Elements Of Data Science - F2023

Introduction to Data Science Tools

07/11/2023

TODOs

- **Read** Preface of PDSH
- **Read** Ch 1 of PDSH
- **Skim** Ch 2 of PDSH: Introduction to NumPy
- Weekly Quiz 01

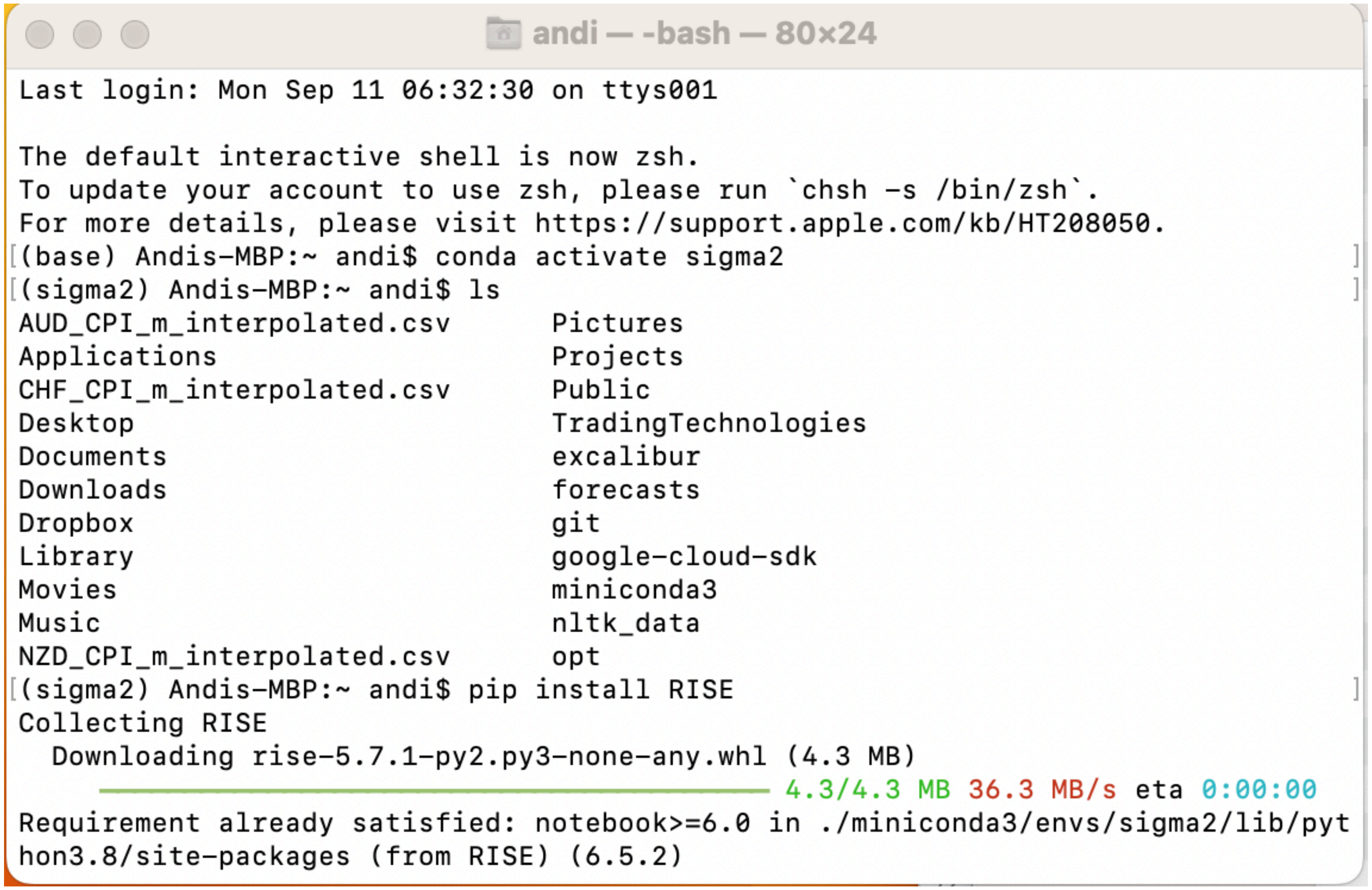
TODAY

- 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



```
Last login: Mon Sep 11 06:32:30 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.
(base) Andis-MBP:~ andi$ conda activate sigma2
(sigma2) Andis-MBP:~ andi$ ls
AUD_CPI_m_interpolated.csv      Pictures
Applications                    Projects
CHF_CPI_m_interpolated.csv      Public
Desktop                         TradingTechnologies
Documents                       excalibur
Downloads                       forecasts
Dropbox                         git
Library                         google-cloud-sdk
Movies                          miniconda3
Music                           nltk_data
NZD_CPI_m_interpolated.csv      opt
(sigma2) Andis-MBP:~ andi$ pip install RISE
Collecting RISE
  Downloading rise-5.7.1-py2.py3-none-any.whl (4.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.3/4.3 MB 36.3 MB/s eta 0:00:00
Requirement already satisfied: notebook>=6.0 in ./miniconda3/envs/sigma2/lib/pyt
hon3.8/site-packages (from RISE) (6.5.2)
```

- 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
- ...
- Link to Tutorial

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:**
    - dynamic typing
        - The Python interpreter does type checking only as code runs, and the type of a variable is allowed to change over its lifetime.
- **Solutions:**
    - numpy + vectorization
    - multiprocessing
    - pypy instead of CPython
    - distributed processing with pyspark?

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...**

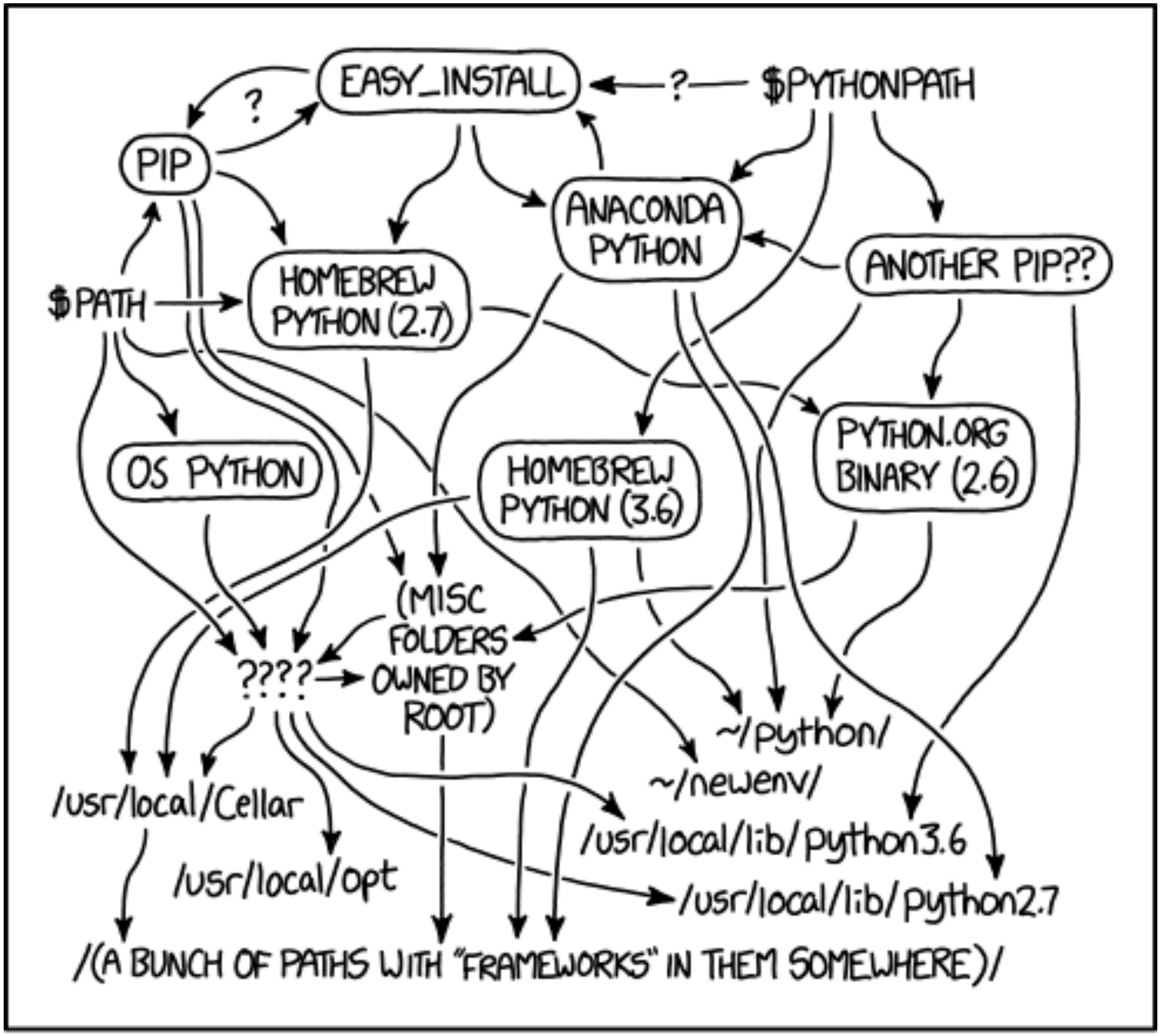
Python 2 vs 3

- 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.

Why Anaconda?

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

Installing Anaconda

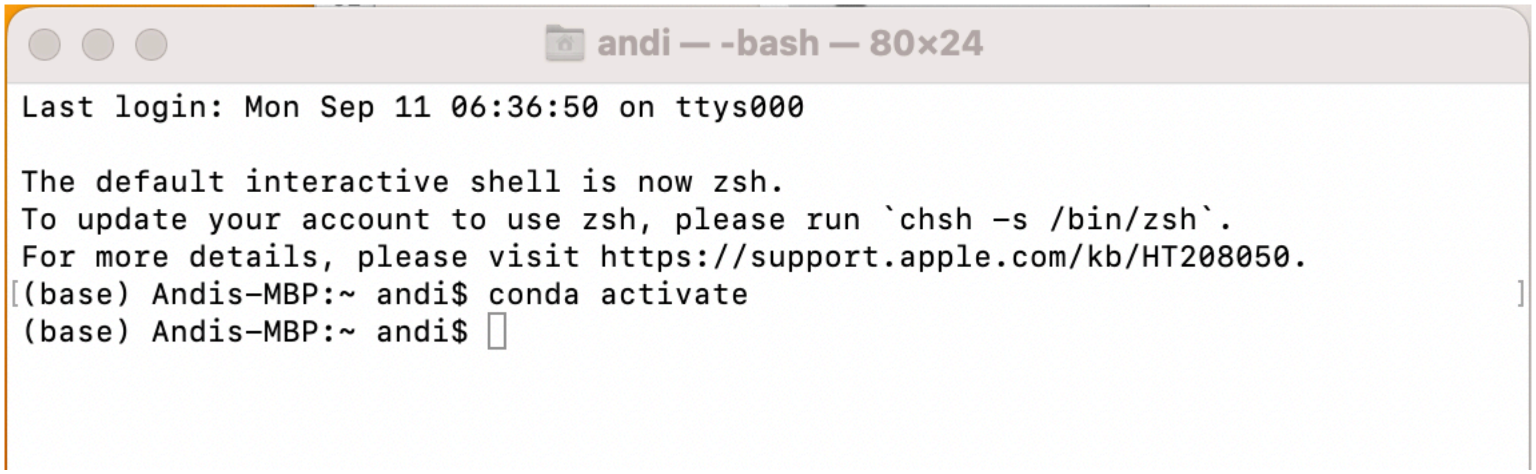
- Download via https://www.anaconda.com/products/individual
- Select OS and Grab Python 3.9 version
- Install somewhere easy to navigate to
    - /home/bgibson/anaconda3
    - C:\Users\brygib\anaconda3
- Recommend letting installer run `conda init` to set up your shell
- Note: base environment activated 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`



andi — -bash — 80×24

```
Last login: Mon Sep 11 06:36:50 on ttys000

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.
(base) Andis-MBP:~ andi$ conda activate
(base) Andis-MBP:~ andi$ 
```

- quit() or Ctrl-D to exit

Running Python

Via command line

```
(base) Andis-MBP:~ andi$ python -c "print('hello')"
hello
```

Via script

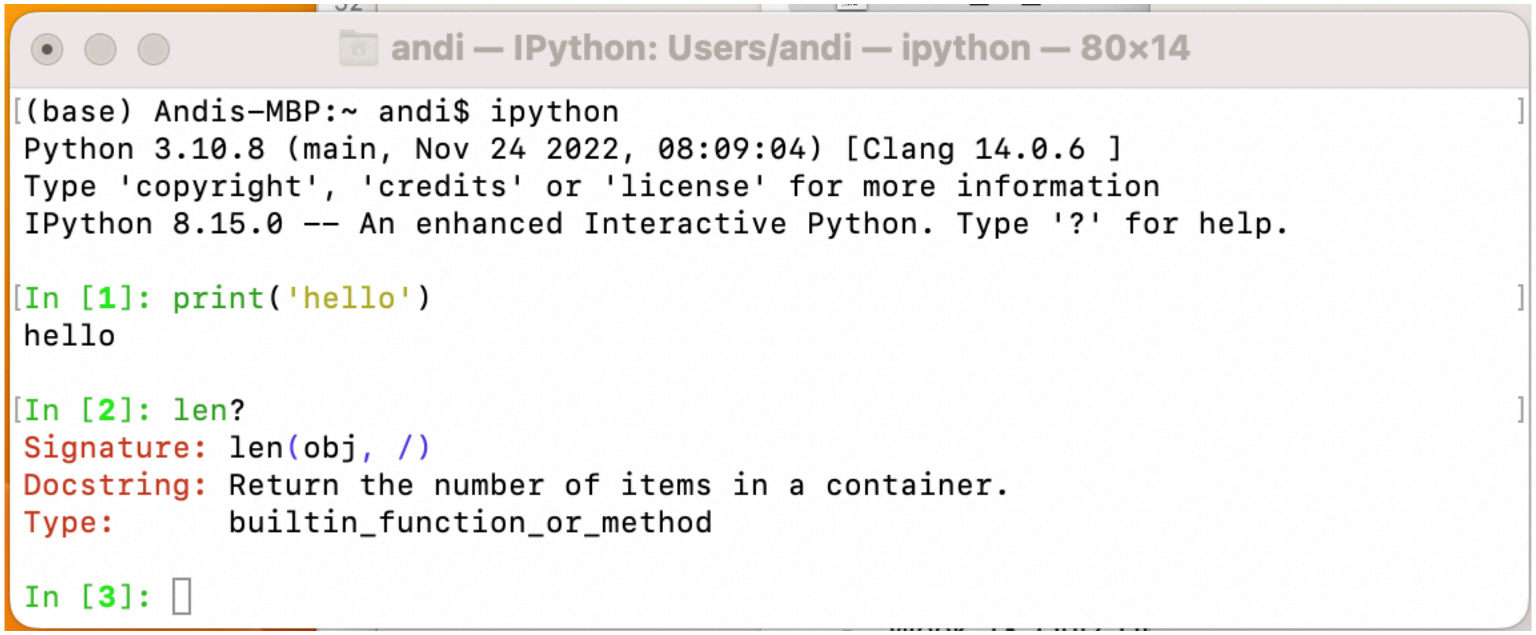
```
(base) Andis-MBP:~$ echo "print('hello')" > /tmp/say_hello.py
(base) Andis-MBP:~$ python /tmp/say_hello.py
hello
```

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



```
[(base) Andis-MBP:~ andi$ ipython
Python 3.10.8 (main, Nov 24 2022, 08:09:04) [Clang 14.0.6 ]
Type 'copyright', 'credits' or 'license' for more information
IPython 8.15.0 -- An enhanced Interactive Python. Type '?' for help.

[In [1]: print('hello')
hello

[In [2]: len?
Signature: len(obj, /)
Docstring: Return the number of items in a container.
Type:      builtin_function_or_method

In [3]: 
```

Ipython Magic Commands

- preceded by % for line, %% for cell

Ipython Magic Commands

- preceded by % for line, %% for cell

In [1]:
```
# The output of the `echo` can be redirected to a file instead of displ
# terminal
```

Ipython Magic Commands

- preceded by % for line, %% for cell

```
In [1]: # The output of the `echo` can be redirected to a file instead of displ
        # terminal
```

```
In [2]: !echo 'print("hello from Room 833")' > /tmp/say_hello.py
        !python /tmp/say_hello.py
```

```
hello from Room 833
```

Ipython Magic Commands

- preceded by % for line, %% for cell

```
In [1]: # The output of the `echo` can be redirected to a file instead of displ
        # terminal
```

```
In [2]: !echo 'print("hello from Room 833")' > /tmp/say_hello.py
        !python /tmp/say_hello.py
```

```
hello from Room 833
```

```
In [3]: %run /tmp/say_hello.py
```

```
hello from Room 833
```

Ipython Magic Commands

- preceded by % for line, %% for cell

In [1]:
```
# The output of the `echo` can be redirected to a file instead of displ
# terminal
```

In [2]:
```
!echo 'print("hello from Room 833")' > /tmp/say_hello.py
!python /tmp/say_hello.py
```

```
hello from Room 833
```

In [3]:
```
%run /tmp/say_hello.py
```

```
hello from Room 833
```

In [4]:
```
%timeit sorted([5,1,2,5])
```

```
166 ns ± 7.09 ns per loop (mean ± std. dev. of 7 runs, 10,000,00
0 loops each)
```

Ipython Magic Commands

- preceded by % for line, %% for cell

```
In [1]:  # The output of the `echo` can be redirected to a file instead of displ
         # terminal
```

```
In [2]:  !echo 'print("hello from Room 833")' > /tmp/say_hello.py
         !python /tmp/say_hello.py
```

```
hello from Room 833
```

```
In [3]:  %run /tmp/say_hello.py
```

```
hello from Room 833
```

```
In [4]:  %timeit sorted([5,1,2,5])
```

```
166 ns ± 7.09 ns per loop (mean ± std. dev. of 7 runs, 10,000,00
0 loops each)
```

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

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) Andis-MBP:~ andi$ cd ~/proj
(base) Andis-MBP:~ andi~/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) Andis-MBP:~ andi$ conda create -n py2 python=2.7
...
```

```
(base) Andis-MBP:~ andi$ conda activate py2
```

```
(py2) Andis-MBP:~ andi$ which python
/Users/andi/miniconda3/envs/py2/bin/python
```

```
(py2) Andis-MBP:~ andi$ python --version
Python 2.7.18 :: Anaconda, Inc.
```

```
(py2) Andis-MBP:~ andi$ conda deactivate
```

```
(base) Andis-MBP:~ andi$ which python
/Users/andi/miniconda3/bin/python
```

```
(base) Andis-MBP:~ andi$ python --version
Python 3.10.8
```

Managing Conda Enviroments

- `conda create -n [env_name]`
- `conda create -n [env_name] [package] [package]=[version]`
- **`conda env create --file [requirementsfile].yml`**
- `conda activate [name]`
- `conda deactivate`
- `conda env list`
- For more information see: https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html

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
```

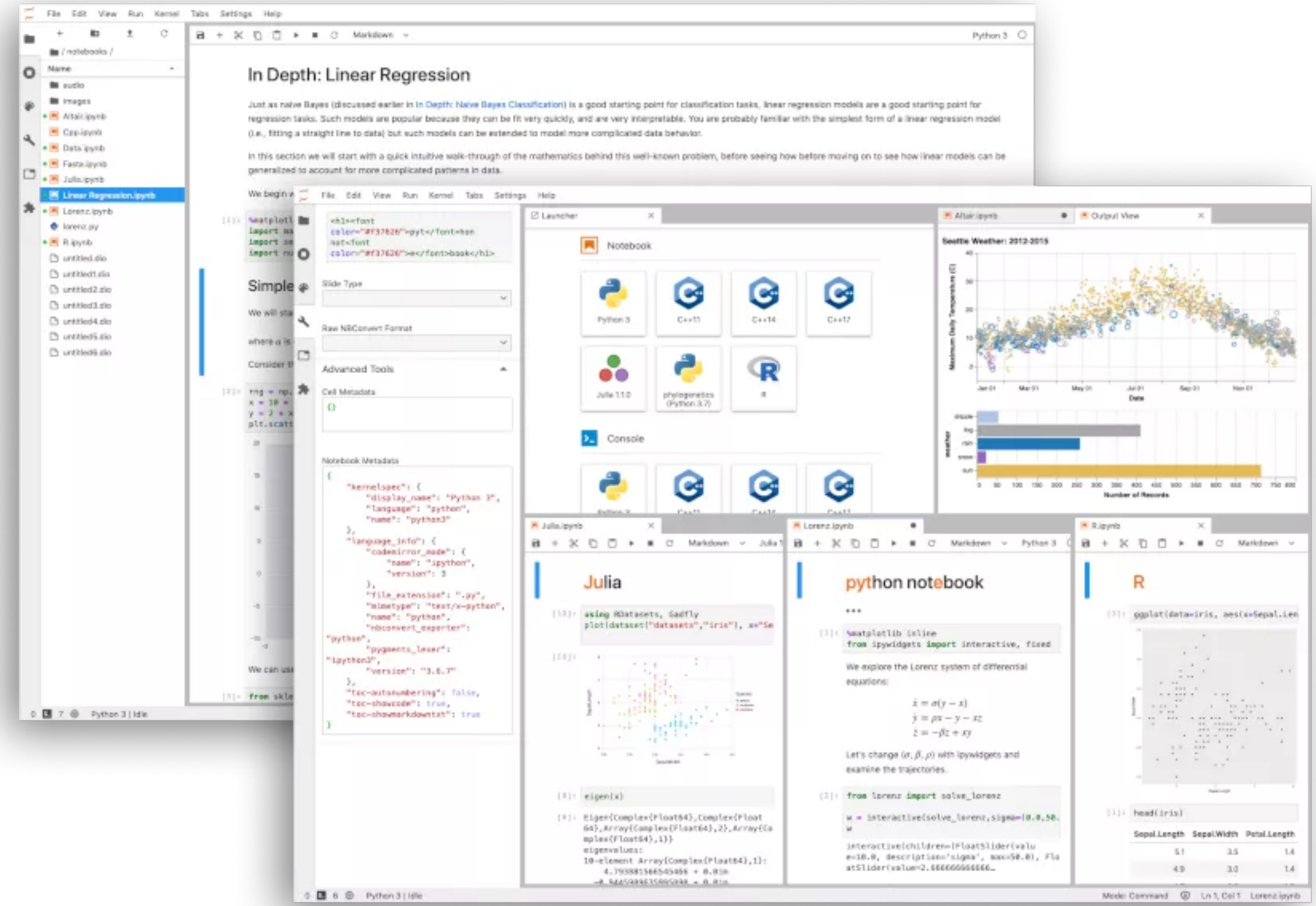
- to list kernels: `jupyter kernelspec list`
- to remove kernel: `jupyter kernelspec uninstall [name]`

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`



In Depth: Linear Regression

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 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 (i.e., fitting a straight line to data) but such models can be extended to model more complicated data behavior.

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 generalized to account for more complicated patterns in data.

```
<h1><font color="#f37626">pyt</font>hon
not<font color="#f37626">e</font>book</h1>
```

Notebook Metadata

```
{
    "kernelspec": {
        "display_name": "Python 3",
        "language": "python",
        "name": "python3"
    },
    "language_info": {
        "codemirror_mode": {
            "name": "ipython",
            "version": 3
        },
        "file_extension": ".py",
        "mimetype": "text/x-python",
        "name": "python",
        "nbconvert_exporter": "python",
        "pygments_lexer": "ipython3",
        "version": "3.6.7"
    },
    "toc-autonumbering": false,
    "toc-showcode": true,
    "toc-showmarkdowntxt": true
}
```

Julia

```
using RDatasets, Gadfly
plot(dataset("datasets","iris"), x="Se
```

```
eigen(x)
```

```
Eigen{Complex{Float64},Complex{Float64},Array{Complex{Float64},2},Array{Complex{Float64},1}}
eigenvalues:
10-element Array{Complex{Float64},1}:
    4.793981566545466 + 0.0im
```

python notebook

```
%matplotlib inline
from ipywidgets import interactive, fixed
```

We explore the Lorenz system of differential equations:

$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = \rho x - y - xz$$
$$\dot{z} = -\beta z + xy$$

Let's change (σ, β, ρ) with ipywidgets and examine the trajectories.

```
from lorenz import solve_lorenz

w = interactive(solve_lorenz,sigma=(0.0,50.
w
```

```
interactive(children=(FloatSlider(value=10.0, description='sigma', max=50.0), FloatSlider(value=2.66666666666…
```

R

```
ggplot(data=iris, aes(x=Sepal.Len
```

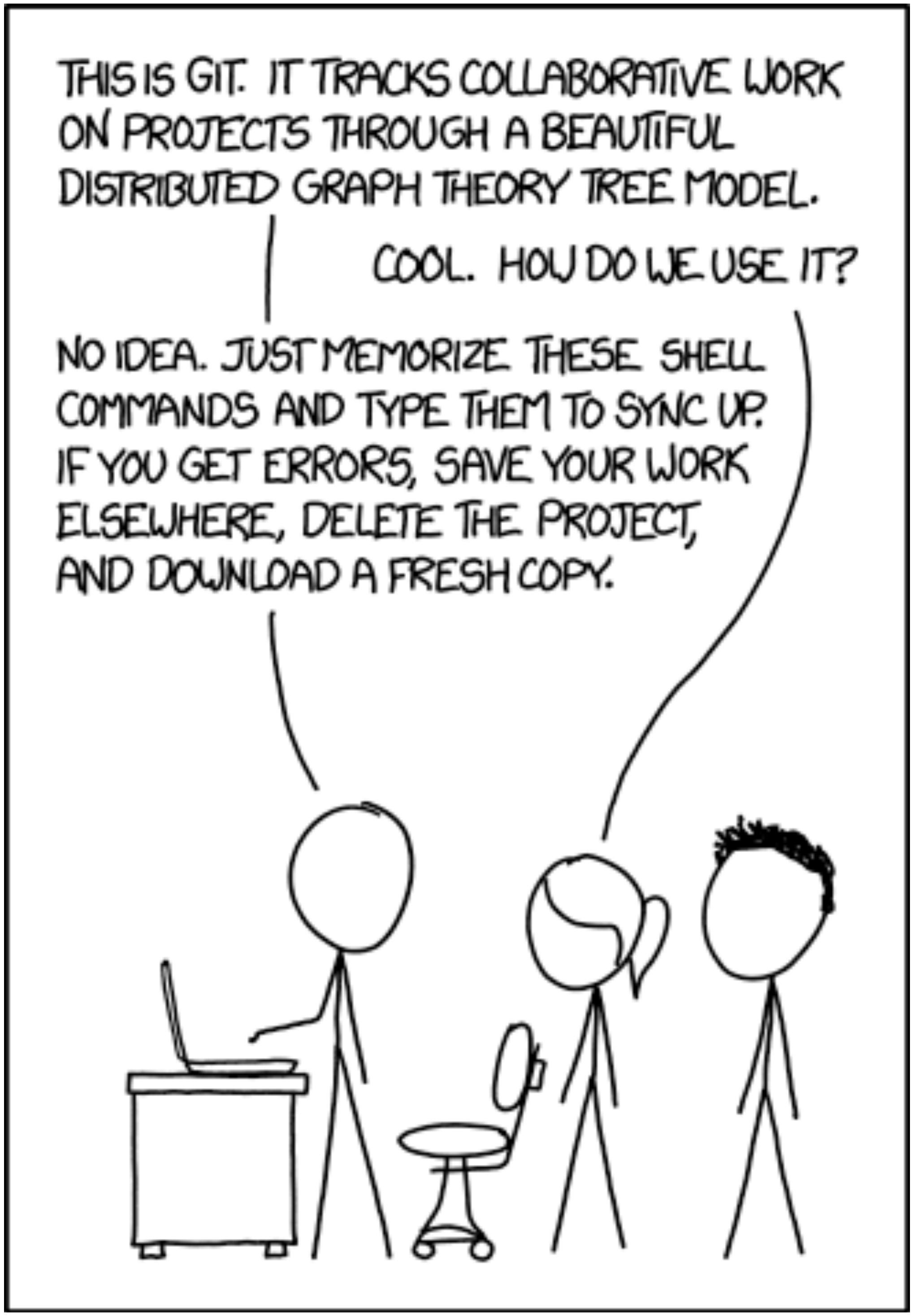
```
head(iris)
```

Sepal.Length	Sepal.Width	Petal.Length
5.1	3.5	1.4
4.9	3.0	1.4

Example Notebooks

Gallery of interesting Jupyter Notebooks

Git and Github


THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

Git

- distributed version control
- for code, documentation, *small* data
- can be 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: TBA
- You'll also want to clone locally:

```
$ cd [your projects folder]
$ git clone **TBA**
```

Demo Week 1 Quiz

Questions?

- Next time: Python review, numpy and pandas