# Elements Of Data Science - F2022

# Introduction to Data Science Tools

9/7/2022

# TODOs

- **Read** Preface of PDSH
- **Read** Ch 1 of PDSH
- **Skim** Ch 2 of PDSH: Introduction to NumPy

- Complete Weekly Quiz 01

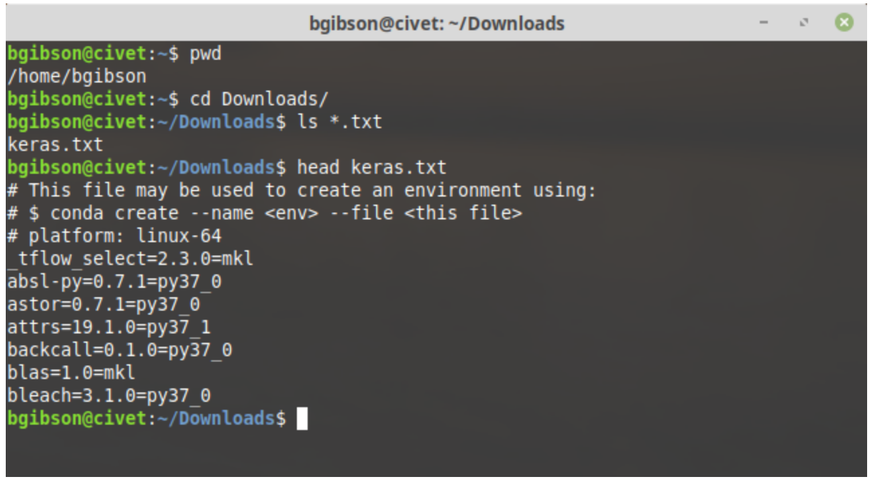
# TODAY

- Software tools we'll be using

# Our Python Data Science Stack

- Python (3.10): Programming language
- Anaconda : Package maintenance and environments
- Jupyter : IDE
- Git : Source control and versioning

# Aside: The Terminal and The Shell




- If not familiar, get aquainted
- Common set of commands (Ex. cd, ls, cat, mv)
- OSX and Linux: Terminal + bash/zsh (already installed)
- Windows: install Git Bash (or use WSL)

# Aside: Common Shell Commands

- **cd** : change directory
- **pwd** : where am i
- **ls** : list directory contents
- **head/tail** : print the beginning/end of a file
- **cat** : print entire file
- **less** : open a file in a pager
- **rm** : remove file
- **which** : path to executable
- ...

- Links to Tutorials

# Data Science Life Skills

- Data munging
- Visualization
- Statistical analysis
- Machine learning
- Reporting
- Prototyping
- Productionizing...

# Why Python?

- Robust and active DS stack
- Cross-platform
- Relatively low learning curve
- Fast to answers and prototypes

- Many other good languages and frameworks (R, Julia, etc.)

# Why Python?

- But isn't python slow?
- **Issues:**
    - GIL (Global Interpereter Lock)
    - dynamic typing
- **Solutions:**
    - numpy + vectorization
    - multiprocessing
    - pypy instead of CPython
    - distributed processing with pyspark?

- Article discussing issues and fixes: "Are your Python programs running slow?..."

# The Python DS Stack

- **Data munging** : pandas, numpy
- **Visualization** : matplotlib, seaborn, plotly
- **Statistical analysis** : scipy, statsmodels, patsy
- **Machine learning** : scikit-learn, tensorflow, pytorch
- **Reporting** : jupyter+ipython, dash
- **Prototyping** : flask
- **Productionizing...**

# Python 2 vs 3

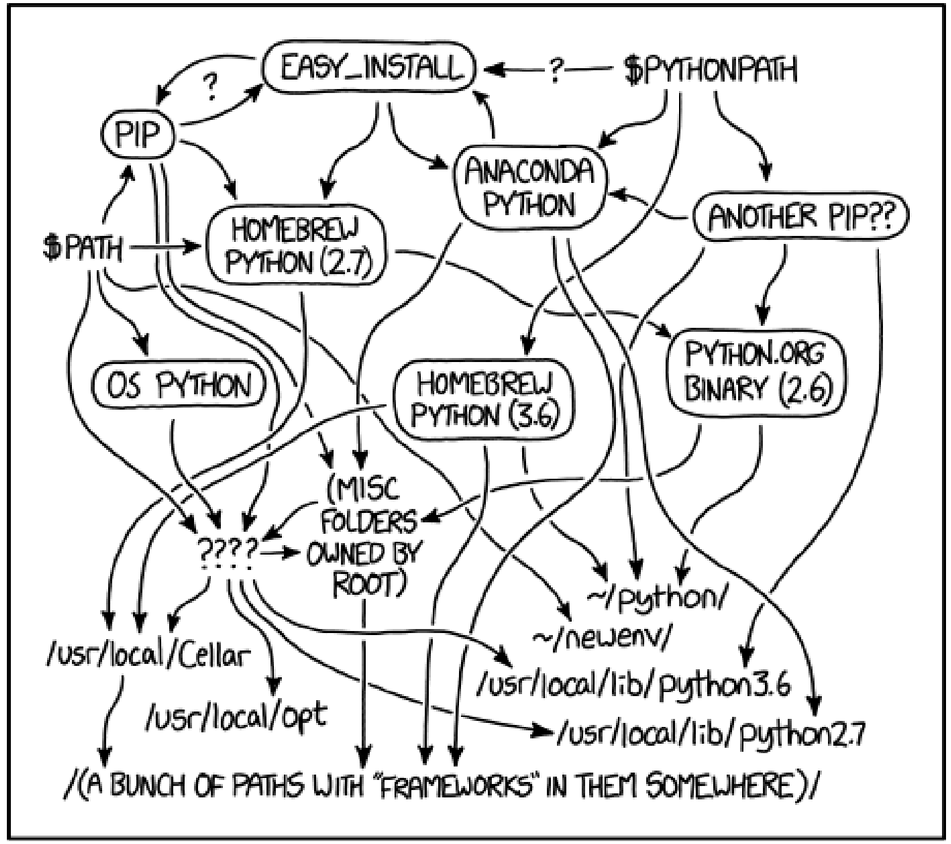
- We'll be using Python 3.10
- Python 2 end of life was Jan 1, 2020

- Need python 2 for another class? Virtual environments!

# How To Get Python

- You might already have it
- But your OS needs it!
- Our solution: Anaconda

# Why Anaconda?




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

https://imgs.xkcd.com/comics/python_environment.png

# Why Anaconda?

- includes most of what we need by default
- package curation
- dependency control
- conda virtual environments
- cross-platform

# Installing Anaconda

- Download via https://www.anaconda.com/products/individual
- Select OS and Grab Python 3.8 version
- Install somewhere easy to navigate to
    - /home/bgibson/anaconda3
    - C:\Users\brygib\anaconda3
- Recommend letting installer run `conda init` to set up your shell
- Note: base environment activated by default
    - To Turn off:

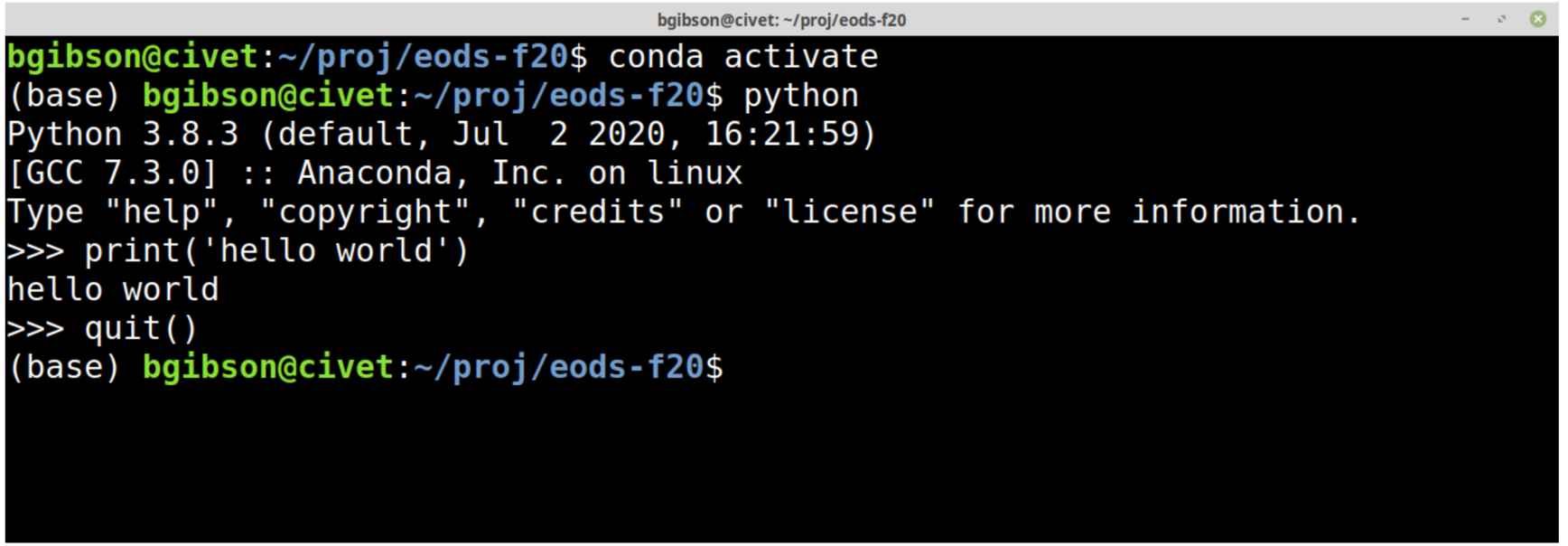
      ```
      conda config --set auto_activate_base false
      ```

# Running Python

- via terminal:
    - python REPL
    - python command line
    - python script
    - ipython REPL
- via jupyter
- via other IDE
- online via Google Colab
- ...

# Running Python

- Via REPL (Read–Eval–Print Loop)
  - `$ conda activate`
  - `(base)$ python`




- quit() or Ctrl-D to exit

# Running Python

Via command line

```
(base) bgibson@civet:~$ python -c "print('hello')"
hello
```

Via script

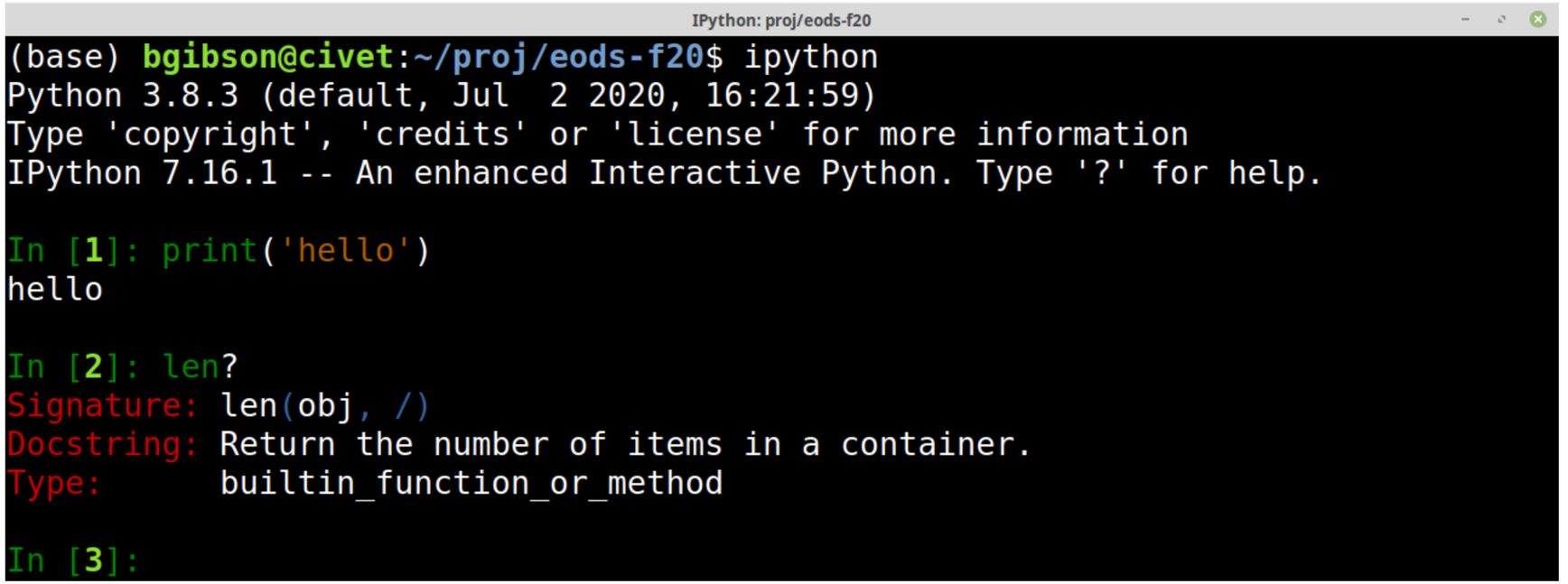
```
(base) bgibson@civet:~$ echo "print('hello')" > /tmp/say_hello.py
(base) bgibson@civet:~$ python /tmp/say_hello.py
hello
```

# Ipython: Interactive Python

- history (`python` does this now as well)
- tab completion (`python` does this now as well)
- "magic" commands
- help via `?` (`python` has `help()` as well)
- (see PDSH Ch 1 for more info)

# Ipython : REPL and Help

- `$conda activate` if (base) not activated



# Ipython Magic Commands

- preceded by % for line, %% for cell

# Ipython Magic Commands

- preceded by % for line, %% for cell

In [1]:
```
!echo 'print("hello from ipython")' > /tmp/say_hello.py
```

# Ipython Magic Commands

- preceded by % for line, %% for cell

In [1]:
```
!echo 'print("hello from ipython")' > /tmp/say_hello.py
```

In [2]:
```
%run /tmp/say_hello.py
```

```
hello from ipython
```

# Ipython Magic Commands

- preceded by % for line, %% for cell

In [1]:
```
!echo 'print("hello from ipython")' > /tmp/say_hello.py
```

In [2]:
```
%run /tmp/say_hello.py
```

```
hello from ipython
```

In [3]:
```
%timeit sorted([5,1,2,5])
```

```
129 ns ± 2.33 ns per loop (mean ± std. dev. of 7 runs, 10,000,000 loops each)
```

# Ipython Magic Commands

- preceded by % for line, %% for cell

In [1]:
```
!echo 'print("hello from ipython")' > /tmp/say_hello.py
```

In [2]:
```
%run /tmp/say_hello.py
```

```
hello from ipython
```

In [3]:
```
%timeit sorted([5,1,2,5])
```

```
129 ns ± 2.33 ns per loop (mean ± std. dev. of 7 runs, 10,000,000 loops each)
```

In [4]:
```
%%timeit
x = []
for i in range(20):
    x.append(i**2)
```

```
3.88 µs ± 33.9 ns per loop (mean ± std. dev. of 7 runs, 100,000 loops each)
```

# Help with Magic Commands

- get information about the %timeit magic function

```
%timeit?
```

- get info on all magic functions

```
%magic
```

- get list of magic functions

```
%lsmagic
```

# Ipython Notebooks with Jupyter

- Jupyter: application that combines code, markup and visualizations
- interact via web browser
- notebooks are easily sharable
- Jupyter can run other kernels as well: R, Julia, C#, etc.
- To launch via command line:

```
(base) bgibson@civet:~$ cd ~/proj
(base) bgibson@civet:~/proj$ jupyter notebook
```

- launches dashboard in your default browser
- Ctrl-C to kill server

# Other IDEs

- jupyterlab
- spyder
- pycharm
- visual studio code ...

# Arguments for Notebooks

- fast to iterate
- easy to test new ideas
- wide adoption

# Arguments against notebooks

- out of order execution
- messy code
- issues with version control
- slides by Joel Grus

# How to deal with version issues? Virtual Environments

- encapsulate python executable and packages
- allow for easy experimentation
- workaround versioning issues

- two major implementations: virtualenv and conda (we'll be using conda)

# Virtual Environments with Conda

Example for creating a new environment called py2 with python=2.7:

```
(base) bgibson@civet:~$ conda create -n py2 python=2.7
...

(base) bgibson@civet:~$ conda activate py2

(py2) bgibson@civet:~$ which python
/home/bgibson/anaconda3/envs/py2/bin/python

(py2) bgibson@civet:~$ python --version
Python 2.7.18 :: Anaconda, Inc.

(py2) bgibson@civet:~$ conda deactivate

(base) bgibson@civet:~$ which python
/home/bgibson/anaconda3/bin/python

(base) bgibson@civet:~$ python --version
Python 3.9.2
```

# Managing Conda Enviroments

# Installing New Packages

- Again, don't want to mess with system packages!

# Installing New Packages

- Again, don't want to mess with system packages!

1. first, try conda (with conda-forge):

```
conda install -n [env_name] -c conda-forge [package]
```

2. next, try another channel : eg. bioconda

```
conda install -n [env_name] -c bioconda [package]
```

3. then, try pip:

```
conda activate [env_name]
pip install [package]
```

# Installing New Packages

- Again, don't want to mess with system packages!

1. first, try conda (with conda-forge):

```
conda install -n [env_name] -c conda-forge [package]
```

2. next, try another channel : eg. bioconda

```
conda install -n [env_name] -c bioconda [package]
```

3. then, try pip:

```
conda activate [env_name]
pip install [package]
```

- when you can, double check the path to your env

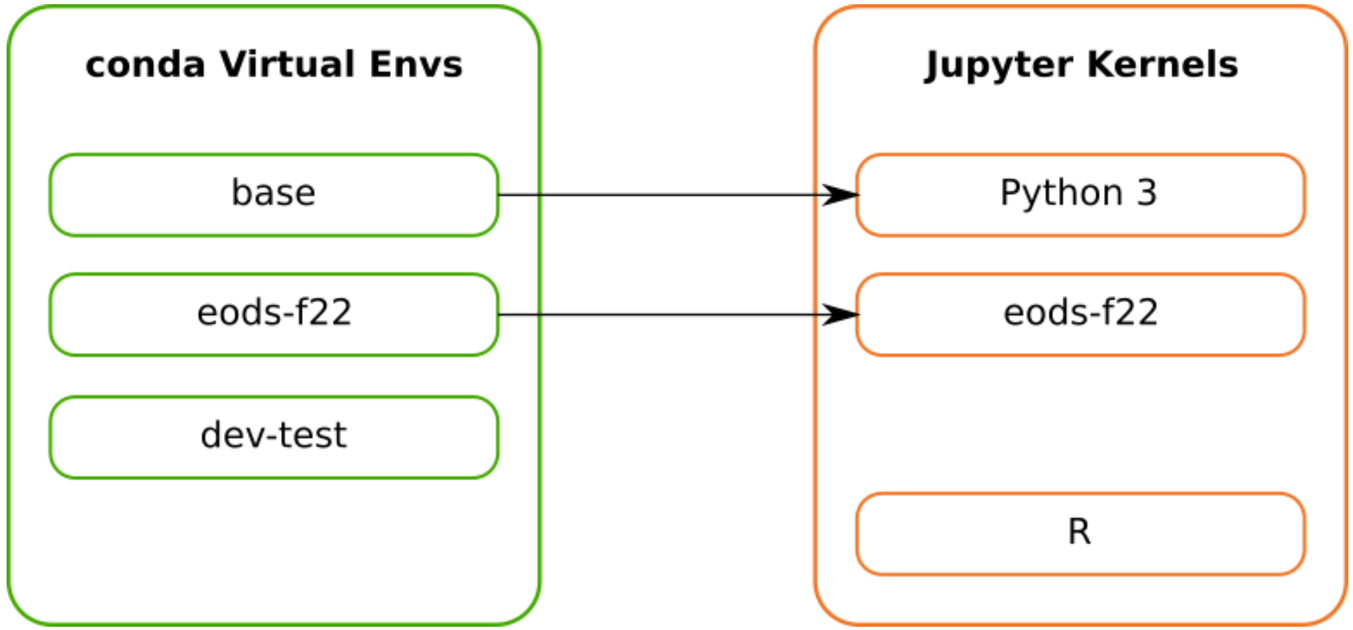
# Conda Channels: default vs conda-forge

- channels: locations where packages are stored
    - default: Anaconda terms specify only used in non-commercial application
    - conda-forge: where all of the good stuff is anyway

```
conda install -n [env_name] -c conda-forge [package]
```

# Conda Virtual Envs and Jupyter Kernels

- jupyter can run many different kernels
- conda envs not automatically added as available kernels



# Controlling Jupyter Kernels

- to install a new kernel in jupyter:

```
(base) $ conda activate py2
(py2) $ conda install -c conda-forge ipykernel
(py2) $ python -m ipykernel install --user --name py2
```

- to list kernels: `jupyter kernelspec list`

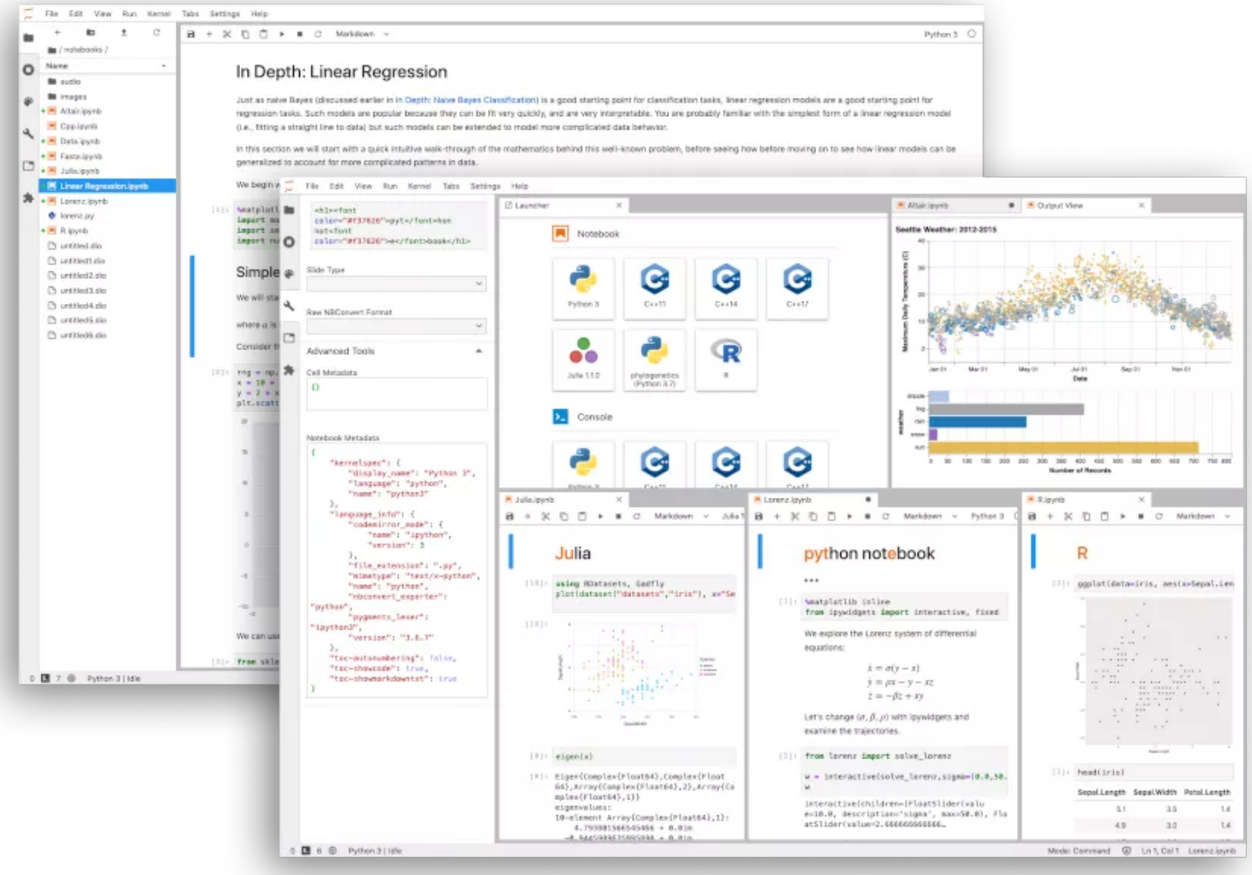
- to remove kernel: `jupyter kernelspec uninstall [name]`

# Jupyter Demo

- Important: h for help
- Markdown syntax help: https://daringfireball.net/projects/markdown/syntax

# Jupyter Classic vs JupyterLab

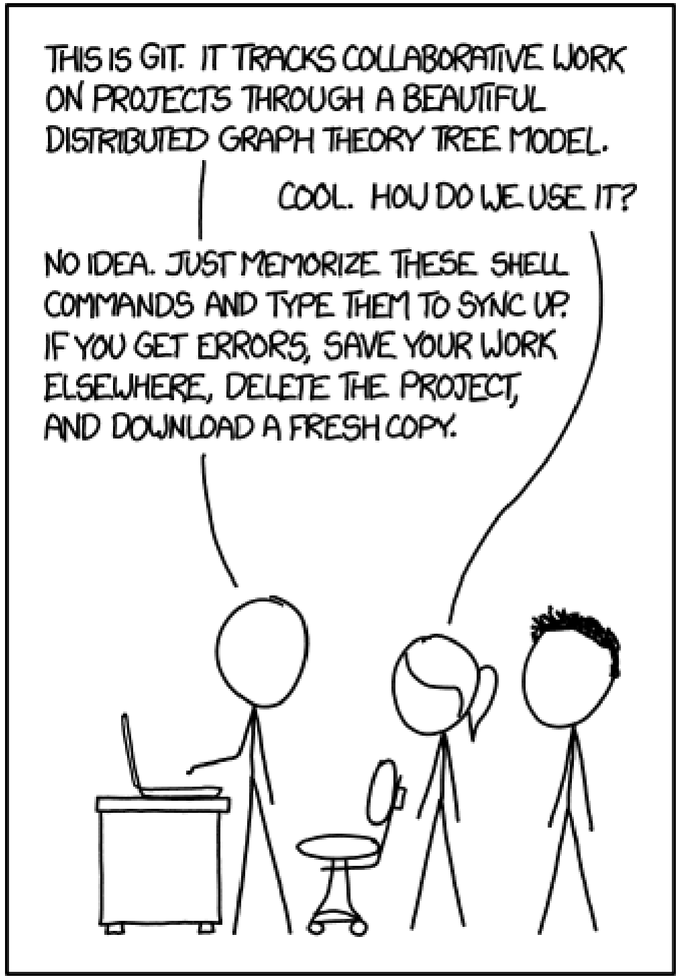
- start as either `jupyter notebook` or `jupyter lab`
- or replace `http://localhost:8888/tree` with `http://localhost:8888/lab`



# Example Notebooks

Gallery of interesting Jupyter Notebooks

# Git and Github




http://imgs.xkcd.com/comics/git.png

# Git

- distributed version control
- for code, documentation, *small* data
- can but used locally or with remote collaborators

# Github

- backup
- sharing
- used for both large and small projects
    - Ex: https://github.com/scikit-learn/scikit-learn

# Getting course material

- Can view online at: https://github.com/bryanrgibson/eods-f22
- You'll also want to clone locally:

```
$ cd [your projects folder]
$ git clone https://github.com/bryanrgibson/eods-f22
```

# Demo Week 1 Quiz

https://github.com/bryanrgibson/eods-f22/weekly_quiz/Week_01_

# Questions?

- Next time: Python review, numpy and pandas