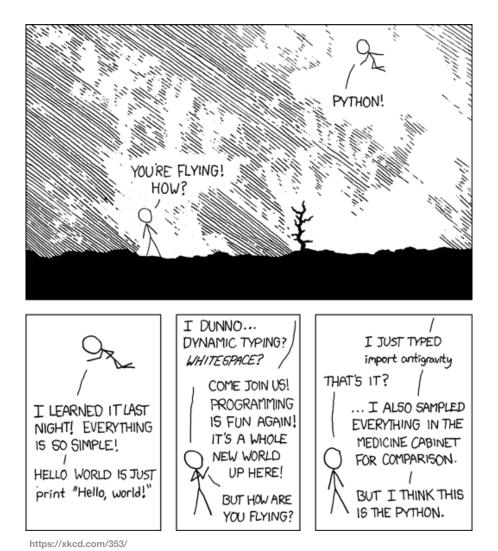
# Getting Started running Python

Virtual environments, installing packages, and running code



Talley Lambert







## Goals for this Talk

### **Understand**

- 1) What Python is
- 2) Where it "lives" on your computer and where it finds code to run
- 3) What packages are, and how they bring in additional functionality.
- 4) What virtual environments are, and how they help us isolate installed packages.

#### Know how to

1) Use **uv** to manage python, virtual environments, and packages; and use it to run code.



This is an opinionated introduction.

There are <u>many</u> different ways to do this stuff.

I'll occasionally provide my personal preference/recommendation

If you already know and like a different approach, this doesn't invalidate that :)



"Python is an interpreted, object-oriented, high-level programming language with dynamic semantics."



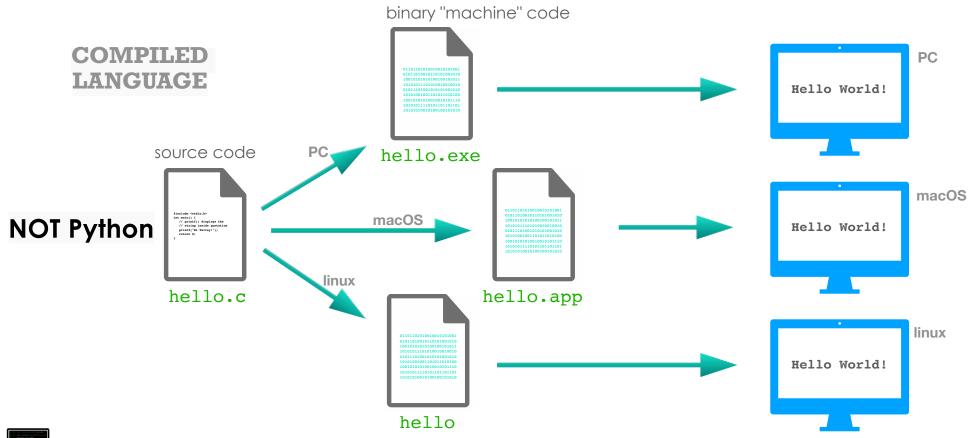
interpreted

#### Tl;dr:

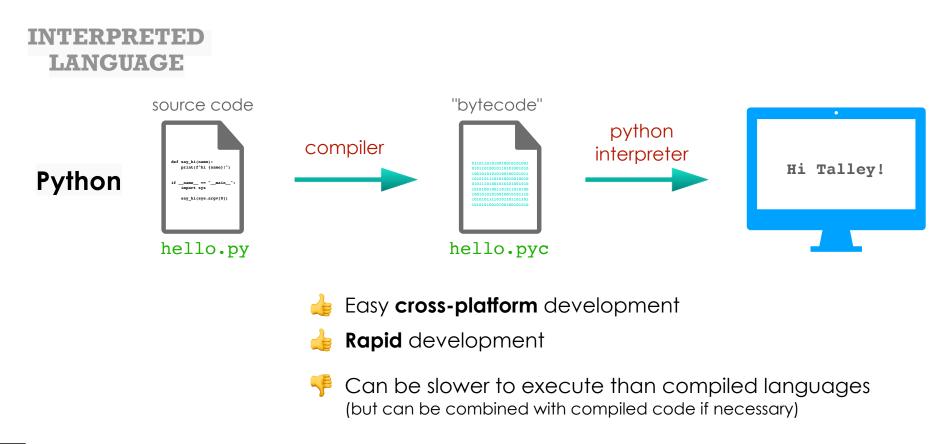
With Python, you share and run "regular" text files. Python does all the OS-specific magic at "runtime".



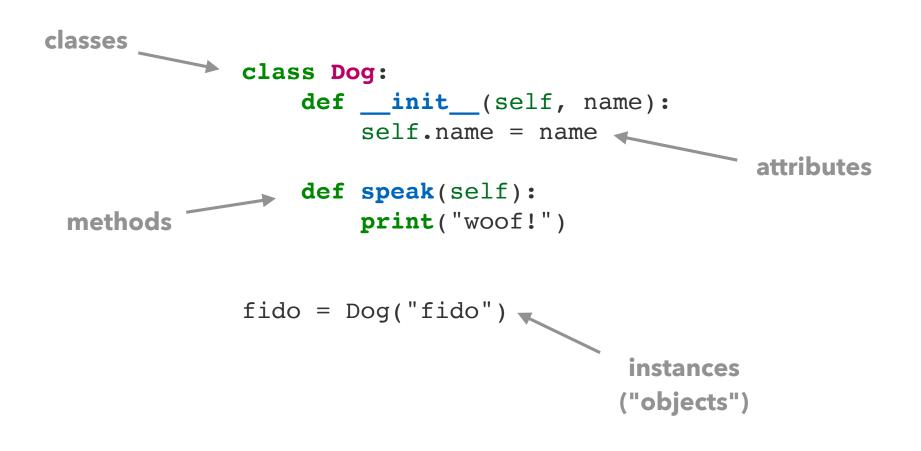
### interpreted



### interpreted



### object-oriented



### high-level

"high-level" ≅

- easy to understand
- closer to human linguistics
- farther from machine code
- Memory management (you don't have to literally allocate and deallocate memory)
- 👍 Dynamic typing
- Built-in data structures (list, dict, set, tuple, etc...) <u>https://docs.python.org/3/tutorial/datastructures.html</u>

### dynamic

```
# make x a string
# no type declaration required!
x = 'hi'
# change it to an integer
# no problem!
x = 2
```

Note: if you want types, there are options as well:

https://docs.python.org/3/library/typing.html





why do scientists use it?

- ✓ Easy to learn, read, and write
- ✓ "Batteries included" (standard library has lots of functionality)
- ✓ Huge community of packages, particularly in data science
- Rapid development (fast edit-test-debug cycle)
- ✓ Totally free & open-source!
- ✓ "Glue-language" (easy to integrate lower level languages)



### Weaknesses

# Faster to develop, but can be slower to execute compared to compiled languages

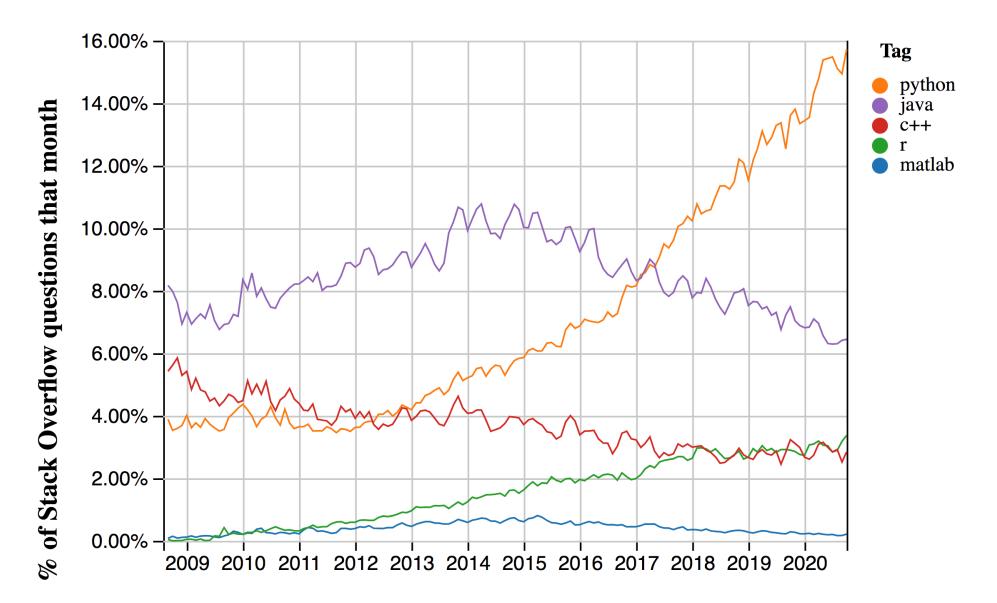
(But... see C-extensions, cython.org, & numba.org, etc...)

Decreased memory efficiency

Parallelism & concurrency requires some workarounds (But: many solutions exist... see <u>dask.org</u>)



## So hot right now...



مەل**ل**ىلىد

Year

**Terms** 

python interpreter	The <b>program</b> that actually "runs" (i.e. parses, compiles, interprets) python code. (ex: "CPython 3.8")
module	An organizational unit of python code. Usually a <b>single file</b> ending in <b>.</b> py that contains python definitions and expressions,
package	A collection of modules. Usually, this is a <b>folder</b> of python modules that also contains an <u>initpy</u> file. "Package" also frequently refers to an installable python library/application (e.g. numpy, matplotlib, pandas)
package manager	A <b>program</b> that automates the installation, updating and removal of packages (e.g. pip, conda)
virtual environment	An <b>isolated collection</b> of packages, settings, and an associated python interpreter, that allows multiple different collections to exist on the same system
environment manager	A <b>program</b> that automates the creation and deletion of virtual environments (e.g. conda, virtualenv, venv)

արիլո

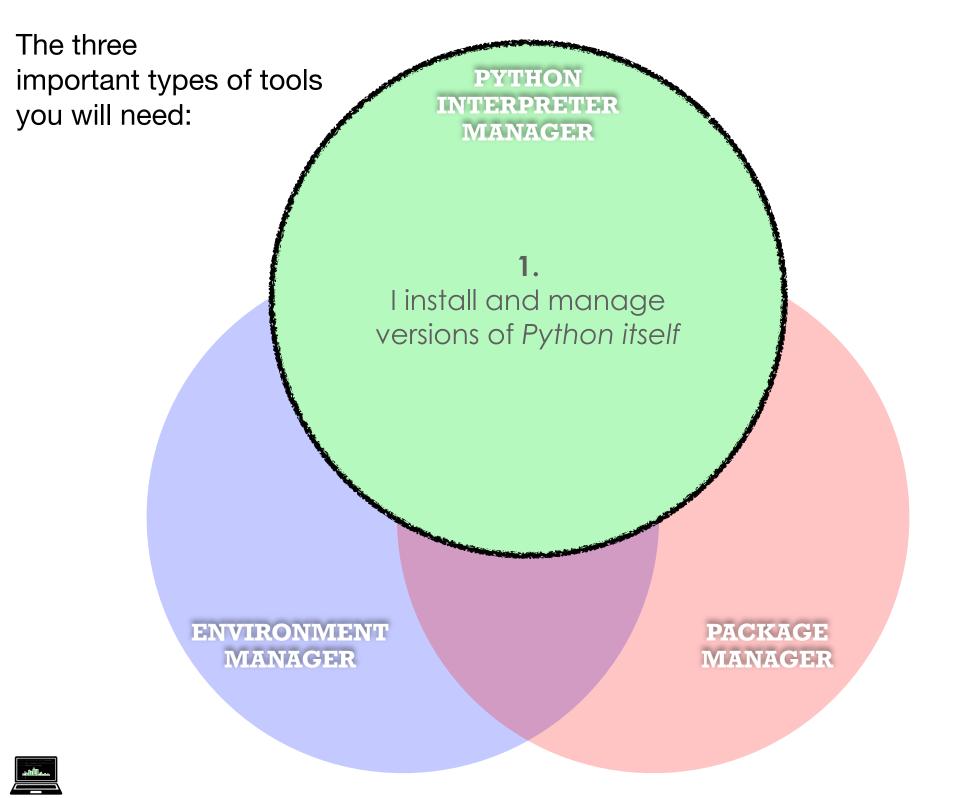
The three important types of tools you will need:











The three important types of tools you will need:

PYTHON INTERPRETER MANAGER

**2.** I create and manage virtual environments

ENVIRONMENT MANAGER PACKAGE MANAGER The three important types of tools you will need:

PYTHON INTERPRETER MANAGER

> **3.** I install and manage Python packages

ENVIRONMENT MANAGER PACKAGE MANAGER

## What does it mean to run code in python?

#### **Executing a script/program/file**

```
# ~/Desktop/hi.py
```

print("hello world!")

\$ python ~/Desktop/hi.py
hello world!

#### Using an interactive console or notebook

```
$ python
Python 3.8.5 | packaged by conda-forge
[Clang 10.0.1 ] on darwin
>>> print("hello world!")
hello world!
```

## Python is modular

One Python module (file) can **import** functionality from another module (file)

# ~/Desktop/do\_math.py

from numpy import add

print(add(1, 2))

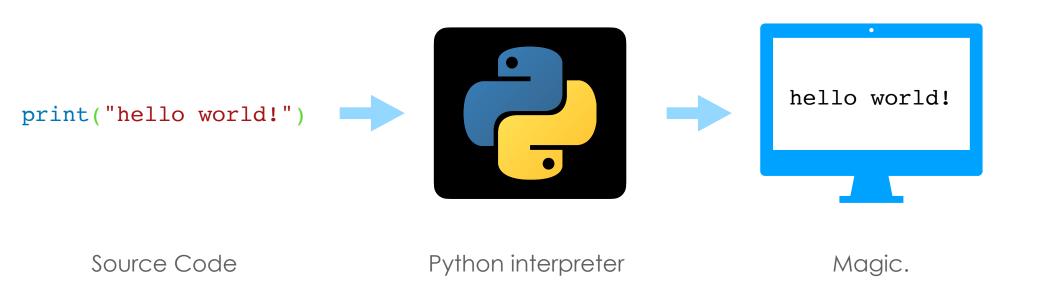
\$ python ~/Desktop/do\_math.py
3

<u>ساتامہ</u>

### What is the Python interpreter?

### \$ python

The **Python interpreter** is a program that parses and compiles source code (human readable code) into "bytecode" (a lower level representation) that is then "interpreted" (executed) by the python "virtual machine"



### What is the Python interpreter?

### \$ python

The **Python interpreter** is a program that parses and compiles source code (human readable code) into "bytecode" (a lower level representation) that is then "interpreted" (executed) by the python "virtual machine"

### Where does it live?

(It can live almost anywhere)

\$ which python
python not found

python is /usr/bin/python # mac ≤ Catalina 10.15

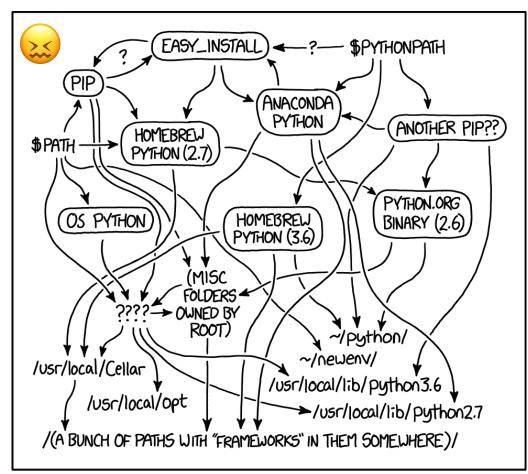
\$ python --version
Python 2.7.16



Python 3 has been out since 2008 Python 2 is end-of-life as of Jan. 2020

For this and many other reasons... don't use your system python (if you have one)!

## OK, then how should I get Python?



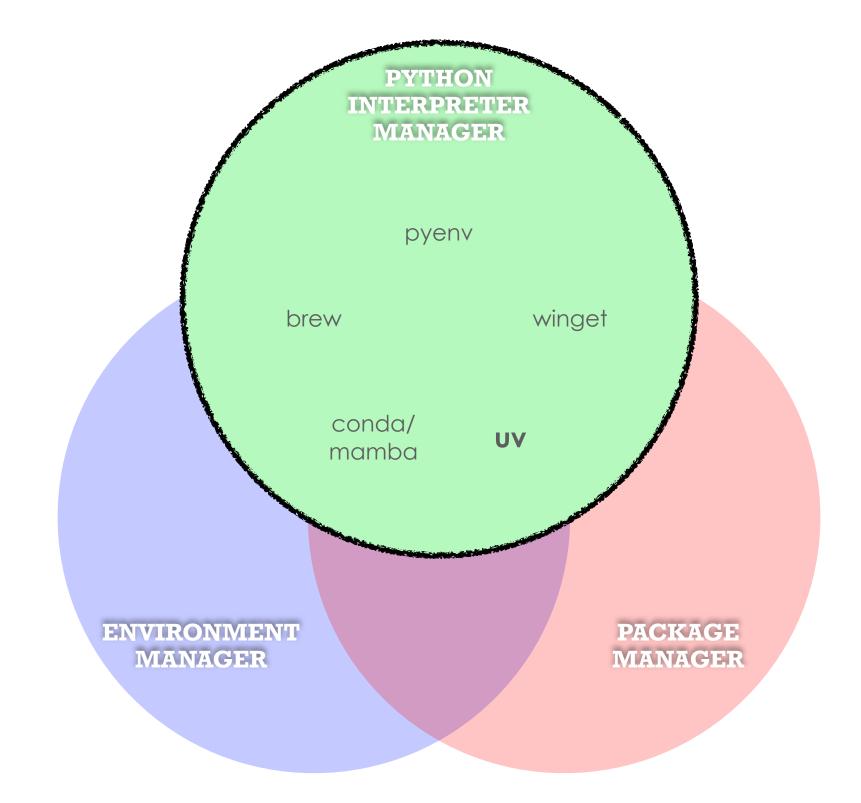
MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

#### Tl;dr: install uv



https://docs.astral.sh/uv/getting-started/installation/





## Installing uv



Instructions at <a href="https://docs.astral.sh/uv/getting-started/installation/">https://docs.astral.sh/uv/getting-started/installation/</a>

#### curl -LsSf https://astral.sh/uv/install.sh | sh

powershell -ExecutionPolicy ByPass -c "irm https://astral.sh/uv/install.ps1 | iex"

## Installing uv



Instructions at <a href="https://docs.astral.sh/uv/getting-started/installation/">https://docs.astral.sh/uv/getting-started/installation/</a>





## **Running Python with uv**

#### uv run [COMMAND]

Ensures that the command or script is run in a Python "environment" (more on that in a moment).

Will download Python for you if needed.

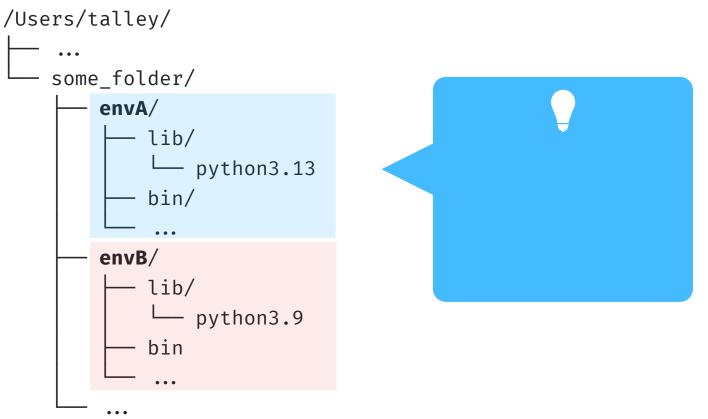


## What is a python virtual environment?

An isolated collection of **packages**, **settings**, and a **python interpreter**,

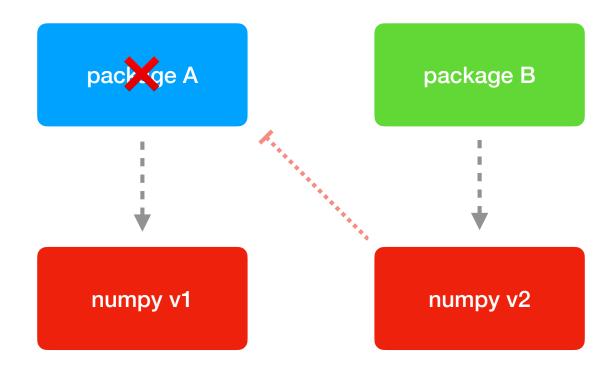
... that allows multiple different collections to exist on the same system

... (It's usually a literal folder somewhere on your computer...)



for more: https://realpython.com/python-virtual-environments-a-primer

Problem 1: conflicting package dependencies



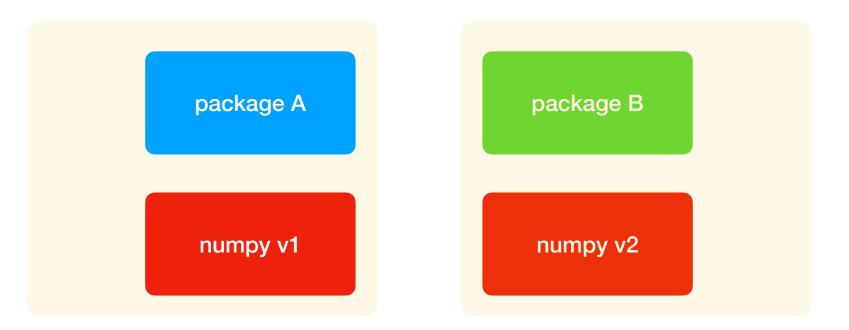


Tip! you can only have one version of any given package installed in an environment

#### Solution:

#### Create **multiple** "virtual" environments

each one contains all of the dependencies and requirements for a particular application



#### Problem 2: "It worked fine on my computer!" 🤥



Using environments also keeps your "globally" installed packages to a minimum, and makes your project dependencies much clearer



#### Try to avoid having one big "base" environment where you install everything!

#### Think of environments as disposable!

#### Don't get "attached" to an environment... Learn to recreate them quickly



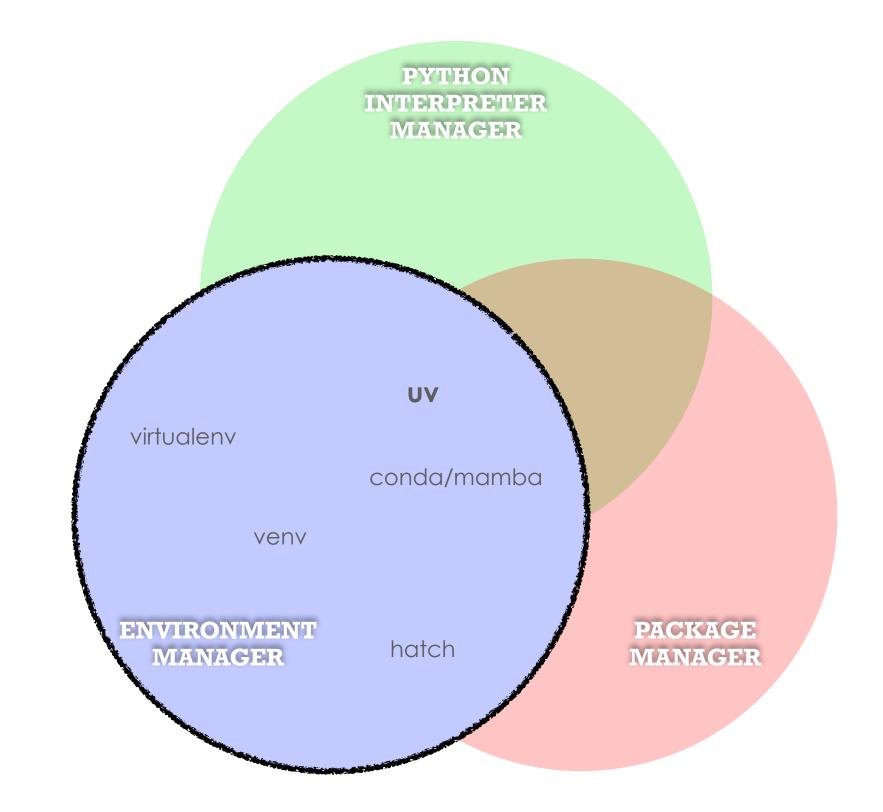
## How do I create a virtual environment?

You need... an environment manager

Examples:

- venv
- virtualenv
- conda/mamba
- pipenv
- poetry
- UV





## How do I create a virtual environment?

uv venv [ENV\_NAME] (defaults to ".venv")





### How do I create a virtual environment?

uv venv [ENV\_NAME]

(defaults to ".venv")

TIP: Specify Python version with --python/-p

uv venv -p 3.10

uv venv -p 3.13



## You need to "activate" it first...

### source .venv/bin/activate

### .venv\Scripts\activate

This sets up your terminal environment to know where to find things...



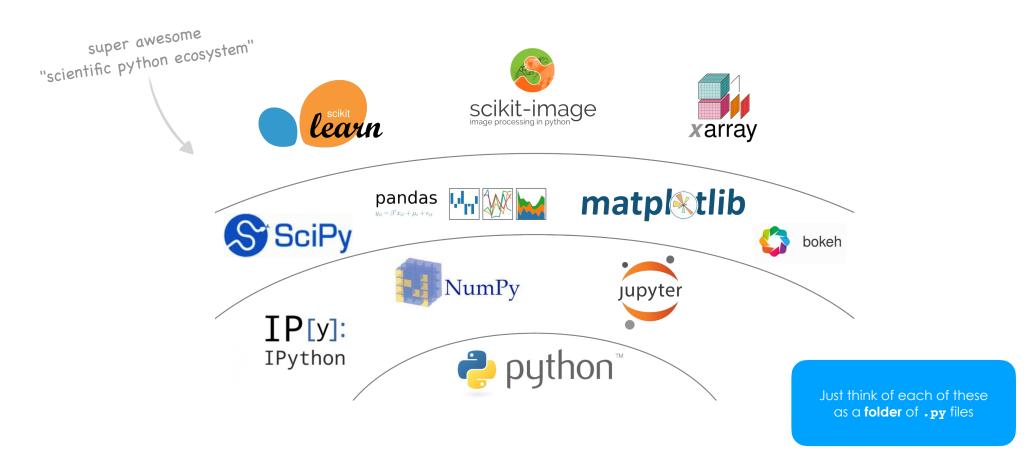
## Now that we've got an environment set up...

let's install some packages!



# "Packages" bring in additional functionality

Packages may sometimes be referred to as "libraries"



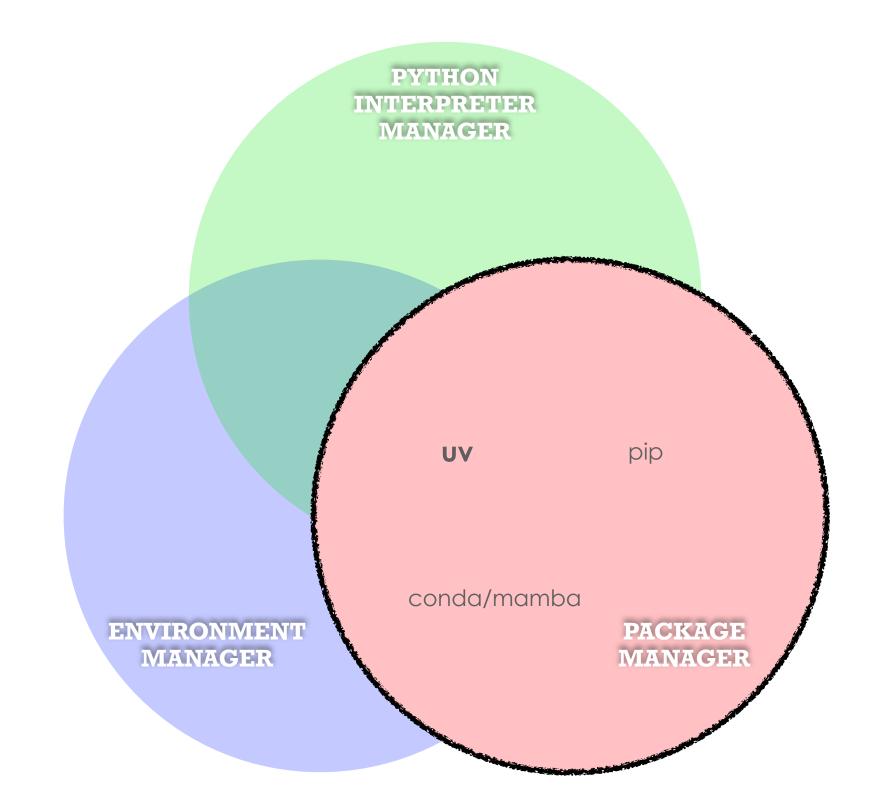
## How do I get Python packages?

You need... a package manager

Examples:

- pip
- conda/mamba
- uv pip





## How do I get Python packages?

### uv pip install <package>





## Tip: show installed packages with

### uv pip list

→ uv pip list	
Package	Version
cellpose	4.0.6
	1.17.1
fastremap	
filelock	3.18.0
fill-voids	2.1.0
fsspec	2025.5.1
imagecodecs	2025.3.30
jinja2	3.1.6
markupsafe	3.0.2
mpmath	1.3.0
natsort	8.4.0
networkx	3.5
numpy	2.3.1
opencv-python-headless	4.11.0.86
pillow	11.3.0
roifile	2025.5.10
scipy	1.16.0
segment-anything	1.0
setuptools	80.9.0
sympy	1.14.0
tiffile	2025.6.11
torch	2.7.1
torchvision	0.22.1
tqdm	4.67.1
typing-extensions	4.14.1

## Where did it install?? How does Python find code?

import os import sys import matplotlib import numpy as np import my module

From where??



How does Python find code?

```
sys
is a module
in the "standard library"
```

```
sys.path
```

path is a variable in the sys module

```
>>> import sys
>>> print(sys.path)
# on mac/linux
[
'.../.venv/lib/python3.X',
'.../.venv/lib/python3.X/lib-dynload',
'',
'.../.venv/lib/python3.X/site-packages',
]
```



## How does Python find code?

```
sys.path
```

```
>>> import sys
                                               >>> import sys
>>> print(sys.path)
                                               >>> print(sys.path)
                                               # on Windows:
# on mac/linux
 '.../.venv/lib/python3.X',
                                                 '...\.venv\Lib',
  .../.venv/lib/python3.X/lib-dynload',
                                                 '...\.venv\DLLs',
 1 1
                                                 1 1
  .../.venv/lib/python3.X/site-packages',
                                                 '...\.venv\Lib\site-packages',
]
                                                ]
```

For more:

https://docs.python.org/3/library/sys.html#sys.path

https://leemendelowitz.github.io/blog/how-does-python-find-packages.html

## How does Python find code?

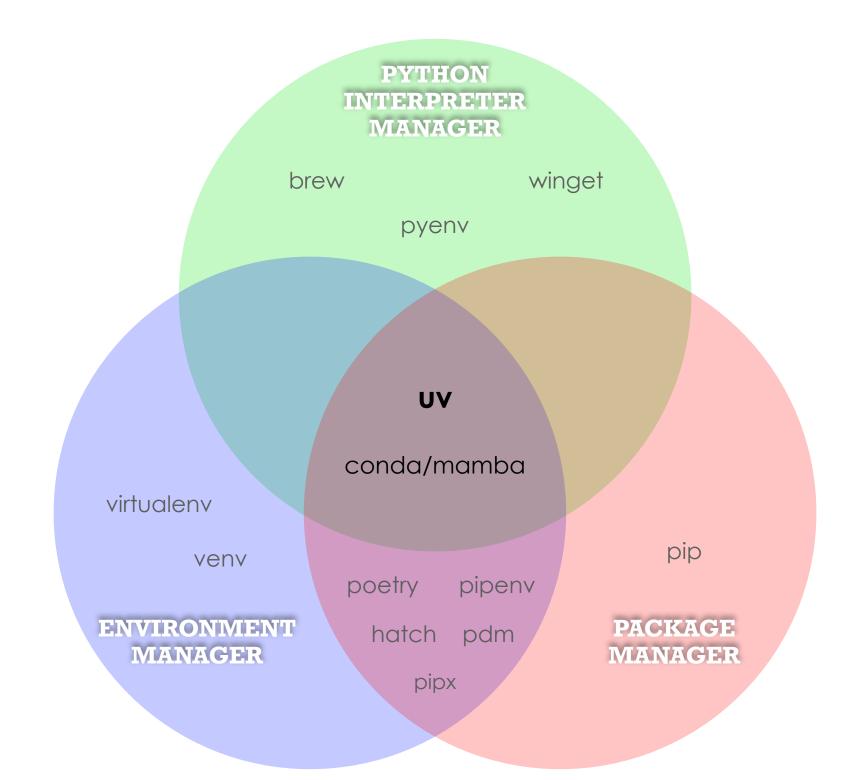


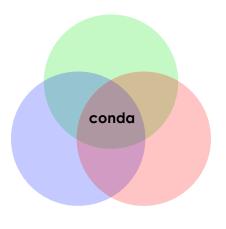
## Now you know...

#### **What Python** is

- Where it "lives" on your computer and where it finds code to run
- What **virtual environments** are, and how they help us isolate installed packages.
- What packages are, and how they bring in additional functionality.









https://github.com/conda/conda

### Conda is an open-source, cross-platform, language-agnostic package management system and environment management system

There are many "flavors" ... I recommend using **micromamba**:

https://mamba.readthedocs.io/en/latest/installation/micromamba-installation.html



For our purposes ... the most important difference between **uv** and **conda** will be where they download packages from

# uv pip vs. conda





https://pypi.org

https://anaconda.org/

pip/uv pip installs **Python** packages from **PyPI** into **any environment**  conda installs **any kind of** package from **Anaconda** into a **conda environment** 

see also: https://jakevdp.github.io/blog/2016/08/25/conda-myths-and-misconceptions/

## uv pip vs. conda

There may be cases where you must install a package via conda, e.g. for certain performance GPU computing tasks

... otherwise, stick with uv venv and uv pip

## (Re)creating environments with dependency files

### pip / uv-pip

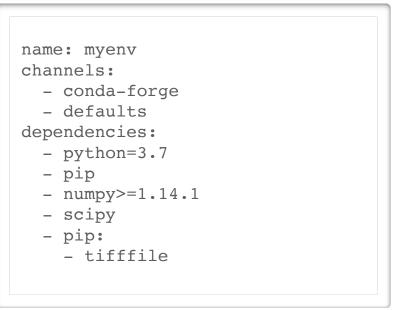
### requirements.txt

```
numpy>=1.14.1
scipy>=1.0.1
matplotlib>=2.0.0,!=3.0.0
networkx>=2.0
pillow>=4.3.0
imageio>=2.3.0
```

#### Create env and install

uv venv uv pip install -r requirements.txt conda

### environment.yml



#### Create env and install conda env create -f environment.yml

https://docs.conda.io/projects/conda/en/latest/user-guide/ tasks/manage-environments.html

## Listing dependencies within a script

my\_file.py

```
# /// script
# requires-python = ">=3.13"
# dependencies = [
# "numpy",
# ]
# ///
from numpy import add
print(add(1, 2))
```

This is a great (newer) convention in the Python world. It is supported by uv.

uv run my\_file.py

## One more cool trick...

### Run a command provided by any Python package

→ uvx cowsay -t 'hello world!'

hello world!

### uvx [COMMAND]

- Creates a Python environment
- Installs package with the same name as [COMMAND]
- Runs that [COMMAND]
- If the package and command don't have the same name:

uvx --from [PACKAGE] [COMMAND]



https://ipython.readthedocs.io

- Souped up interactive console (use ipython instead of python)
- Tab autocompletion

In [ <b>1</b> ]: <b>imp</b>	ort math					
In [2]: mat	n. <press tab=""></press>		<pre>factorial()</pre>			
	<pre>acosh() asin() asinh() atan() atan2()</pre>	<pre>cosh() degrees() dist() e erf()</pre>	<pre>floor() fmod() frexp() fsum() gamma()</pre>	<pre>isinf() isnan() isqrt() ldexp() lgamma()</pre>	nan perm() pi pow() prod()	sqrt() tan() tanh() tau trunc()
	atanh() ceil() comb() copysign()	erfc() exp() expm1() fabs()	gcd() hypot() inf isclose()	log() log10() log1p() log2()	<pre>radians() remainder() sin() sinh()</pre>	



https://ipython.readthedocs.io

- Souped up interactive console (use ipython instead of python)
- Tab autocompletion
- Easy help/documentation (? and ??)

```
In [5]: math.isclose?
Signature: math.isclose(a, b, *, rel_tol=1e-09, abs_tol=0.0)
Docstring:
Determine whether two floating point numbers are close in value.
  rel_tol
    maximum difference for being considered "close", relative to the
    magnitude of the input values
    abs_tol
    maximum difference for being considered "close", regardless of the
    magnitude of the input values
Return True if a is close in value to b, and False otherwise.
For the values to be considered close, the difference between them
must be smaller than at least one of the tolerances.
```

https://ipython.readthedocs.io

- Souped up interactive console (use ipython instead of python)
- Tab autocompletion
- Easy help/documentation (? and ??)
- "magics" (e.g. %run, %debug, ...) https://ipython.readthedocs.io/en/stable/interactive/magics.html

#### In [1]: %run some\_script.py

... then resume interactive usage with all of the variables from some\_script.py available



https://ipython.readthedocs.io

- Souped up interactive console (use ipython instead of python)
- Tab autocompletion
- Easy help/documentation (? and ??)
- "magics" (e.g. %run, %debug, ...) https://ipython.readthedocs.io/en/stable/interactive/magics.html
- Command history (press up/down to navigate previous commands)



# JupyterLab

https://jupyterlab.readthedocs.io/

File Edit View Run Kernel Tabs Settings Help ю С 🖻 + 🛠 🗊 🗂 🕨 🔳 C Markdown Python 3 O 0 Name In Depth: Linear Regression images Just as naive Bayes (discussed earlier in In Depth: Naive Bayes Classification) is a good starting point for classification tasks, linear regression models are a good starting point for Altair.ipvnb regression tasks. Such models are popular because they can be fit very quickly, and are very interpretable. You are probably familiar with the simplest form of a linear regression model & teaching. Cpp.ipynb (i.e., fitting a straight line to data) but such models can be extended to model more complicated data behavior. Data.ipynb In this section we will start with a quick intuitive walk-through of the mathematics behind this well-known problem, before seeing how before moving on to see how linear models can be Easta.ipynb generalized to account for more complicated patterns in data. Julia.ipynb We begin y File Edit View Run Kernel Tabs Settings Help 🗖 🗔 Linear Re 🗯 🗖 Lorenz.ipynb [2] Launcher Altair.ipynb Output View × smatplotl <h1><font 🍦 lorenz.py color="#f37626">pyt</font>hom import ma : 2012-2015 R.ipynb import : not<font Notebook color="#f37626">e</font>book</h1> import nu 🔘 🗅 untitled.dio 🗅 untitled1.dio Slide Type Simple 🗅 untitled2.dio ۲ C G G 🗅 untitled3.dio We will st 🗅 untitled4.dio Python 3 C++11 C++14 C++17 Raw NBConvert Forma untitled5.dio where a is 🗅 untitled6.dio **?** R Consider Advanced Tools **O** rng = np. x = 10 \* \* Cell Metadata Julia 1.1.0 phylogenetics (Python 3.7) Ð y = 2 \* x plt.scatt >\_ Console Notebook Metadata C C "kernelspec": { 350 400 450 "display\_name": "Python 3", "language": "python", "name": "python3" 🗏 Julia.ipynb Lorenz.ipynb R.ipynb . language\_info": { B + X □ □ ▶ ■ C Markdown ∨ Python 3 🖻 + 🛠 🗇 🗂 🕨 🔳 😋 Markdown 🗸 🖻 + 💥 🗋 🖻 🕨 🔳 C Markdown 🗸 Julia 1 'codemirror\_mode": {
 "name": "ipython", "version": 3 R Julia python notebook },
"file\_extension": ".py", "mimetype": "text/x-python", "name": "python", "nbconvert\_exporter": ... using RDatasets, Gadfly [3]: ggplot(data=iris, aes(x=Sepal,Len plot(dataset("datasets","iris"), x="Se smatplotlib inline "python". from ipywidgets import interactive, fixed "pygments\_lexer": 'ipython3" We explore the Lorenz system of differential We can us "version": "3.6.7" equations: 'toc-autonumbering": false, from sk Species antina vendular "toc-showcode": t  $\dot{x} = \sigma(y - x)$ 0 5 7 @ Python 3 | Idle -showmarkdowntxt": true  $\dot{v} = \rho x - v - xz$  $\dot{z} = -\beta z + xy$ Let's change  $(\sigma, \beta, \rho)$  with ipywidgets and examine the trajectories. [8]: eigen(x) [2]: from lorenz import solve\_lorenz [1]: head(iris) [8]: Eigen{Complex{Float64},Complex{Float 64},Array{Complex{Float64},2},Array{Complex{Float64},1} w = interactive(solve\_lorenz,sigma=(0.0,50. Sepal.Length Sepal.Width Petal.Length interactive(children=(FloatSlider(valu eigenvalues: 5.1 3.5 e=10.0, description='sigma', max=50.0), Flo atSlider(value=2.6666666666666... 10-element Array{Complex{Float64},1}: 4.793881566545466 + 0.0im 4.9 3.0 0.9445989635995898 + 0.0in 0 🛐 6 @ Python 3 | Idle Mode: Command 🛞 Ln 1, Col 1 Lorenz.ipynb

Jupyter Notebooks offer inline figures and integrated (markdown) documentation. ... right next to your interactive code.

1.4

1.4

It's a nice format for exploration, communication

## Running notebooks with uv via juv



## uvx juv init notebook.ipynb uvx juv run notebook.ipynb

- Ensures a Python environment with Jupyter installed
- Starts a Jupyter server
- Installs any dependencies listed inline at the top of the notebook

## Jupyter (notebook/lab) is great particularly for sharing code with figures & teaching ...

### but

being comfortable using Python/IPython without a notebook is critical



#### PyCharm https://www.jetbrains.com/pycharm/

self\_bars = ax.bar(x-WIDTH, ratio\_self, width=WIDTH, color='b', align='center')
others\_bars = ax.bar(x, ratio\_others, width=WIDTH, color='g', align='center')

B python survey [C:\Users\Ernst.Haagsman\PycharmProjects\python survey] - ...\main.py [python survey] - PyCharm

Eile Edit View Navigate Code Refactor Run Tools VCS Window Help

RATIO\_COUNT = ratio\_self.count()
x = np.arange(RATIO\_COUNT)
WIDTH = 0.4

ax.set\_xlabel('Ratios')
ax.set\_ylabel('Observations')

python\_survey ) 🔂 main.py

fig = plt.figure()

ax = fig.add\_subplot(111)

63

### **Integrated Development Environments ("IDE")**

### **VS** Code

<pre>&gt; ('Self', 'Most popular')) &gt; plt.show() </pre>	File Edit Selection View Go Debug Terminal	s tio	Python Interactive	- hello - Visual Studio C	code.visuals		
Python Console ax.set_ylabel('Observations') labels = [str(lb1) for lb1 in ratio_self.index ax.set_yticks( - 0,5 * WUTH) ax.legend((self_bars[0], others_bars[0]), ('Self', 'Most popular')) plt.show() >>>	<pre>     jupyter.py ×</pre>	··· ≣ Python Interact		Count (100,)	X Value array([ 0. , 0.202020	୯ ୭ ∎ ଅ 🛱	□□ ··· ) @ @ []
T <u>e</u> TODO V <u>e</u> Version Control <b>●</b> Python Control ■ Terminal <b>●</b> Docker excited out step_7 (a minute ago)	<pre>7 print(msg) 8 8 8 8 8 9 #%% Plot values 10 import numpy as np 11 import matplotlib.pyplot as plt</pre>	value row pascal n msg index fibonacci [1] msg = "Hell	int list list str int ndarrav	10 10 11 (200.)	7 [1, 9, 36, 84, 126, 1 [[1], [1, 1], [1, 2, 10 'Hello again' 9 arrav([0.00000000e+00	1], [1, 3, 3, 1]	
*	12 13 x = np.linspace(0, 20, 100) 14 plt.plot(x, np.sin(x)) 15 plt.show() 16 17 Run Cell Run Above Run Below 18 #%% Fibonacci calculation 19 n = 200 20 fibonacci[0] = 0 21 fibonacci[0] = 0 22 fibonacci[0] = 1 23 index = 2 hon 3.7.3 64-bit (Continuum: virtualenv)	Hello Worl [2]) msg = "Hel? Hello aga [3]) #%% Plot v [6] Shift-ente	C/Users/TestUser/Documents/Spyder File Edit Search Source Run Del D D D D D D D D D D D D D D D D D D D	sug Consoles Projects Tools View Help I I I I I I I I I I I I I I I I I I I	han 'scalar's lie fra . John Jones Markan Sang Bars Sang Lie Law Angel wet yee yer   wet yeen ye   second wet   second yeen yeen yeen yeen yeen yeen yeen ye	X Outre ●X	series Series (1,) Series object of pandas.core.series
Pyti			D defound     fonts     D images     D locale     D plugins     D tests	25 26 # %% Perform calculations 27 28 # Spline parameters		<ul> <li>print_File</li> <li>Exomple Stizer closs</li> <li>Source Stizer closs</li> <li>Source Stiener</li> <li>_init</li> </ul>	test_none NoneType 1 NoneType object of builtins modul Variable explorer Help File explorer Find in files Breakpoints Static code analysis Profiler Onlin Drython console

0 X

÷.

din.

👎 python\_survey 🖌 🕨 🗰 🌸 🛞 🧊 📰 🚽 🖓 😰 🖕 🔍

Most popular

🗶 🔝 🔲 🔍 🔍 🔍 🖉 🥒 4-bit color) 14.09K

SciView: Data Plots

1750 -

# Summary

- Get Python by installing uv: <a href="https://docs.astral.sh/uv/getting-started/installation/">https://docs.astral.sh/uv/getting-started/installation/</a>
- Create new environments (often) with: uv venv
- Consider environments "disposable" ... don't get attached :) Use requirements.txt (for pip) or environment.yml (for conda) files to rebuild them easily
- Use uv pip install to install packages. (For some difficult binary packages, you'll need conda)
- If you're experiencing mysterious errors... make a new environment!
- uv run <command> to quickly execute a script/command in an environment
- uvx <command> to run a command from any package in an isolated env.
- Use Jupyter for a browser based IDE/notebook/console with rich media, widgets, markdown, etc... But know how to use Python without it!

# Any questions?

# lingering confusions?

