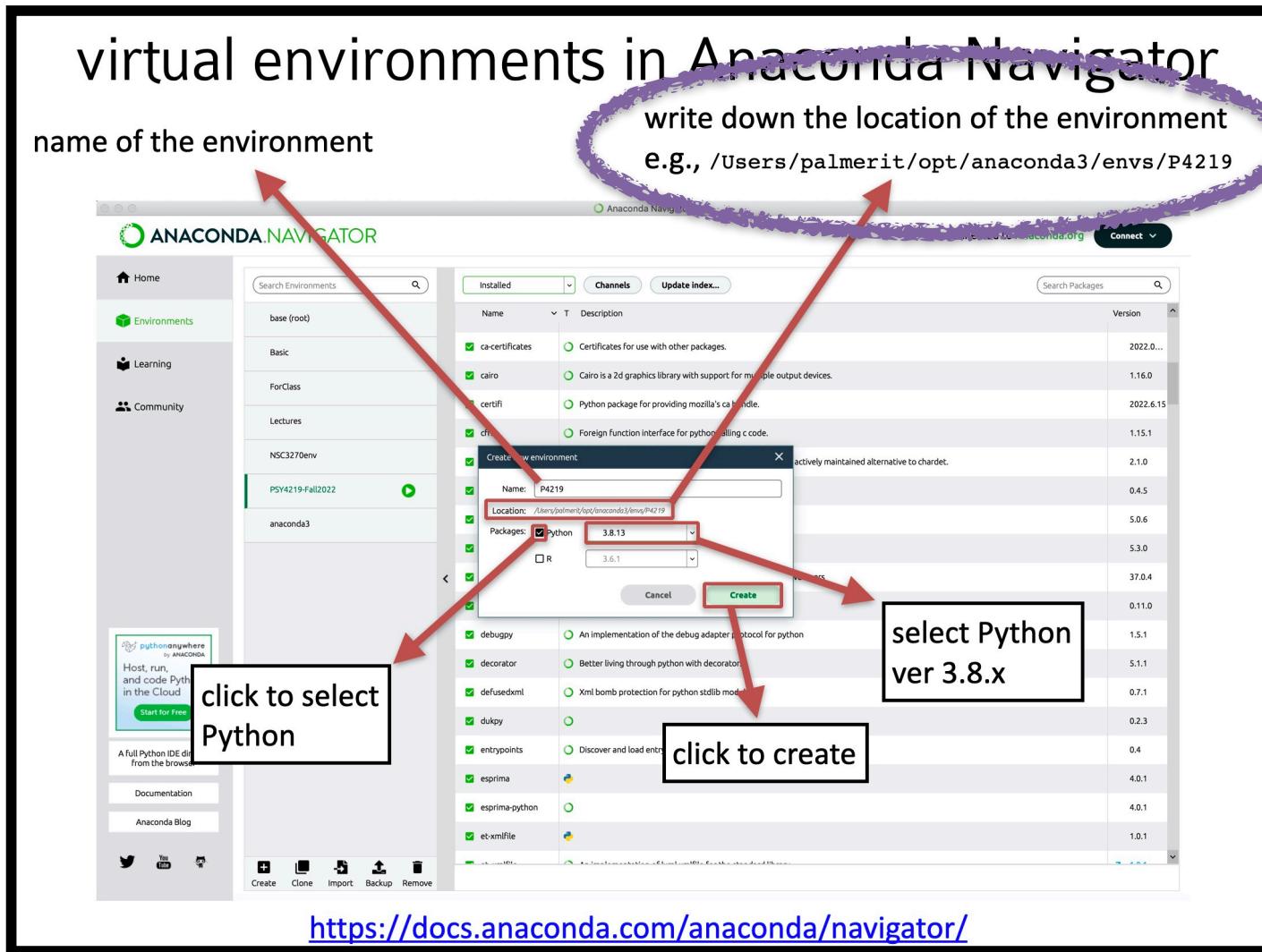
(if you are using another IDE, you will need to make sure you set up that IDE to use that environment - Jason knows VS Code, but for other IDEs you will need to play around on your own or use Piazza for help)

Installing PyCharm

- Do not install PyCharm from Anaconda (to ensure that you install the full free educational version)
- <https://www.jetbrains.com/shop/eform/students>
- make sure you register using your vanderbilt.edu email address to get the full featured free educational version
- we will talk about using PyCharm later (feel free to explore on your own)

Setting up PyCharm

remember when I said you need to write down the location?
--- you'll need it now ---

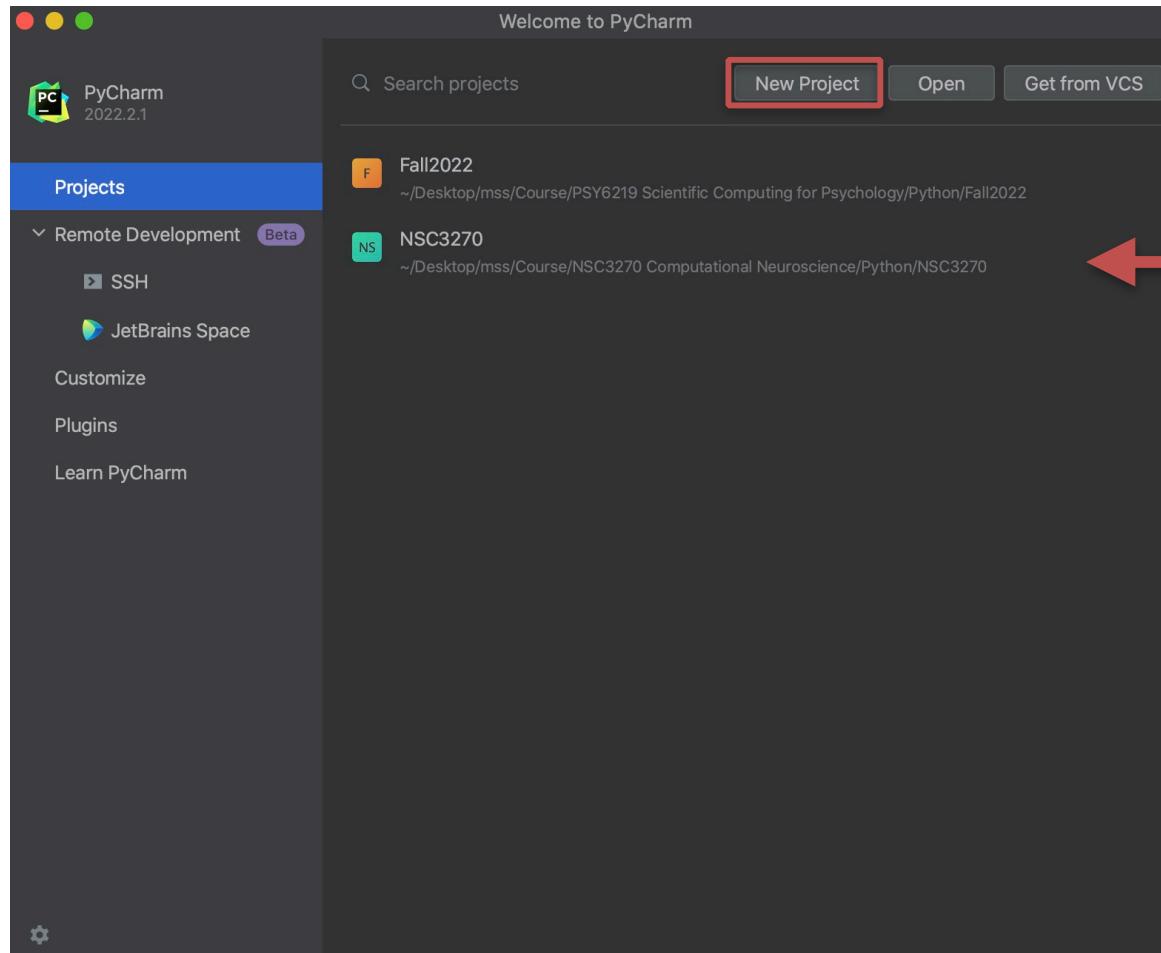


Setting up PyCharm

launch Python ... click "New Project"

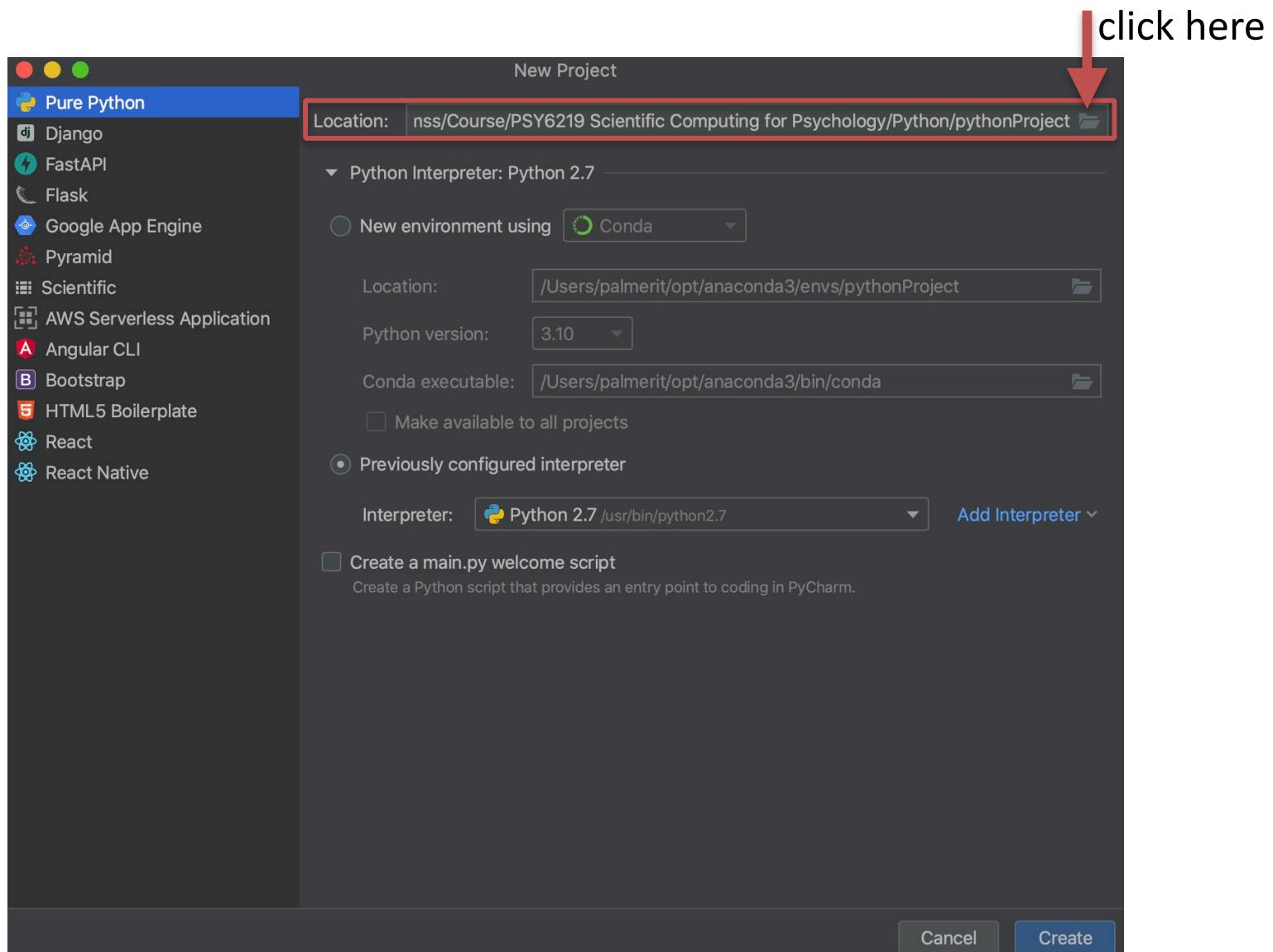
you can create a new project for every assignment
or use the same project for every assignment

note that after you create a project, PyCharm will open with that project (until you close it)



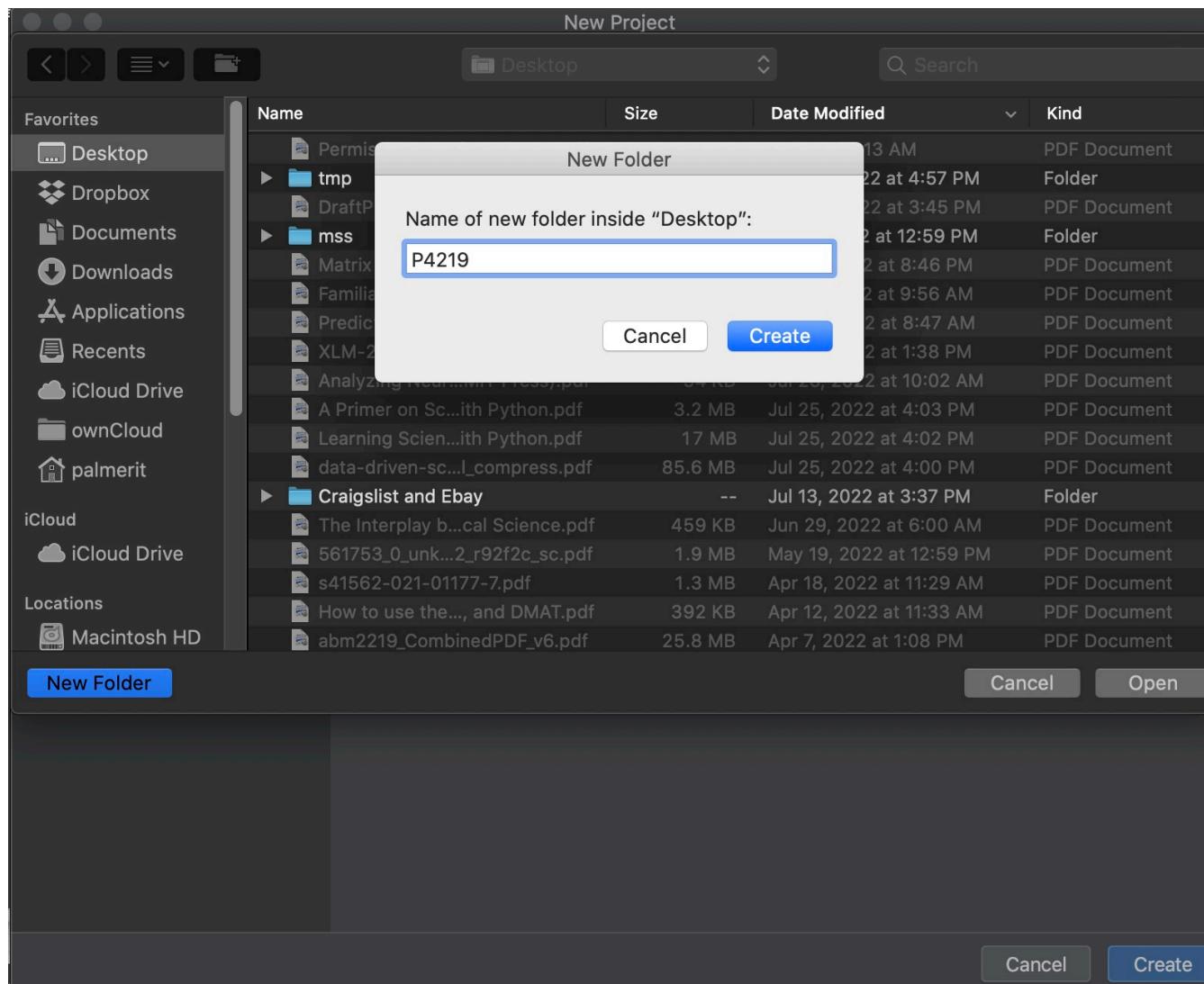
Setting up PyCharm

select a location for your project
(the folder/directory where your code will be stored)



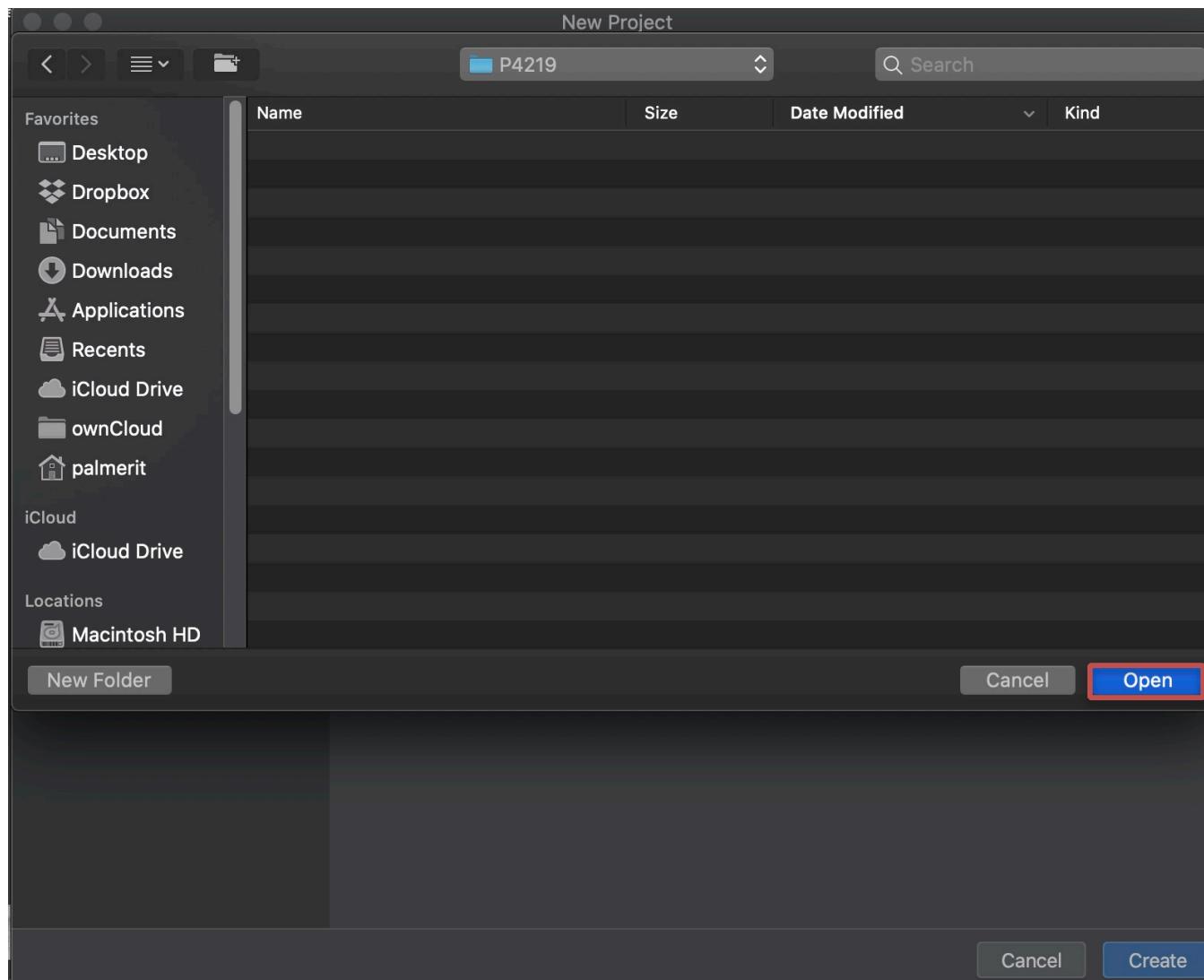
Setting up PyCharm

here, I'm creating a new folder
you can also select an existing folder



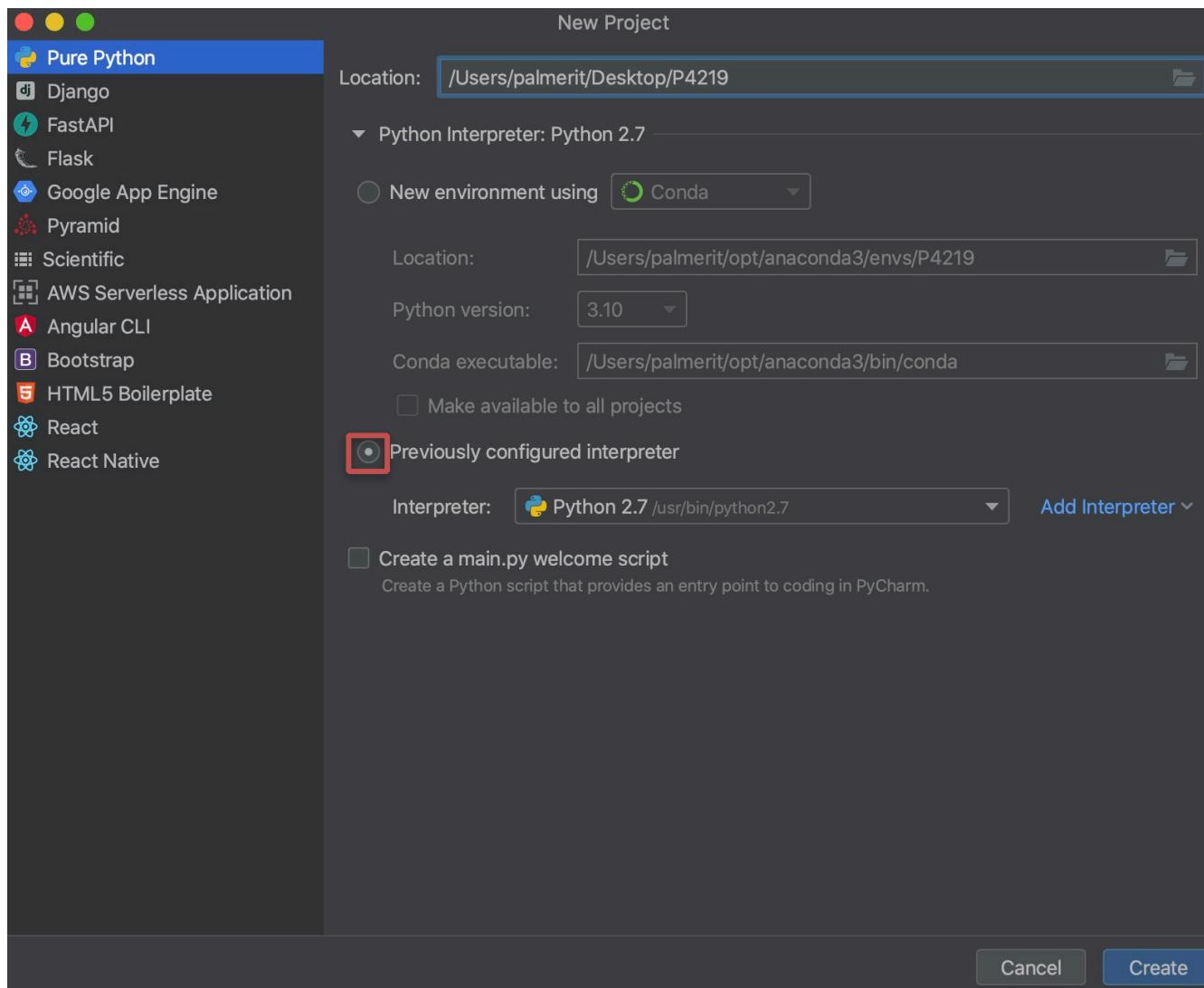
Setting up PyCharm

click "Open"



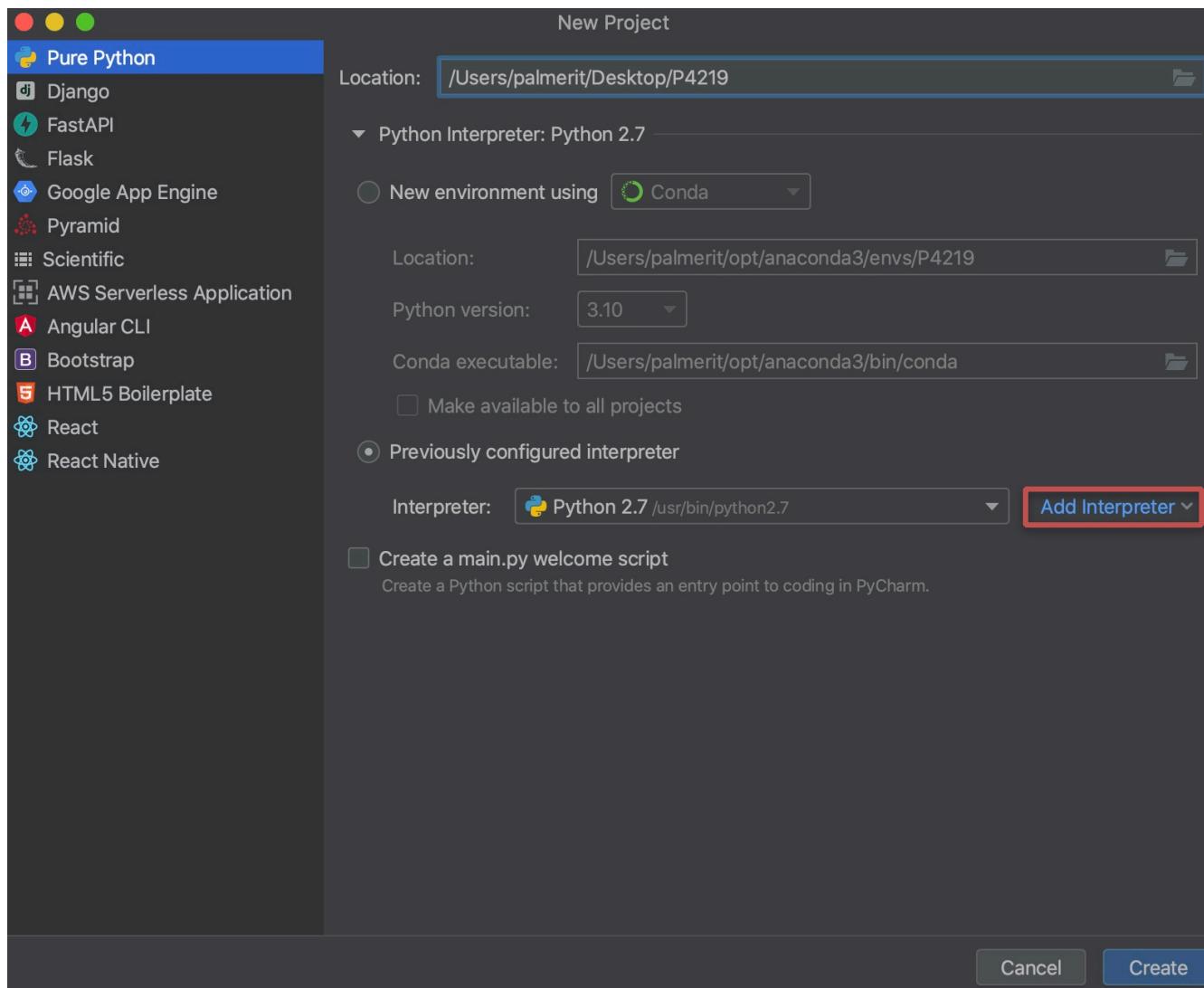
Setting up PyCharm

select "Previously configured interpreter"
(environment)



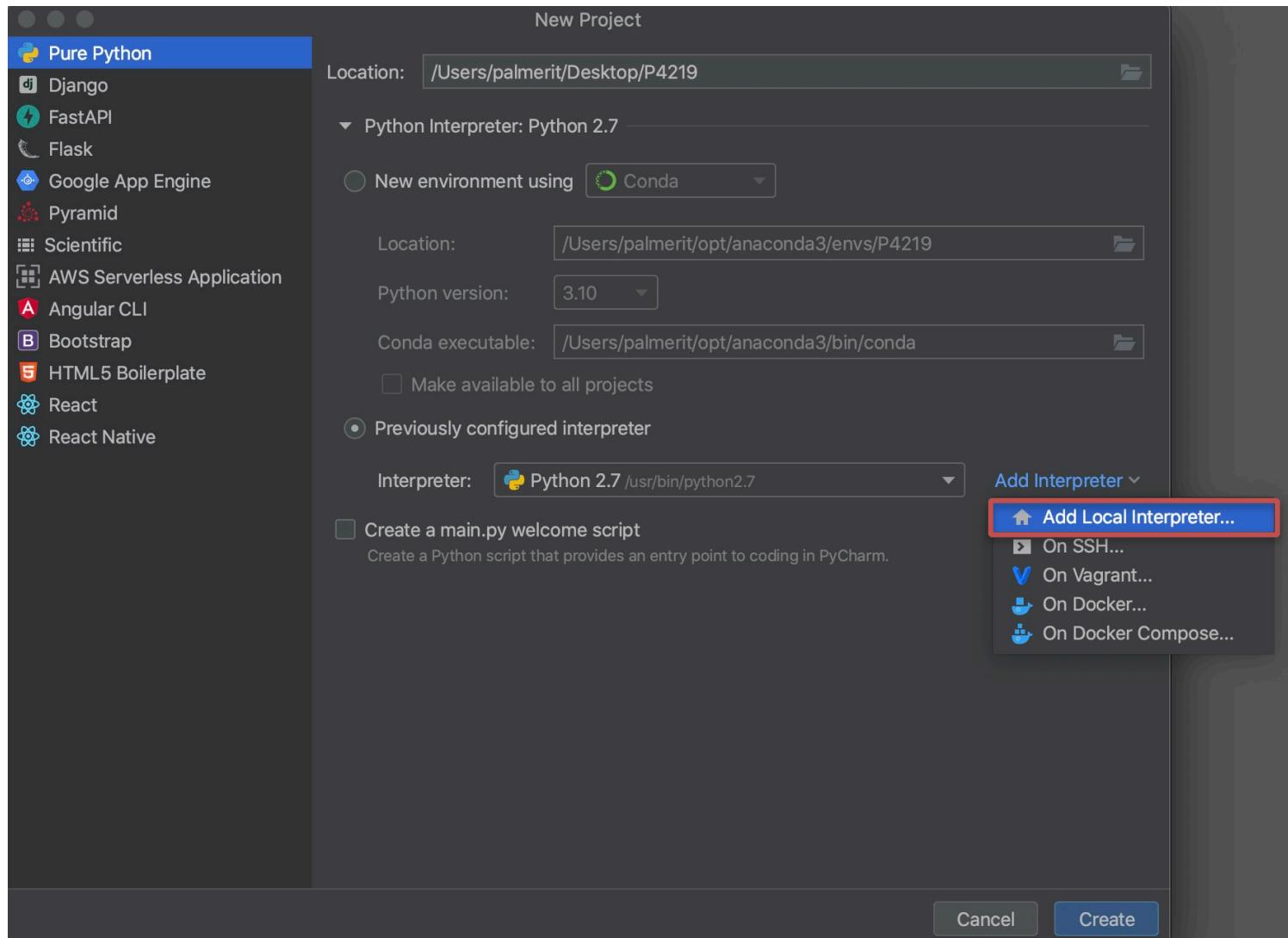
Setting up PyCharm

click "Add Interpreter"



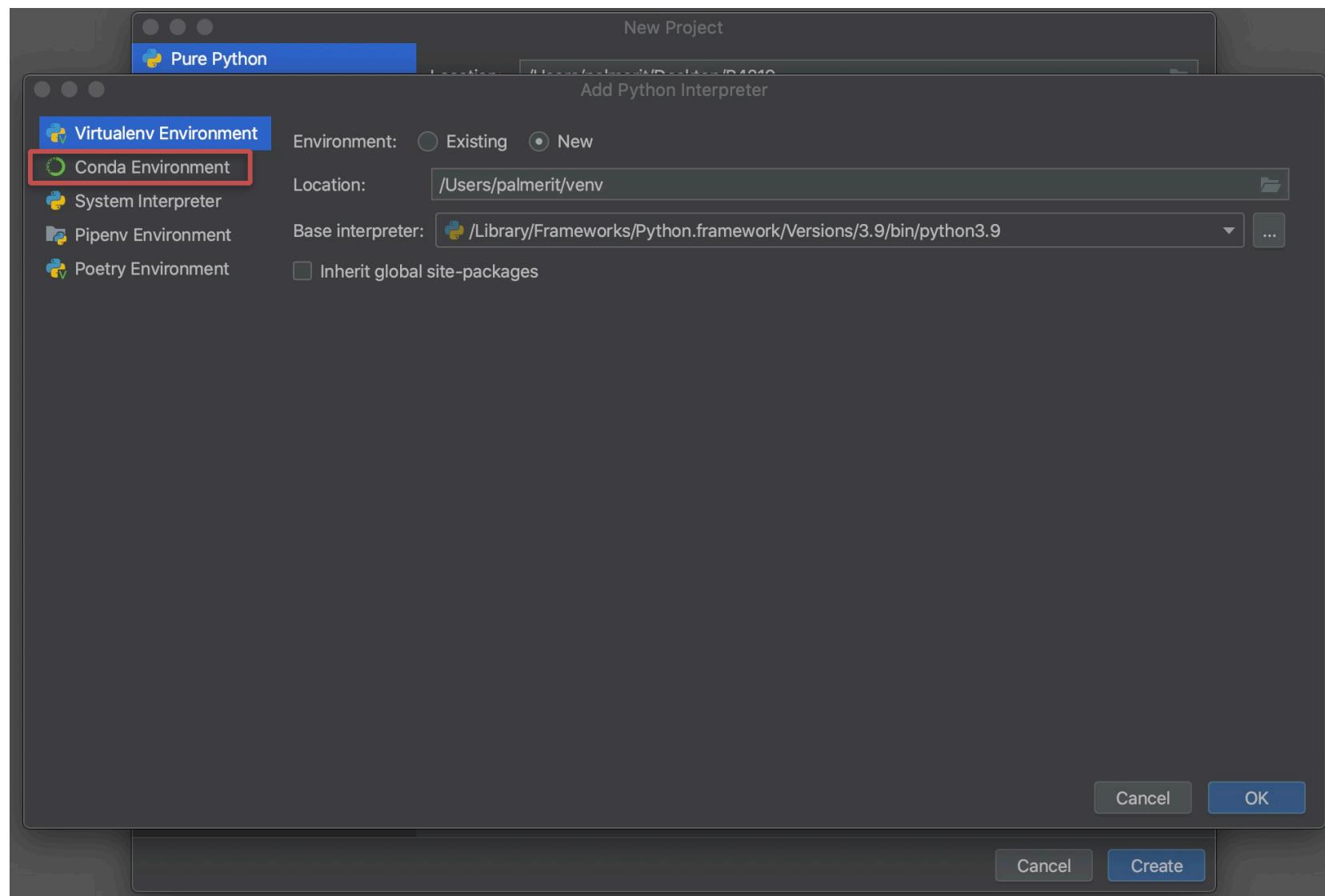
Setting up PyCharm

click "Add Local Interpreter..."



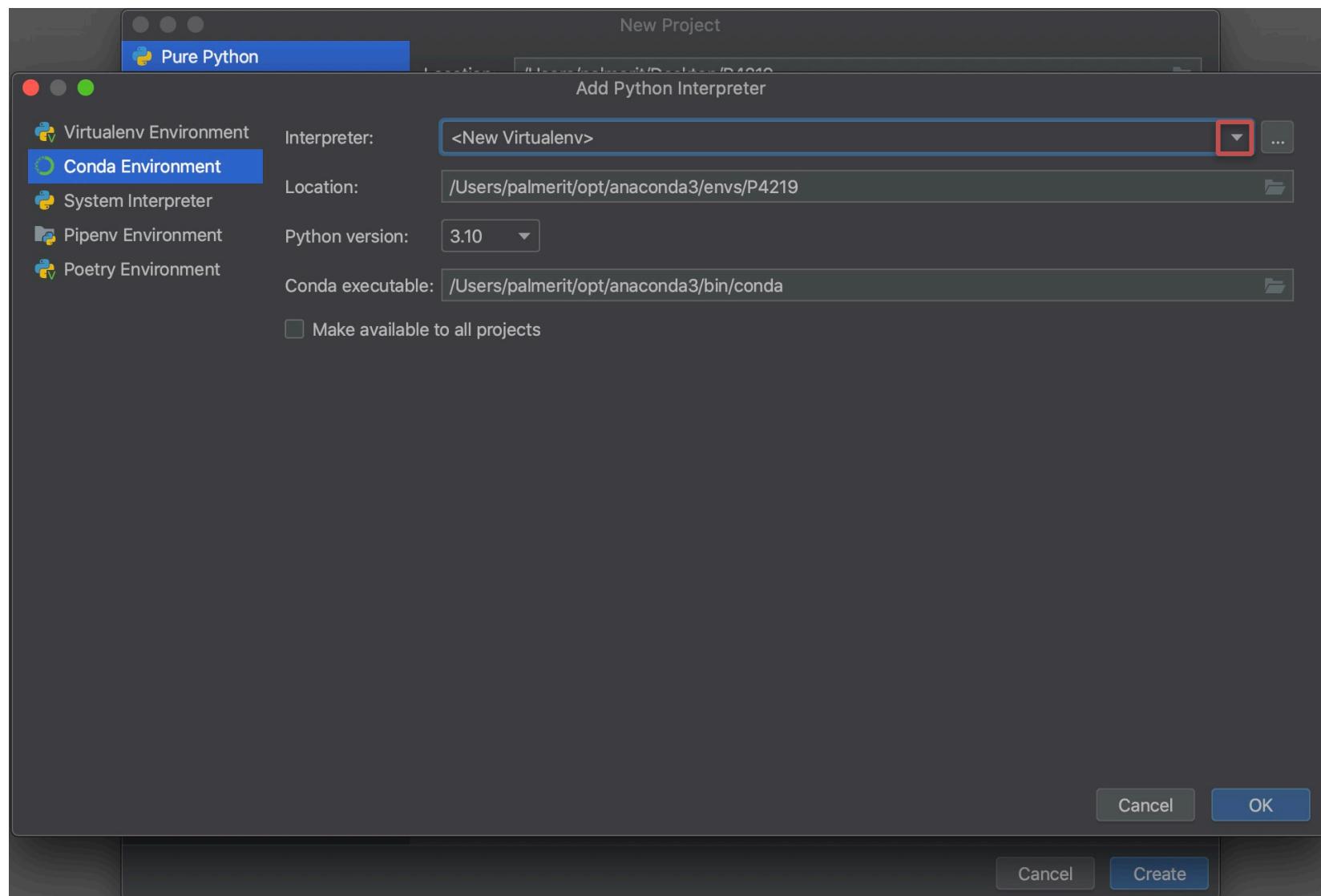
Setting up PyCharm

select "Conda Environment"



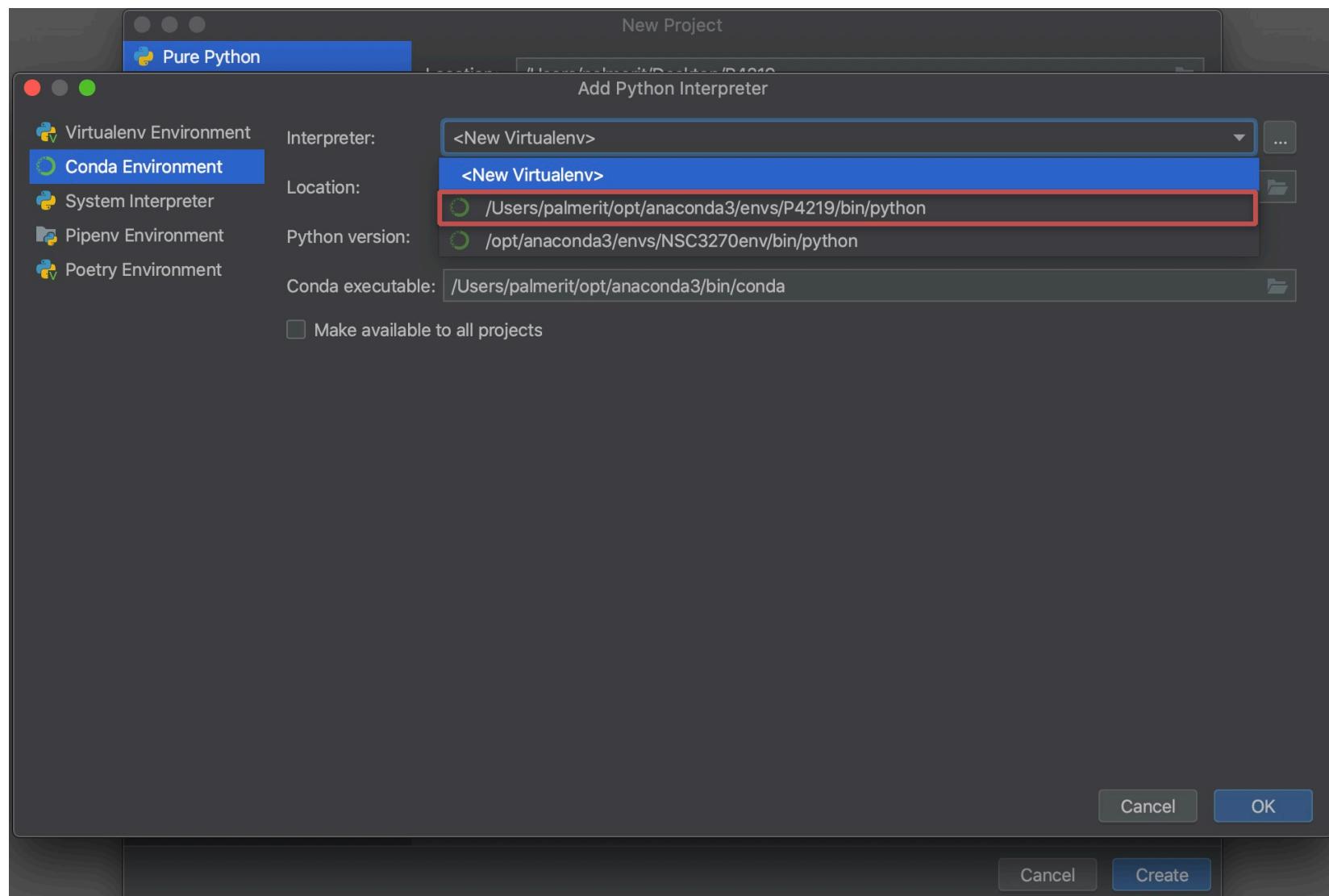
Setting up PyCharm

select Interpreter



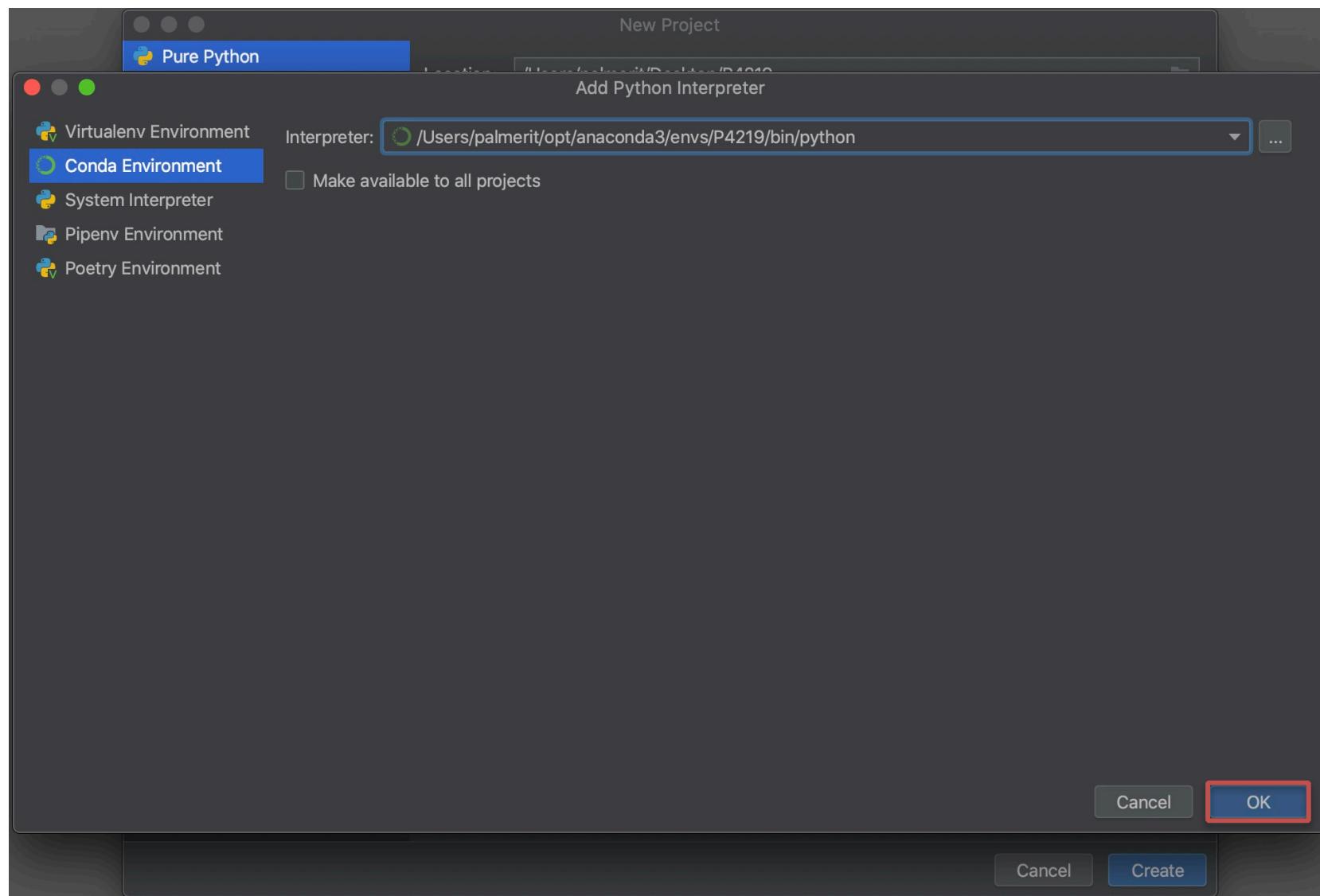
Setting up PyCharm

select the Python in the folder for your virtual environment



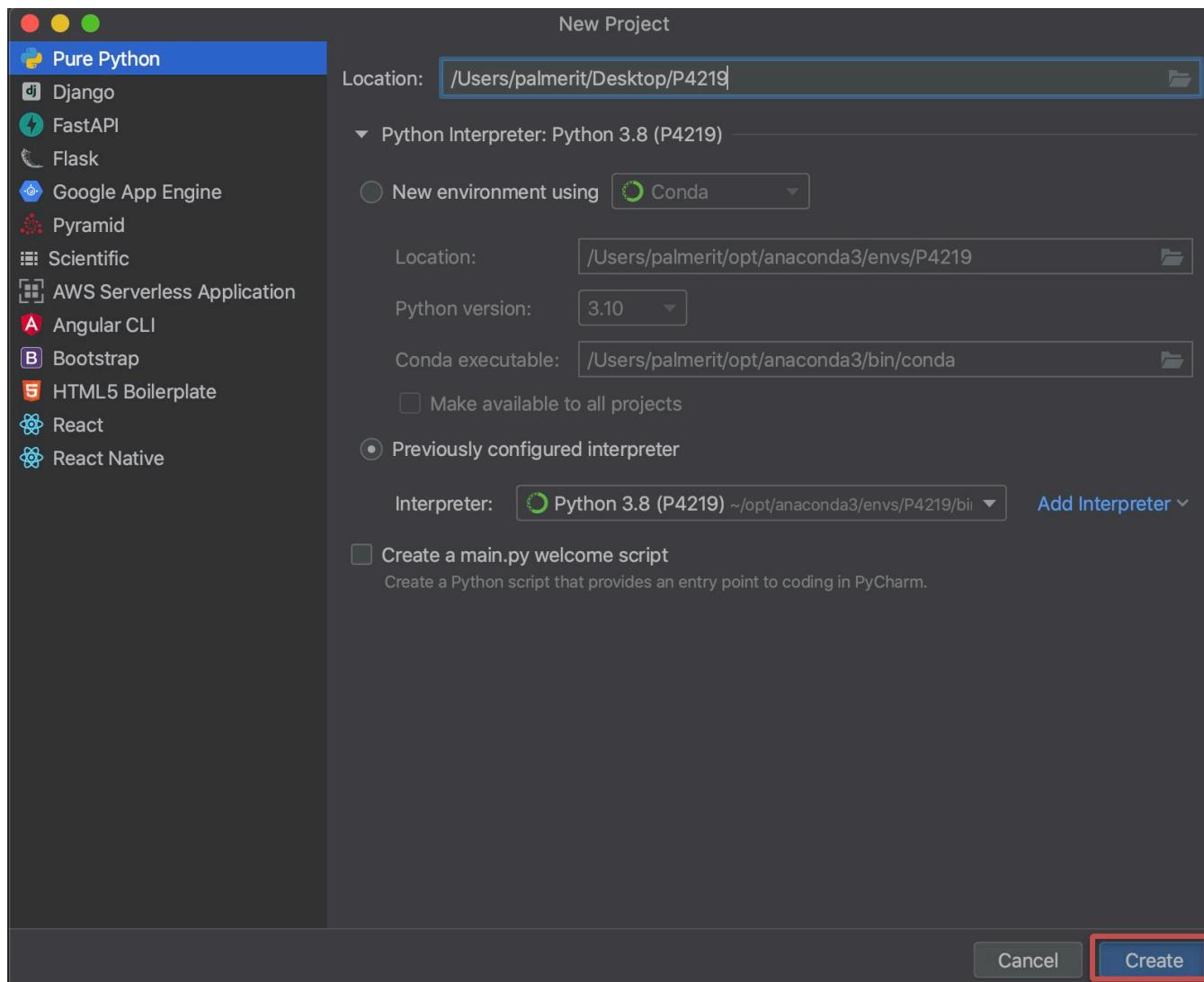
Setting up PyCharm

click "OK"



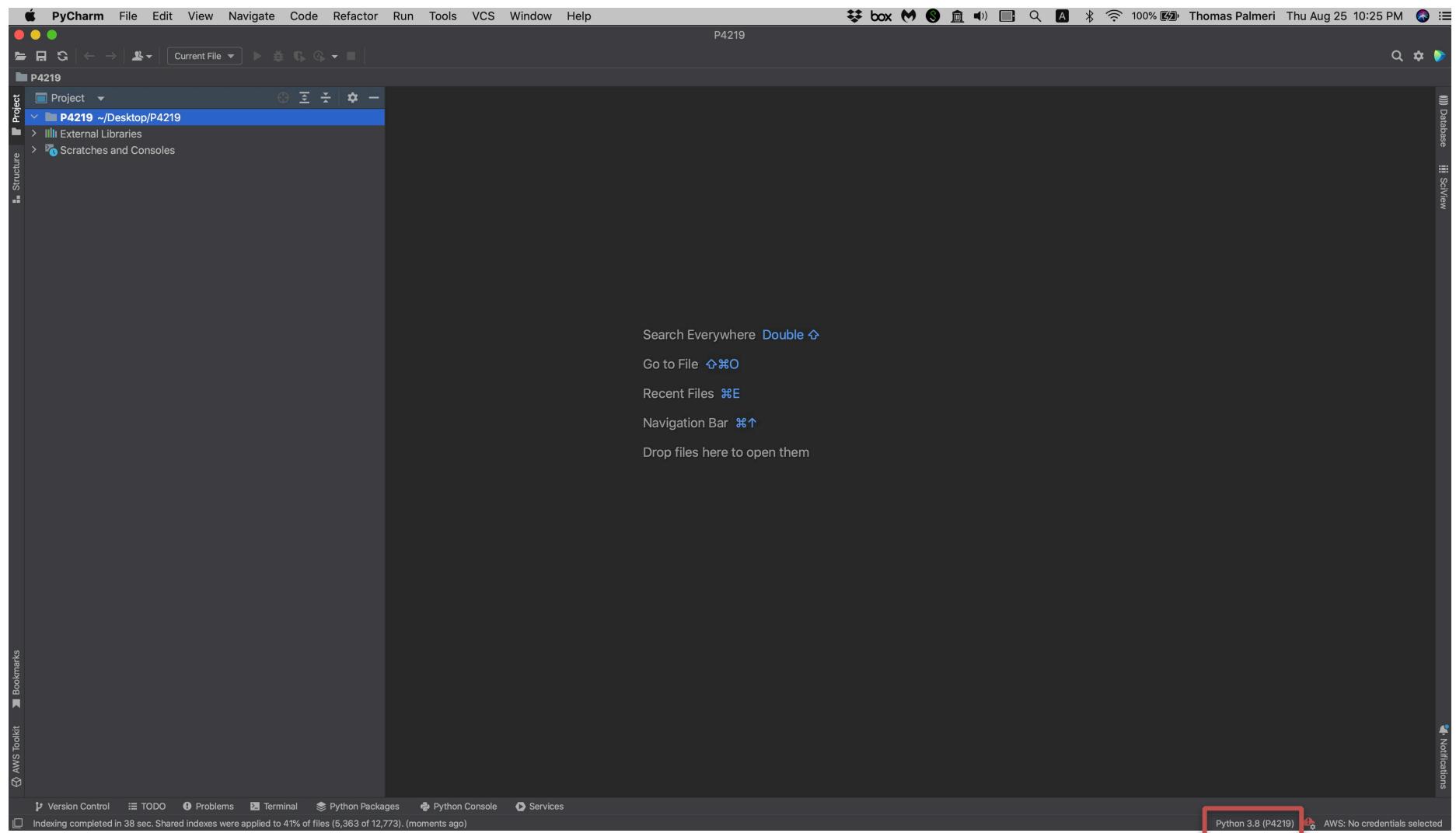
Setting up PyCharm

click "Create"



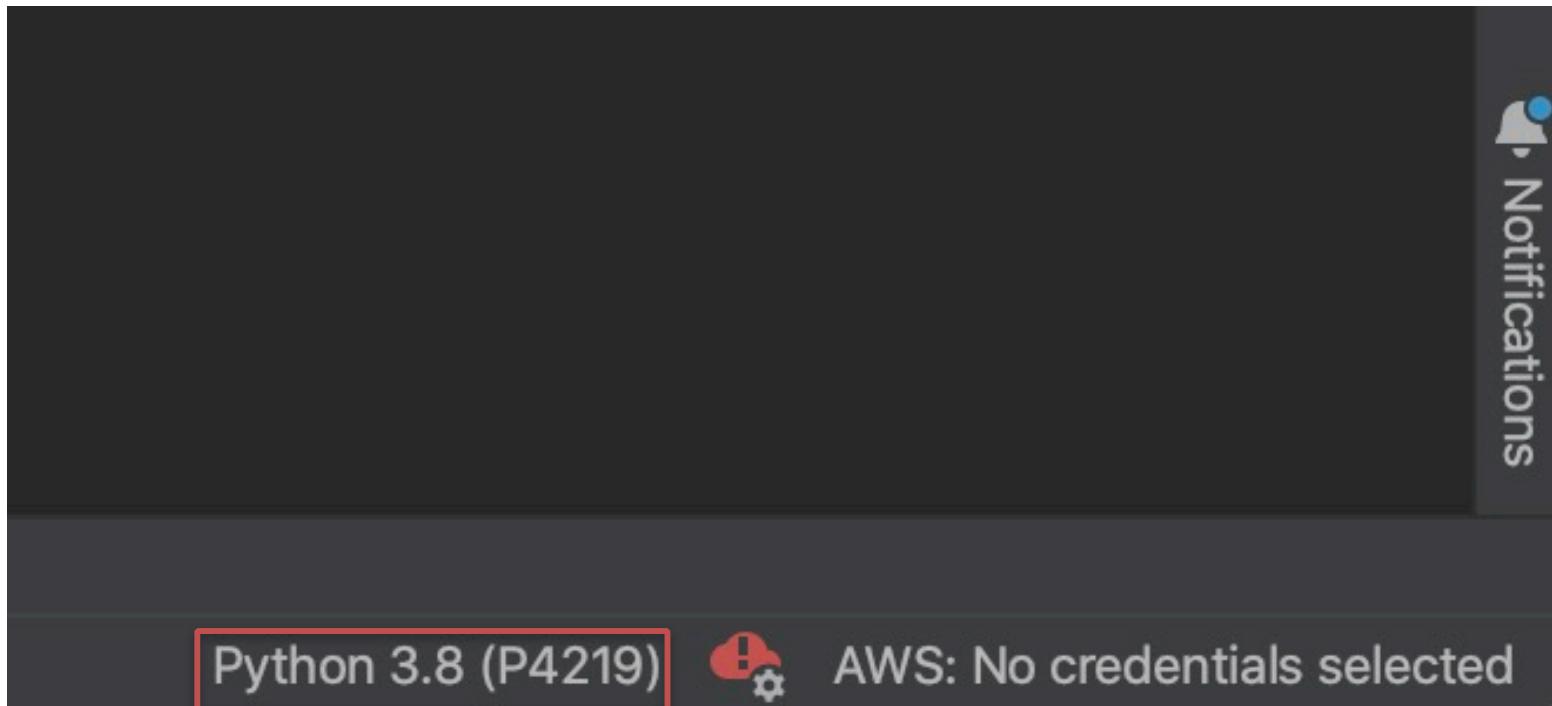
Setting up PyCharm

confirm that your environment is selected in lower right corner



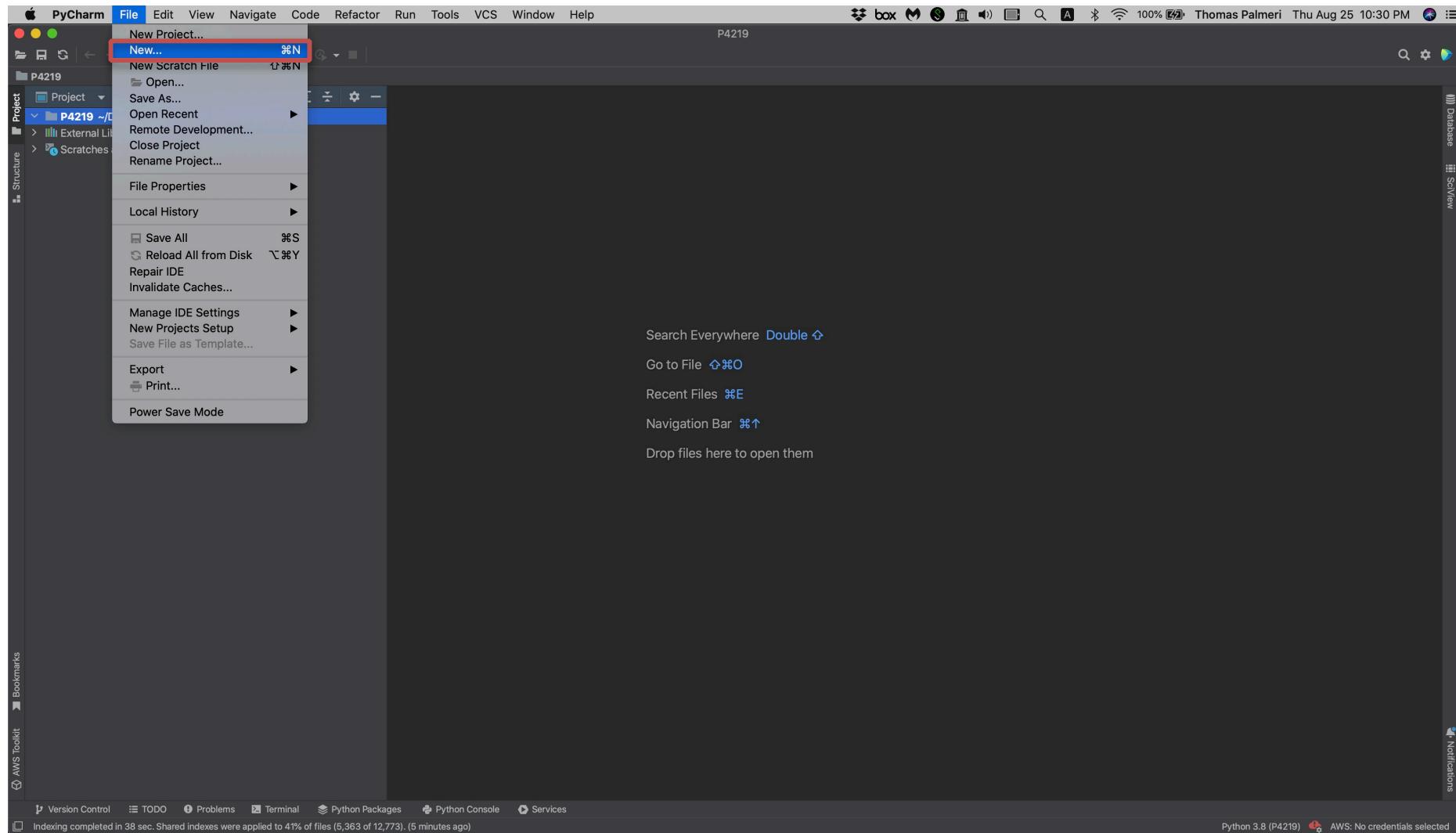
Setting up PyCharm

confirm that your environment is selected in lower right corner



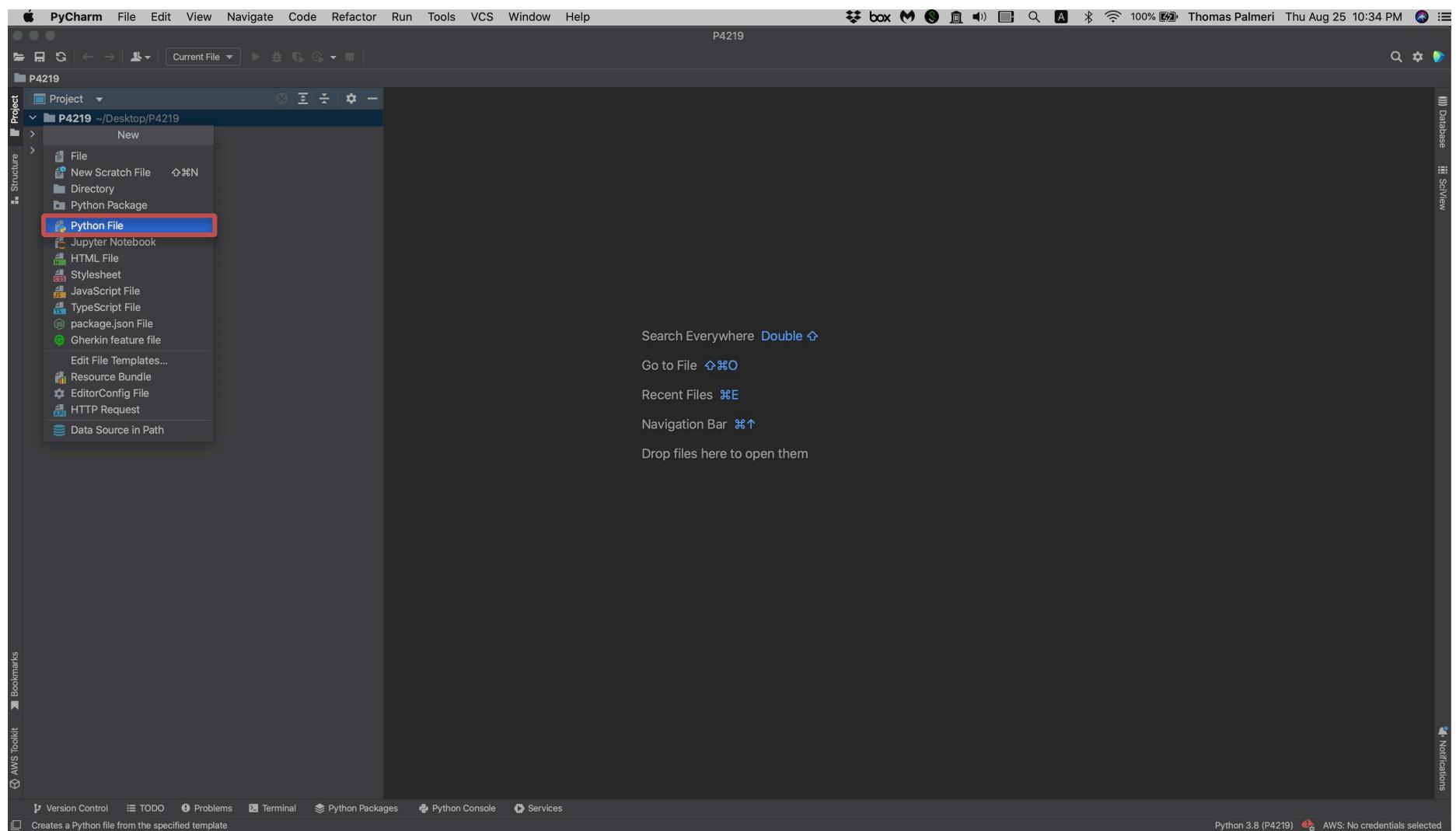
Setting up PyCharm

select "File --> New"



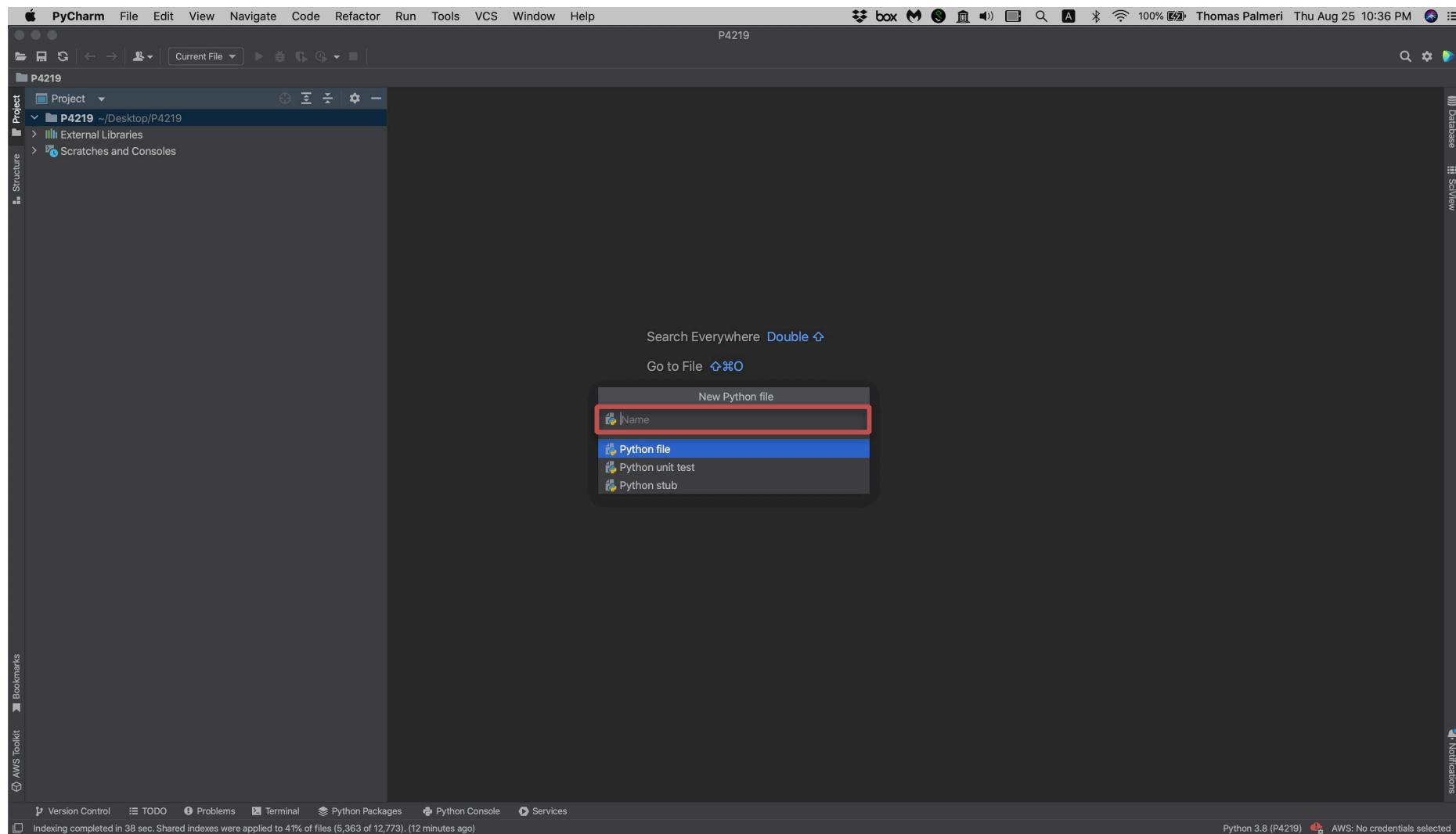
Setting up PyCharm

select "Python File"



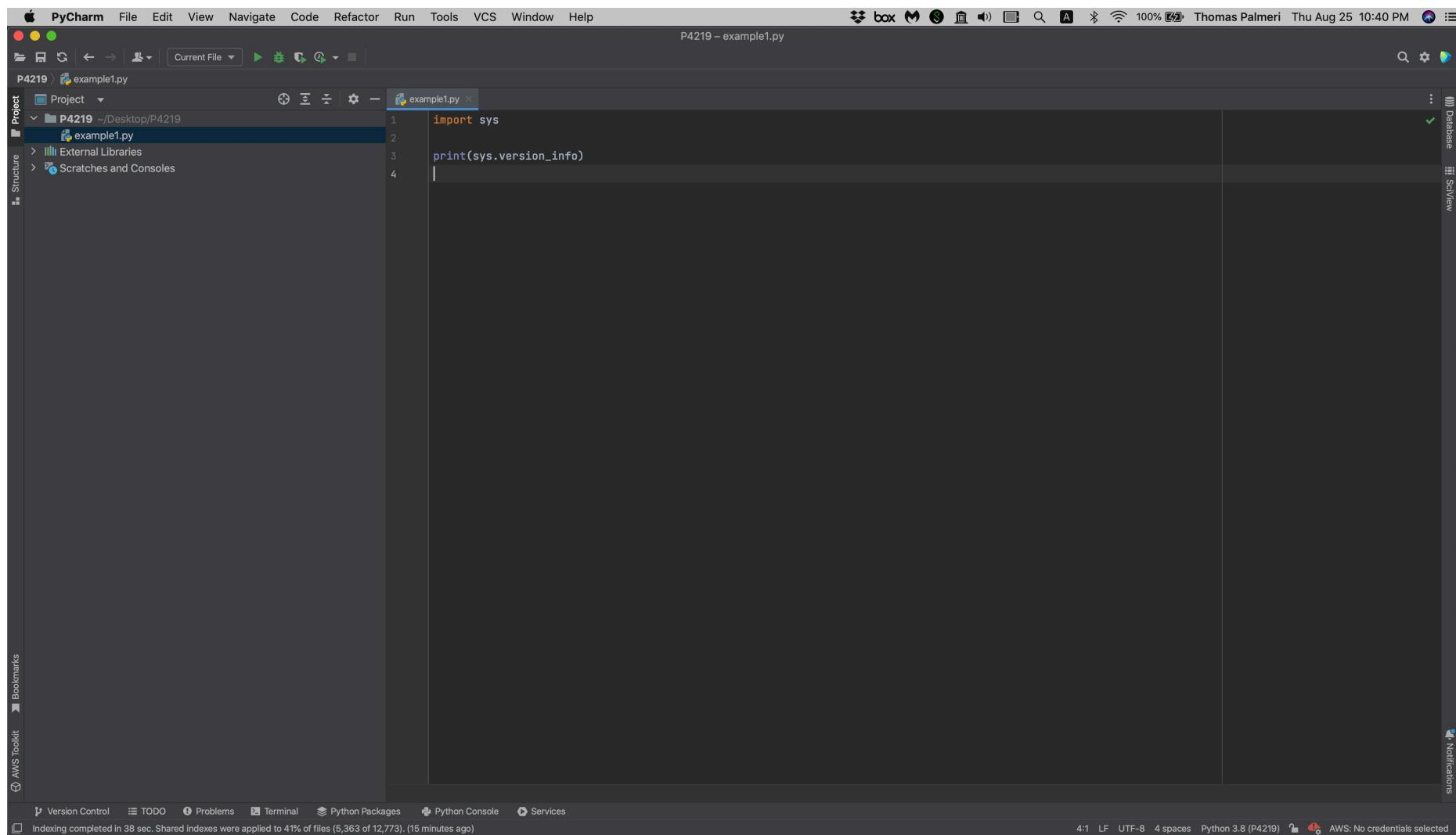
Setting up PyCharm

enter file name (e.g., "example1")



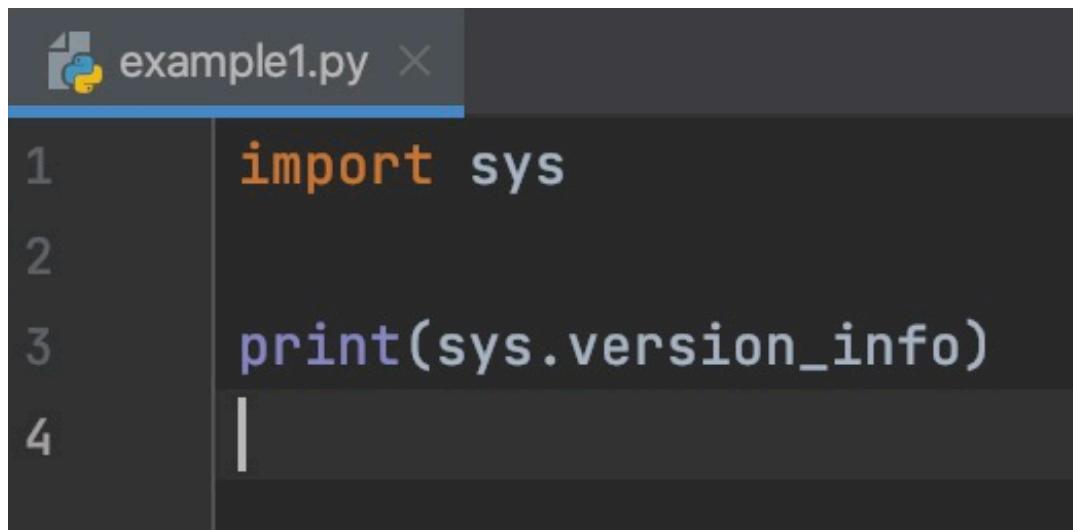
Setting up PyCharm

type in some Python code



Setting up PyCharm

type in some Python code

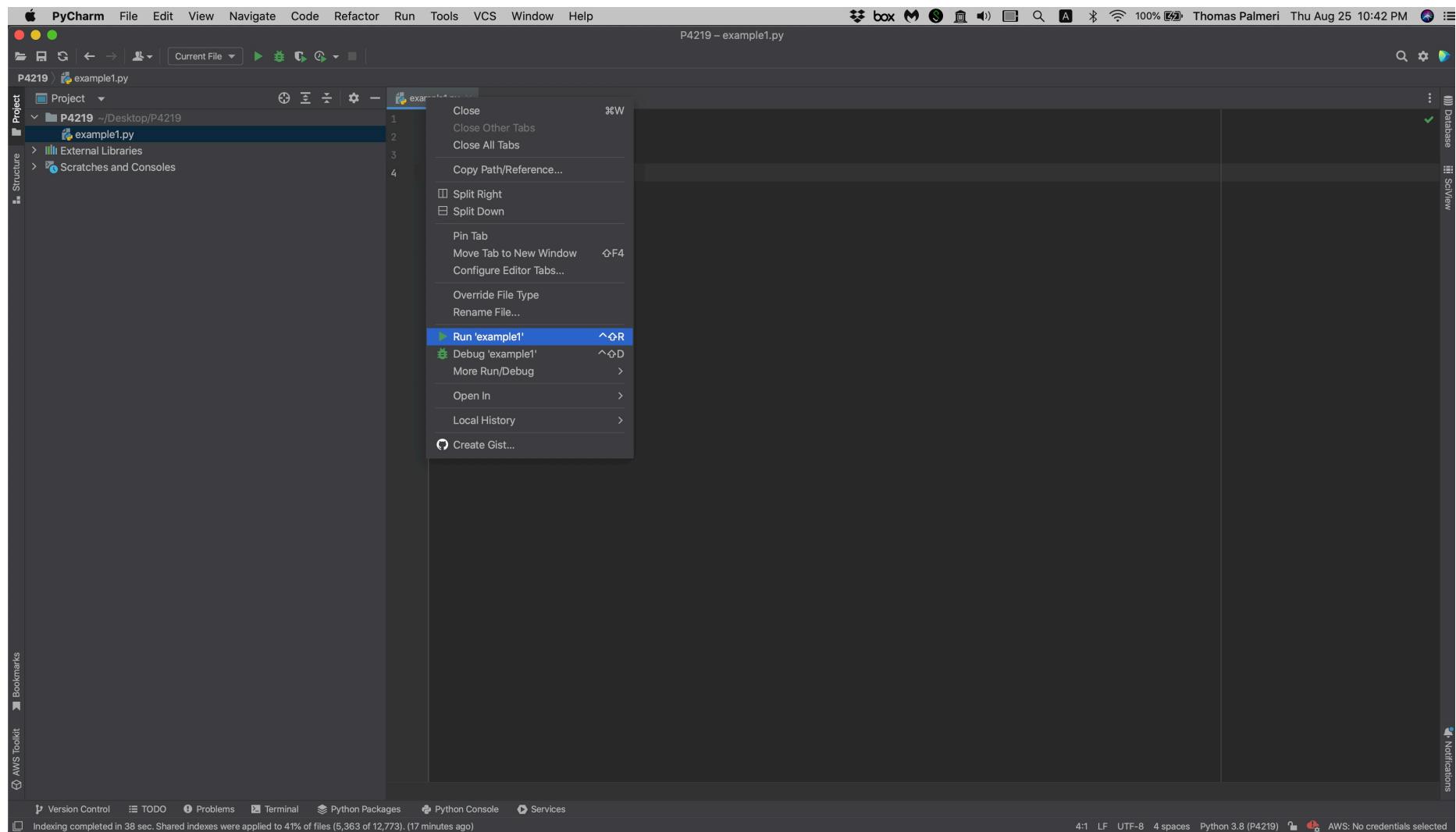


A screenshot of the PyCharm IDE interface. The title bar shows the file name "example1.py". The code editor contains the following Python code:

```
1 import sys
2
3 print(sys.version_info)
4
```

Setting up PyCharm

right click on the tab with the file name, select "Run"



Setting up PyCharm

see the printout from running the program

The screenshot shows the PyCharm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. The title bar indicates the project is "P4219" and the file is "example1.py". The code editor window displays the following Python script:

```
1 import sys
2
3 print(sys.version_info)
4
```

The bottom panel, titled "Run", shows the output of running the script. A red box highlights the output area. The output shows the Python version information:

```
example1 x
↑ /Users/palmerit/opt/anaconda3/envs/P4219/bin/python /Users/palmerit/Desktop/P4219/example1.py
sys.version_info(major=3, minor=8, micro=13, releaselevel='final', serial=0)

Process finished with exit code 0
```

The status bar at the bottom provides system information: 100% battery, Thomas Palmeri, Thu Aug 25 10:43 PM, and AWS: No credentials selected.

Run PyCharm

(or another IDE)

(part of) HOMEWORK 1:

load `TestPsychoPy.py` into PyCharm (or another IDE)
(on Brightspace) and run it

grab a screen shot (after it runs) and submit (with other parts)

Run TestPsychoPy.py

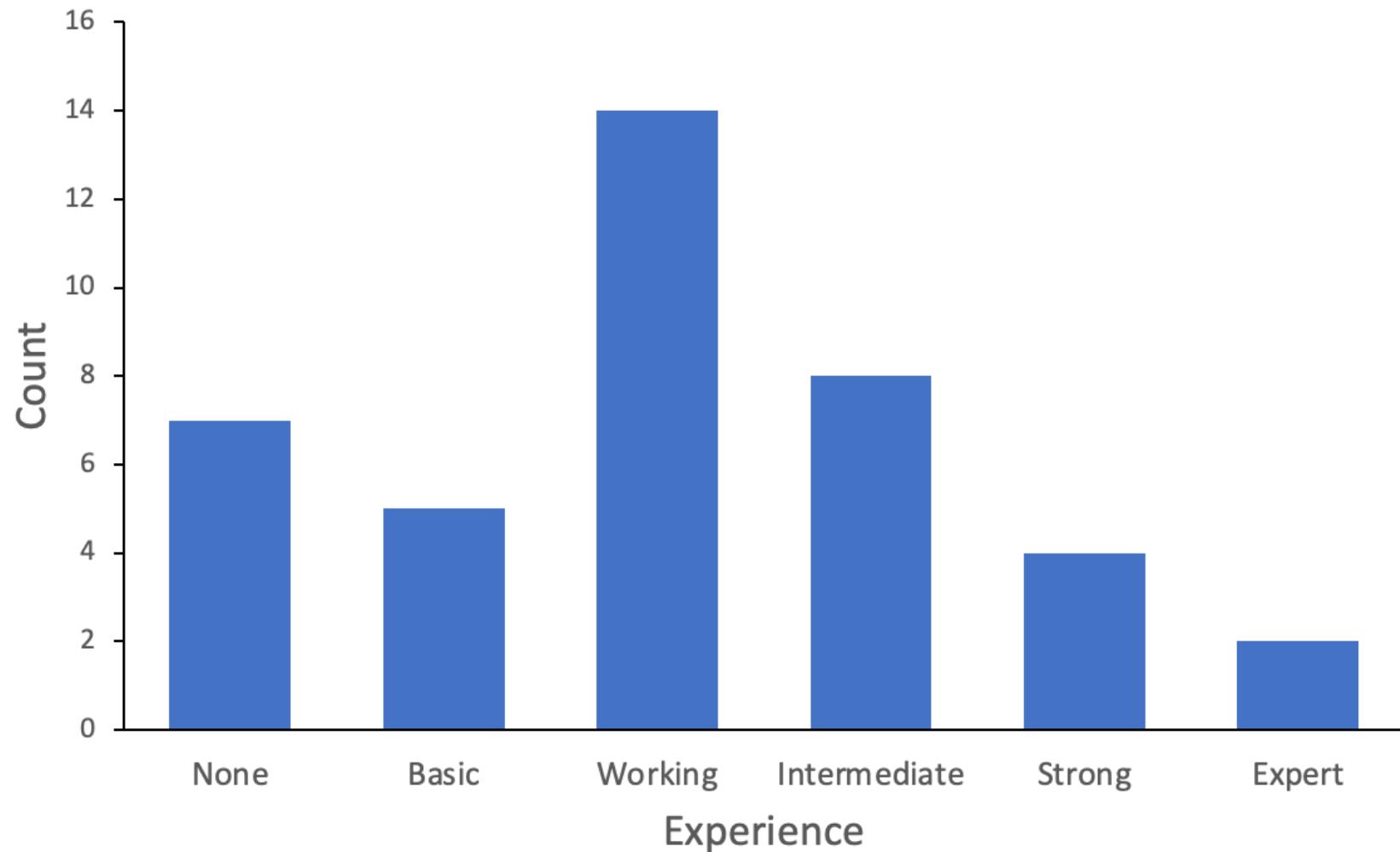
Jupyter Notebooks

we'll use PyCharm later - for now, we'll use Notebooks

Jupyter Notebooks

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

Jupyter Notebooks



Google Colab

<https://colab.research.google.com>

The screenshot shows the Google Colab interface. On the left, there's a sidebar with a 'Table of contents' section containing links to 'Getting started', 'Data science', 'Machine learning', 'More Resources', and 'Featured examples'. Below that is a 'Section' button. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help' menus, along with 'Share', 'Settings', and a profile icon. The main area has tabs for '+ Code', '+ Text', and 'Copy to Drive'. A central box displays 'Welcome to Colab!' and a 'Recent' tab selected in a navigation bar above a list of notebooks. The list includes:

Title	Last opened	First opened	Actions
Index.ipynb	August 22	Jun 8, 2020	🔍 🔍
Index.ipynb	August 22	August 22	🔍 🔍
Test.ipynb	August 21	August 19	🔍 🔍
Welcome To Colaboratory	August 21	Jun 8, 2020	🔗
Negative Neurons - Feature Visualization	Oct 30, 2021	Oct 30, 2021	🔍 🔍

At the bottom of the main area, there's a note about Colab notebooks and a code cell example:

The document you are reading is not a static web page, but an interactive environment called a **Colab notebook** that lets you write and execute code.

For example, here is a **code cell** with a short Python script that computes a value, stores it in a variable, and prints the result:

```
[ ] seconds_in_a_day = 24 * 60 * 60
seconds_in_a_day
```

can upload notebooks (.ipynb files), load notebooks from GitHub, or create a new notebook, and obviously you can then download it

Google Colab

<https://colab.research.google.com>

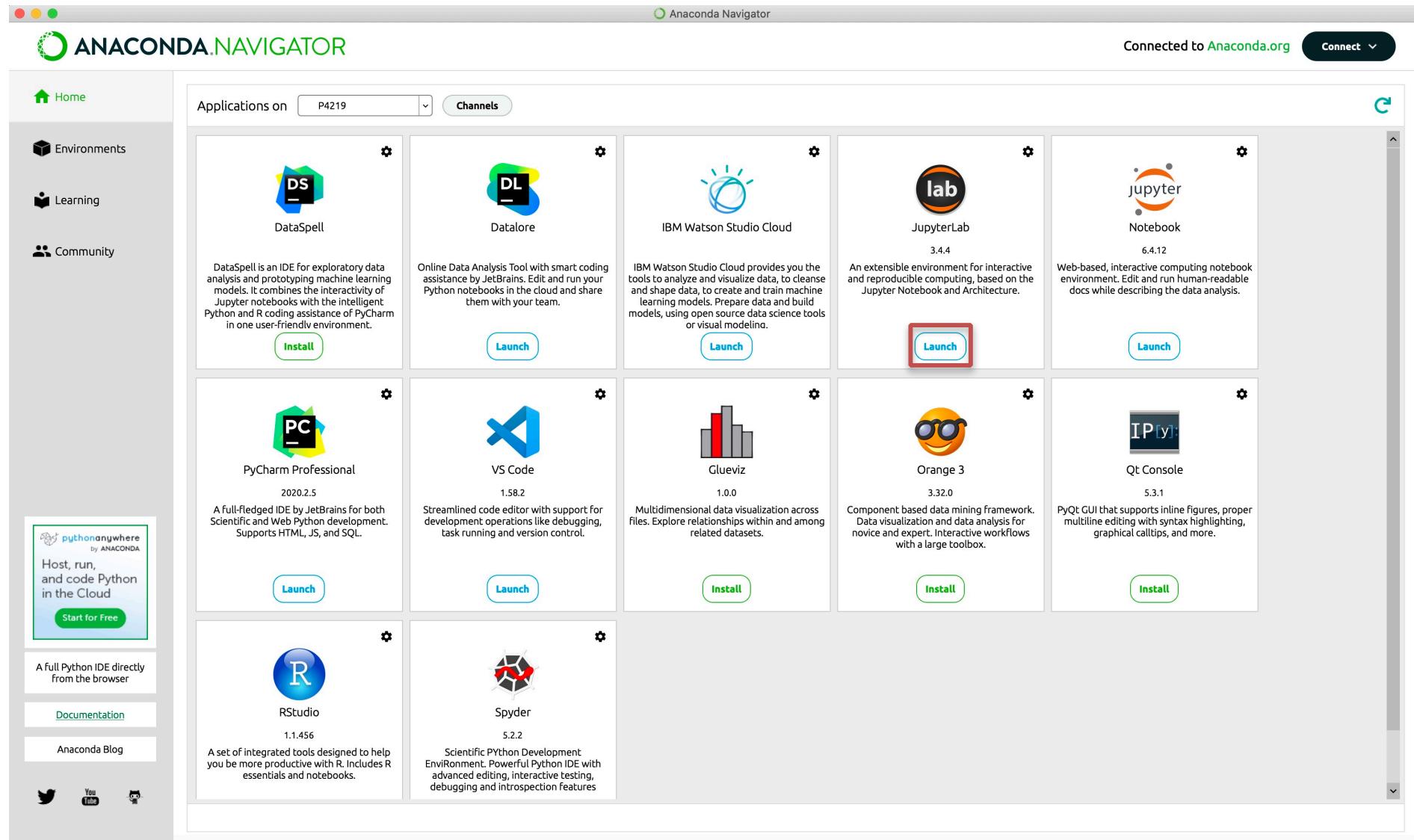
(part of) HOMEWORK 1:

upload JupyterNotebooks.ipynb to Google CoLab
(on Brightspace) and run it

grab a screen shot (after it runs) and submit (with other parts)

we'll talk about how to run in a bit

launch JupyterLab (or Jupyter Notebook)



The screenshot shows the Anaconda Navigator interface. On the left, there's a sidebar with links to Home, Environments, Learning, and Community. A callout box highlights the "pythonanywhere by ANACONDA" section, which says "Host, run, and code Python in the Cloud" and has a "Start for Free" button. Below it, there are links to Documentation and the Anaconda Blog, along with social media icons for Twitter, YouTube, and GitHub.

The main area is titled "Anaconda Navigator" and shows a grid of application cards. The cards include:

- DataSpell**: An IDE for exploratory data analysis and prototyping machine learning models. It combines the interactivity of Jupyter notebooks with the intelligent Python and R coding assistance of PyCharm in one user-friendly environment. [Install](#)
- Datalore**: Online Data Analysis Tool with smart coding assistance by JetBrains. Edit and run your Python notebooks in the cloud and share them with your team. [Launch](#)
- IBM Watson Studio Cloud**: IBM Watson Studio Cloud provides you the tools to analyze and visualize data, to cleanse and shape data, to create and train machine learning models. Prepare data and build models, using open source data science tools or visual modeling. [Launch](#)
- JupyterLab**: An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture. [Launch](#)
- Notebook**: Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis. [Launch](#)
- PyCharm Professional**: A full-fledged IDE by JetBrains for both Scientific and Web Python development. Supports HTML, JS, and SQL. [Launch](#)
- VS Code**: Streamlined code editor with support for development operations like debugging, task running and version control. [Launch](#)
- Glueviz**: Multidimensional data visualization across files. Explore relationships within and among related datasets. [Install](#)
- Orange 3**: Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox. [Install](#)
- Qt Console**: PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more. [Install](#)
- RStudio**: A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks. [Install](#)
- Spyder**: Scientific PYthon Development Environment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features. [Install](#)

At the top right of the main area, it says "Connected to [Anaconda.org](#)".

you can also run from command line:



Windows 10 - from Command Prompt

C:\> jupyter notebook

MAC OSX - from Terminal

\$ jupyter notebook

if you know how to activate environments
from the command line

you can also run from command line:



JupyterLab

2.1.5

Windows 10 - from Command Prompt

C:\> jupyter lab

MAC OSX - from Terminal

\$ jupyter lab

if you know how to activate environments
from the command line

you can also run from command line:



JupyterLab

2.1.5

Windows 10 - from Command Prompt

C:\> jupyter lab

MAC OSX - from Terminal

\$ jupyter lab

if you know how to activate environments
from the command line

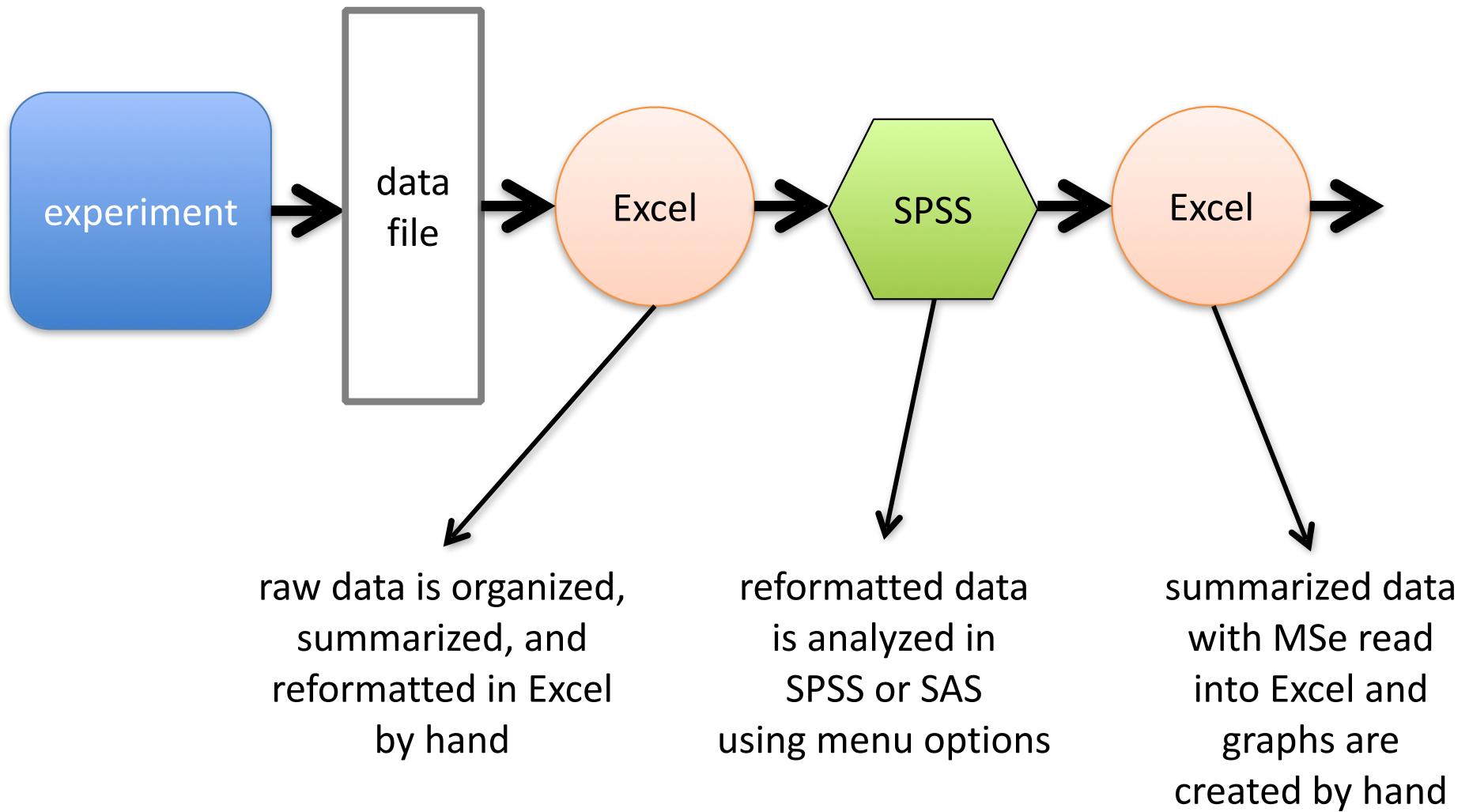
you can also run from the terminal
within an IDE like PyCharm

akin to lab digital notebooks

Jupyter Notebooks (are great) as Scripts

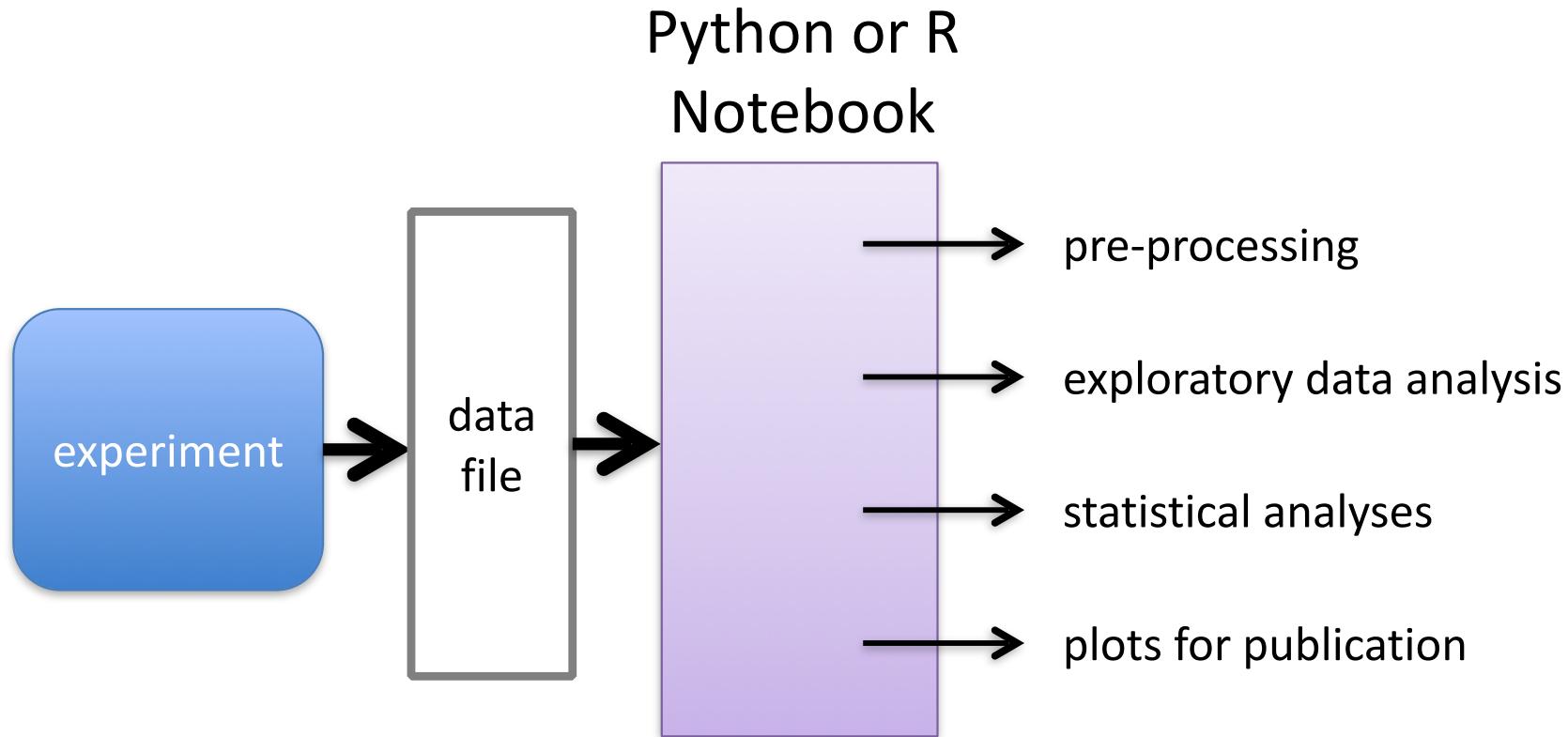
- Jupyter Notebooks are great for trying out snippets of Python code (as is the Python Console, which we will use within the PyCharm IDE)
- “Scripts” are a series of commands (though you can define functions within them). Scripts are not stand-alone programs nor are they modules/packages that can be used by other programs.
- Jupyter Notebooks are fantastic for fully documenting a data analysis / modeling pipeline. They are a computational equivalent of a paper lab notebook. Preferred over more ad hoc analysis pipelines commonly used.

common (old) data analysis pipeline



maybe (hopefully) notes are kept
(paper or electronic, often separate from analyses)

Best Practice : Jupyter Notebooks



promises more reliable, robust, reproducible research

share data and Jupyter Notebook publicly (lab, GitHub, OSF)

using Jupyter

JupyterNotebooks.ipynb

using JupyterLab (or Jupyter Notebook)

make sure you reset the kernel and clear all outputs and rerun your code before turning in

cells can be rerun in any order (note numbers next to cells), but we will run your code from top to bottom

both .py and .ipynb files are text files

TestPsychoPy.py

```
Fall2022 — bash — 87x41
Tom-MacBook-Pro-2020:Fall2022 palmerit$ 
Tom-MacBook-Pro-2020:Fall2022 palmerit$ 
Tom-MacBook-Pro-2020:Fall2022 palmerit$ 
Tom-MacBook-Pro-2020:Fall2022 palmerit$ cat TestPsychoPy.py
#
# TestPsychoPy.py
#
# drawing example
#
from psychopy import visual, core, event
import numpy as np

def main(mywin):
    # https://www.psychopy.org/api/visual/line.html#psychopy.visual.Line
    el1 = visual.Line(win=mywin, start=[+.3,-.2], end=[-.4,-.3], lineWidth=20, lineColor=r='blue')
    el1.autoDraw = True
    print('el1 : ', dir(el1))
    #el1.draw()
    mywin.flip(); core.wait(0.5)

    # https://www.psychopy.org/api/visual/circle.html#psychopy.visual.Circle
    el2 = visual.Circle(win=mywin, radius=.1, pos=[-.2,.1], edges=128, fillColor='green')
    el2.autoDraw = True
    print('el2 : ', dir(el2))
    #el2.draw()
    mywin.flip(); core.wait(0.5)

    # https://www.psychopy.org/api/visual/rect.html#psychopy.visual.Rect
    el3 = visual.Rect(win=mywin, size=[.2,.1], pos=[+.3,+.2], lineWidth=10, lineColor='#F016A3')
    print('el3 : ', dir(el3))
    el3.autoDraw = True
    #el3.draw()
    mywin.flip(); core.wait(0.5)

    # https://www.psychopy.org/api/visual/shapestim.html#psychopy.visual.ShapeStim
```

JupyterNotebooks.ipynb

```
Jupyter — bash — 87x41
Tom-MacBook-Pro-2020:Jupyter palmerit$ 
Tom-MacBook-Pro-2020:Jupyter palmerit$ cat JupyterNotebooks.ipynb
{
  "cells": [
    {
      "cell_type": "markdown",
      "id": "ecf4a5b0-d17f-4391-a06c-7b34f3593f01",
      "metadata": {},
      "tags": []
    },
    "source": [
      "#< Introduction to Jupyter Notebooks\n",
      "- two main types of cells - Markdown and Code\n",
      "- click to highlight a cell\n",
      "- double-click to edit a formatted Markdown cell\n",
      "- click the \"Run\" button above to execute highlighted cell\n",
      "- or do Shift-Return (Mac) or Shift-Enter (PC)\n",
      "- Cell -> Run All to run all cells in the notebook\n",
      "- Kernel -> Restart and then Cell -> Run All or Kernel -> Restart & Run All (reset everything and rerun)\n",
      "\n",
      "before finalizing any Jupyter notebook and turning it in for homework, please make sure you do a Kernel -> Restart to make sure your code runs completely (since when I run your code to grade it, everything will be similarly reset when I load it on my computer)"
    ],
    {
      "cell_type": "code",
      "execution_count": null,
      "id": "dbf639a1-062f-4931-864c-6d9bd3fffa30",
      "metadata": {},
      "outputs": [],
      "source": [
        "x = 1\n",
        "print(\"x = \", x)"
      ]
    },
    {
      "cell_type": "markdown",
      "id": "12f6a269-a5c0-433d-997b-7a016586537e",
```

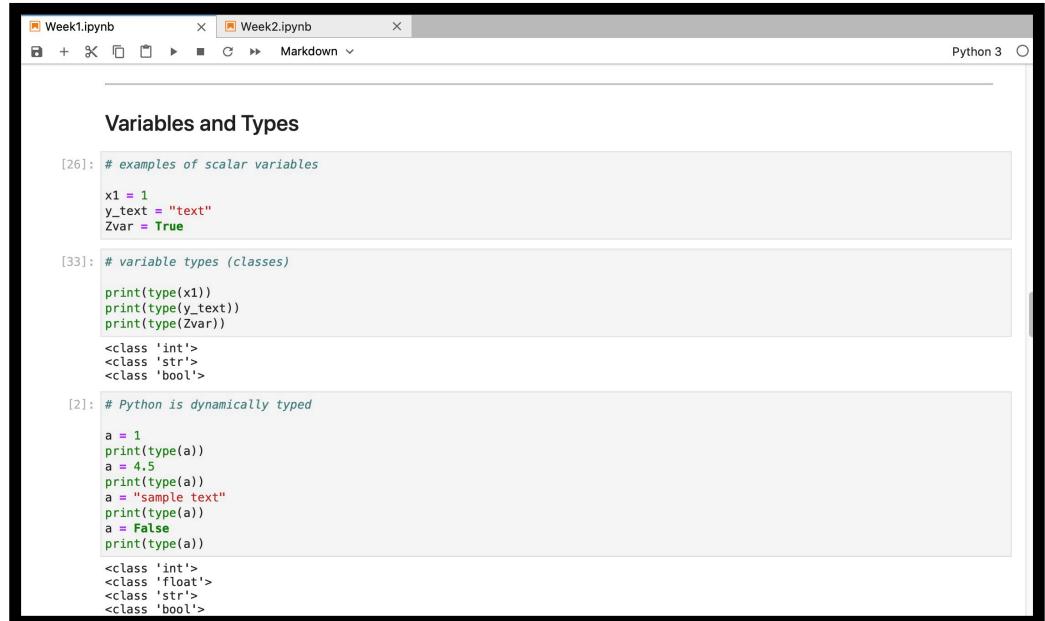
(JSON format)

swap between lecture slides and Python slides

Jupyter or PyCharm

Numeric Types

- int types
`x = 1`
`y = 3`
- float types
`x = 1.`
`y = 3.4`



The screenshot shows a Jupyter Notebook interface with two tabs at the top: "Week1.ipynb" and "Week2.ipynb". The "Week2.ipynb" tab is active. The notebook contains three code cells:

- Cell [26]:

```
# examples of scalar variables
x1 = 1
y_text = "text"
zvar = True
```
- Cell [33]:

```
# variable types (classes)
print(type(x1))
print(type(y_text))
print(type(zvar))
<class 'int'>
<class 'str'>
<class 'bool'>
```
- Cell [2]:

```
# Python is dynamically typed
a = 1
print(type(a))
a = 4.5
print(type(a))
a = "sample text"
print(type(a))
a = False
print(type(a))
<class 'int'>
<class 'float'>
<class 'str'>
<class 'bool'>
```

sometimes I will talk about things on the slides,
sometimes things in the code

you will need to review both the slides and the code
(I will sometimes skim over details in one or the other)