A Flock of Birds...

A Heard of Cows...

A Huge Mess of Pythons...

A quick look at why nothing ever works on your machine

A Huge Mess of Pythons...

Pycharm

IDE

Python2

System Python

Notebook

Python3.4

numpy
v1.18.4

Python3.8

pip

Anaconda

Jupyter kernel

numpy
v1.22.4

conda

Spyder

matplotlib

A Huge Mess of Pythons...

Pycharm

IDE

Python2

System Python

Notebook

Python3.4

numpy
v1.18.4

Python3.8

pip

Anaconda

Jupyter kernel

numpy
v1.22.4

conda

Spyder

matplotlib

Python Installations

A Huge Mess of Pythons...

Pycharm IDE

System Python

Python2

Notebook

Python3.4

numpy
v1.18.4

Python3.8

pip

Anaconda

Jupyter kernel

numpy
v1.22.4

conda

matplotlib

Spyder

Python Installations package managers

A Huge Mess of Pythons...

Pycharm

IDE

System Python

Python2

Notebook

Python3.4

numpy
v1.18.4

Python3.8

pip

Anaconda

Jupyter kernel

numpy
v1.22.4

conda

Spyder

matplotlib

Python Installations

package managers

packages

A Huge Mess of Pythons...

Pycharm

IDE

Python2

System Python

Notebook

Python3.4

numpy
v1.18.4

Python3.8

pip

Anaconda

Jupyter kernel

numpy
v1.22.4

conda

Spyder

matplotlib

Python Installations

package managers

packages

editors

A Huge Mess of Pythons...

Python Installations	package managers	packages	editors
System Python	conda	numpy v1.18.4	IDE
Python2	pip	numpy v1.22.4	Pycharm
Python3.4		matplotlib	Spyder
Python3.8			Notebook
Anaconda			Jupyter kernel

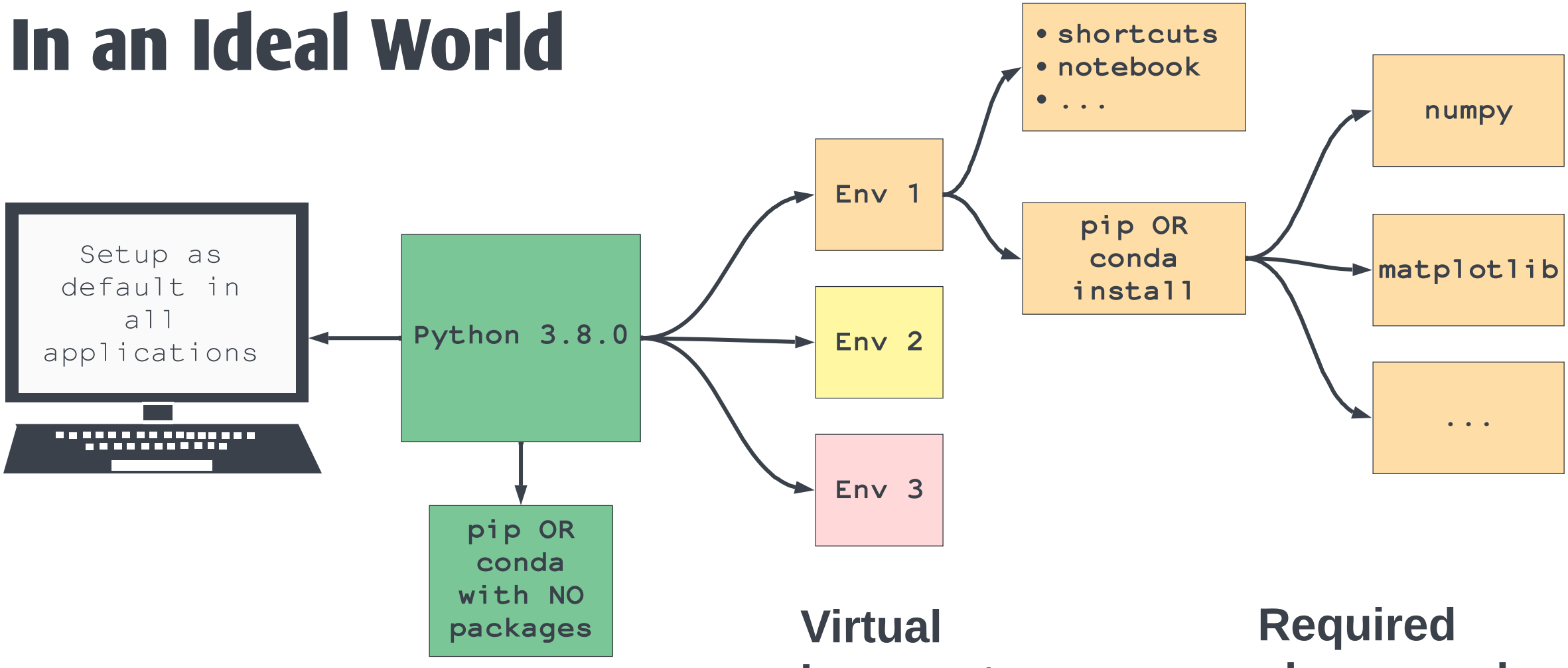
A Huge Mess of Pythons...

- Python is not like R or Matlab; your computer may actually be running Python to do it's own thing sometimes (e.g. MacOS System Python)
- You may have (inadvertently) installed **multiple versions of Python** and Python related tools.
- Having a mixed bunch or Pythons makes for hard to solve, **highly individual problems**

A Huge Mess of Pythons...

- Python is not like R or Matlab; your computer may actually be running Python to do it's own thing sometimes (e.g. MacOS System Python)
- You may have (inadvertently) installed **multiple versions of Python** and Python related tools.
- Having a mixed bunch or Pythons makes for hard to solve, **highly individual problems**
- The only way to fix it is to wipe your entire hard drive, **burn your computer, and start from scratch**
- well, maybe not...

In an Ideal World



One version of Python

Virtual environments for individual projects

Required packages and setup for each environment

Python installations

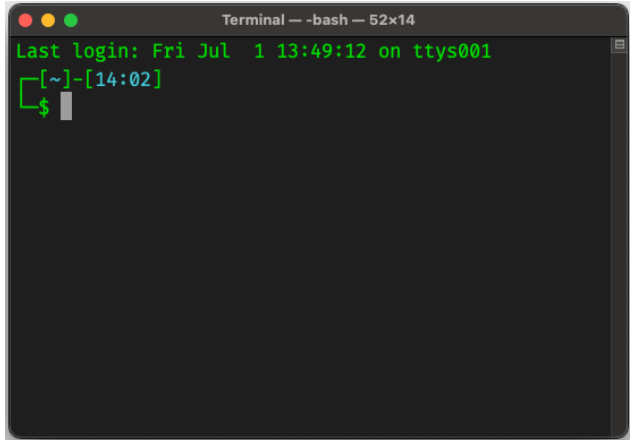
- Ideally you have **one Python**, that you have chosen
- Installing tools will sometimes install its own version of Python
- It's also fine to have multiple Pythons, if you know and keep track
- **Anaconda** is a Software bundle that makes the install (slightly) easier
  - It includes **Python**, a bunch of preinstalled packages, and their own package manager **conda**

ANACONDA.

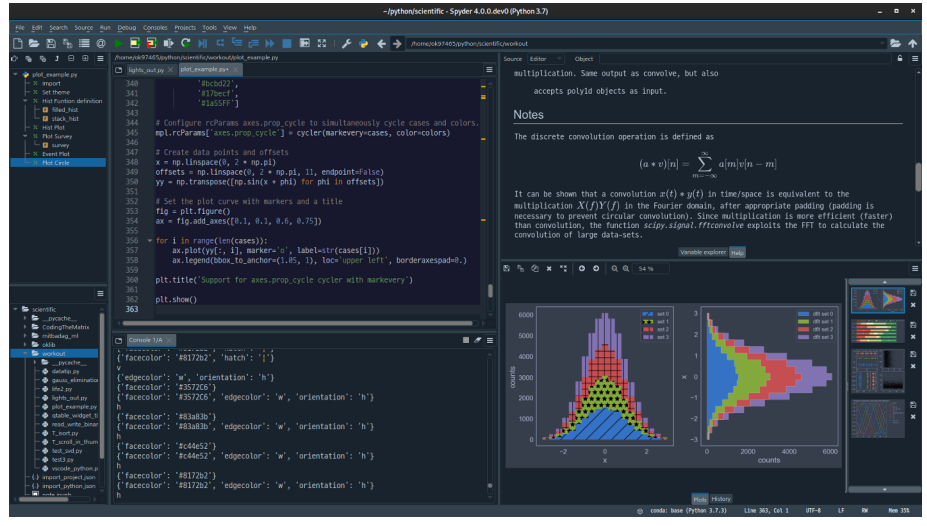
Package Managers

- Typically **pip** and **conda**
- **Don't mix and match using pip and conda.** It will work. But each package manager doesn't know what the other has done and you may end up with a huge mess.
  - **Except:** sometimes you have to mix and match! If things are not available in conda you have to use pip. But then only use pip for individual problematic cases.

But how do I WORK with it?



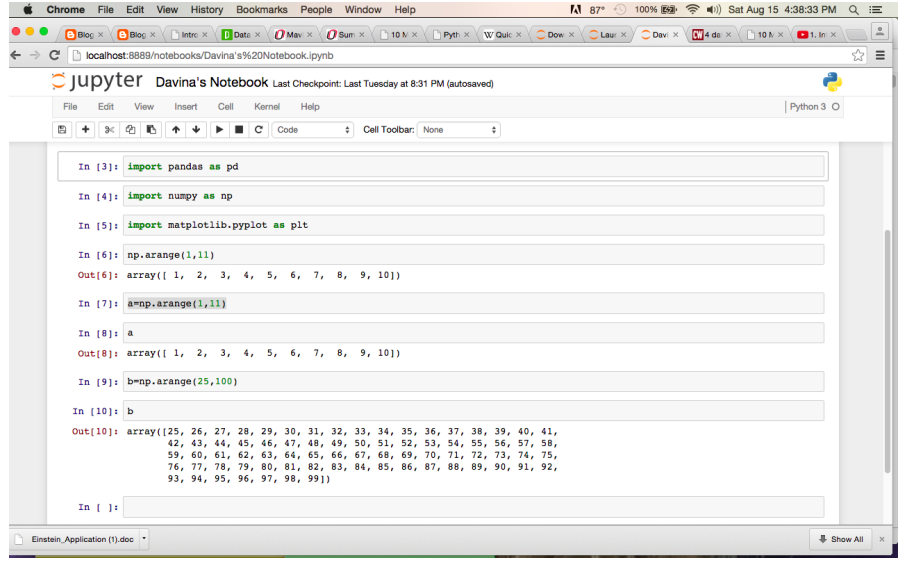
Back to the roots: Terminal



IDE

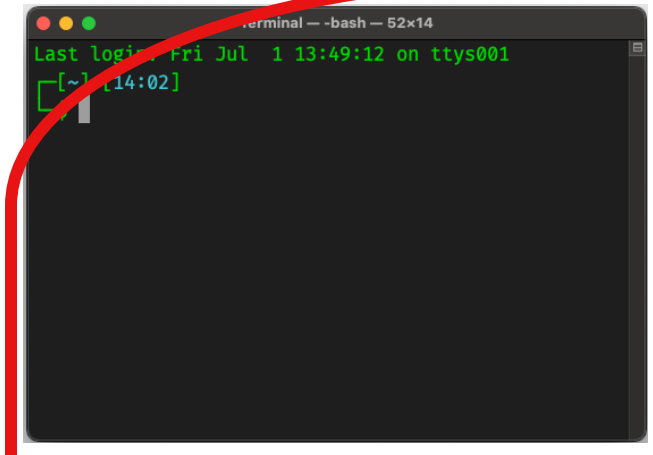
```
"""
SAW
"""
import numpy as np
# from numba import jit
# import matplotlib.pyplot as plt
# import time

np.seterr(under='warn', over='warn', divide="warn")


class saw_model():
    def __init__(self, parameters, seed=None):
        if seed is None:
            self.seed = np.random.randint(10000)
        else:
            self.seed = seed
        self.RS = np.random.RandomState(self.seed)

        self._L = None
        self.initialize = 2000  # number of steps to take for initializ

        # parameters
        self._ri = None
        self._rj = None
        self.eta = None  # transition probs
        self.delay = None
        self.TSIZE = None  # width of ellipse
        self._VMAX = None
        self._vrad = None
```

Basic code editor



Notebook

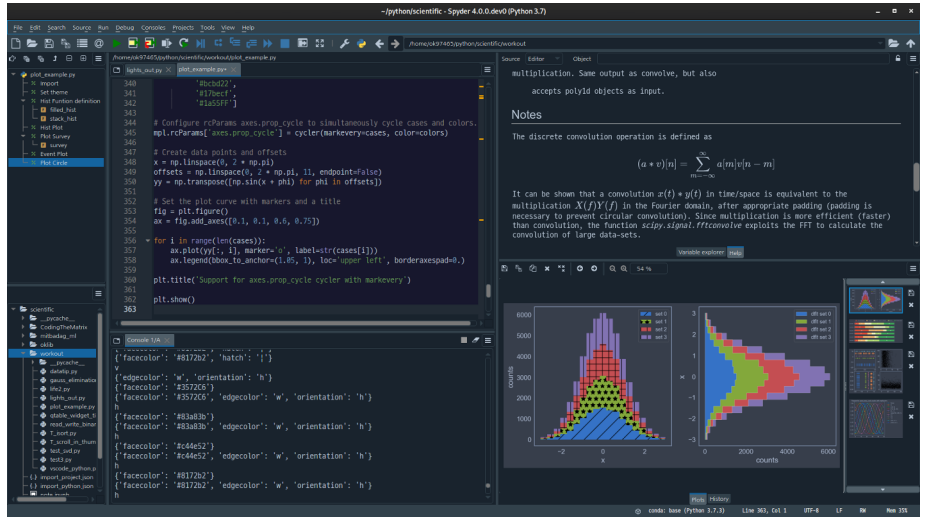
But how do I WORK with it?



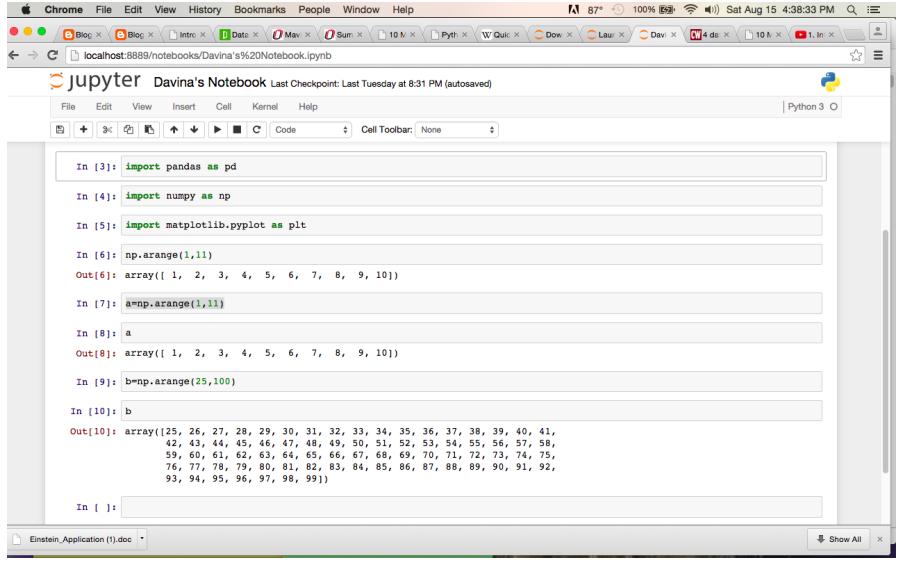
Back to the roots: Terminal

```
"""
SAW
"""
import numpy as np
# from numba import jit
# import matplotlib.pyplot as plt
# import time

np.seterr(under='warn', over='warn', divide="warn")


class saw_model():
    def __init__(self, parameters, seed=None):
        if seed is None:
            self.seed = np.random.randint(10000)
        else:
            self.seed = seed
        self.RS = np.random.RandomState(self.seed)

        self._L = None
        self.initialize = 2000  # number of steps to take for initializ

        # parameters
        self._ri = None
        self._rj = None
        self.eta = None  # transition probs
        self.delay = None
        self.TSIZE = None  # width of ellipse
        self._vr = None
        self._vrad = None
```

Basic code editor

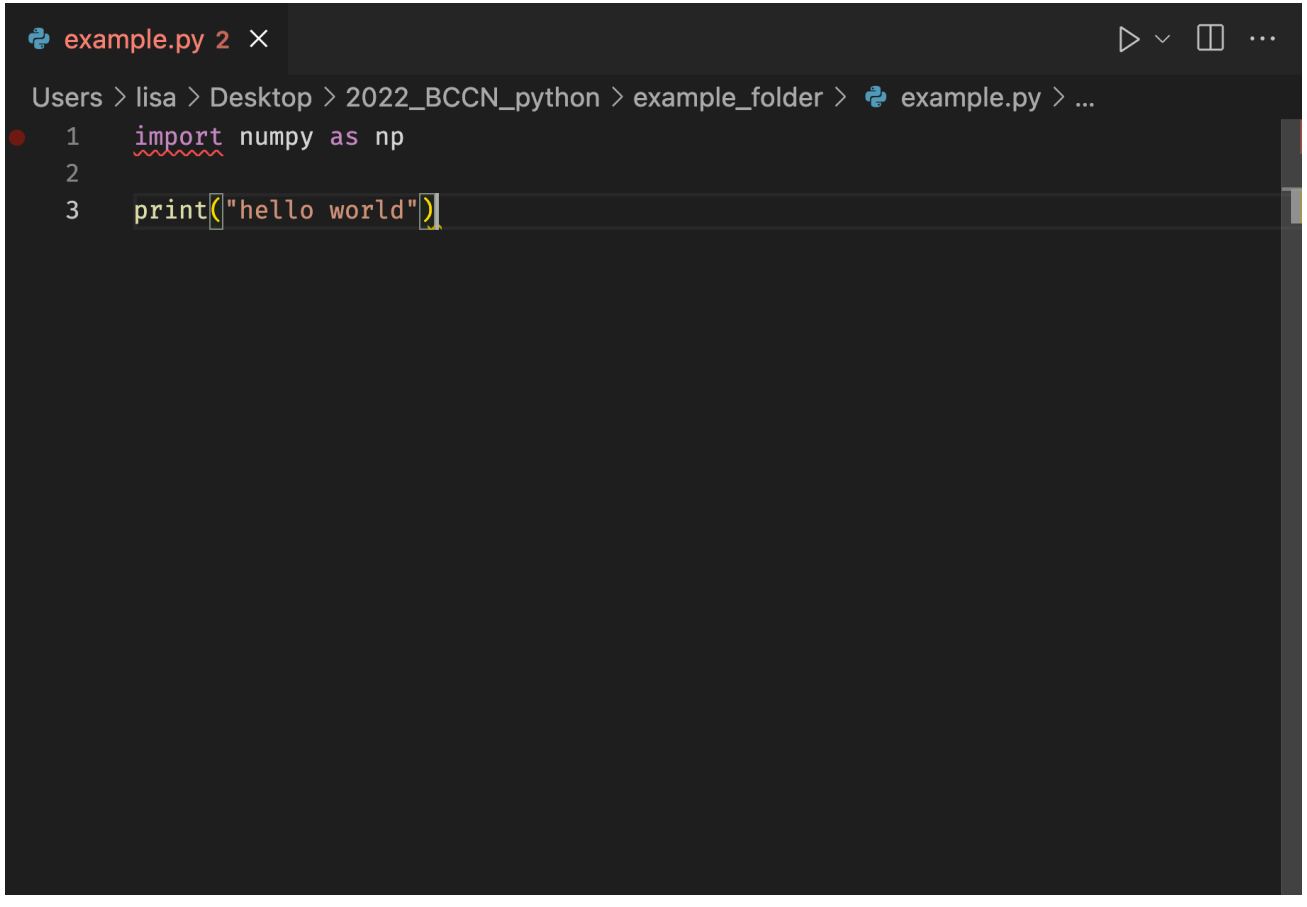


IDE

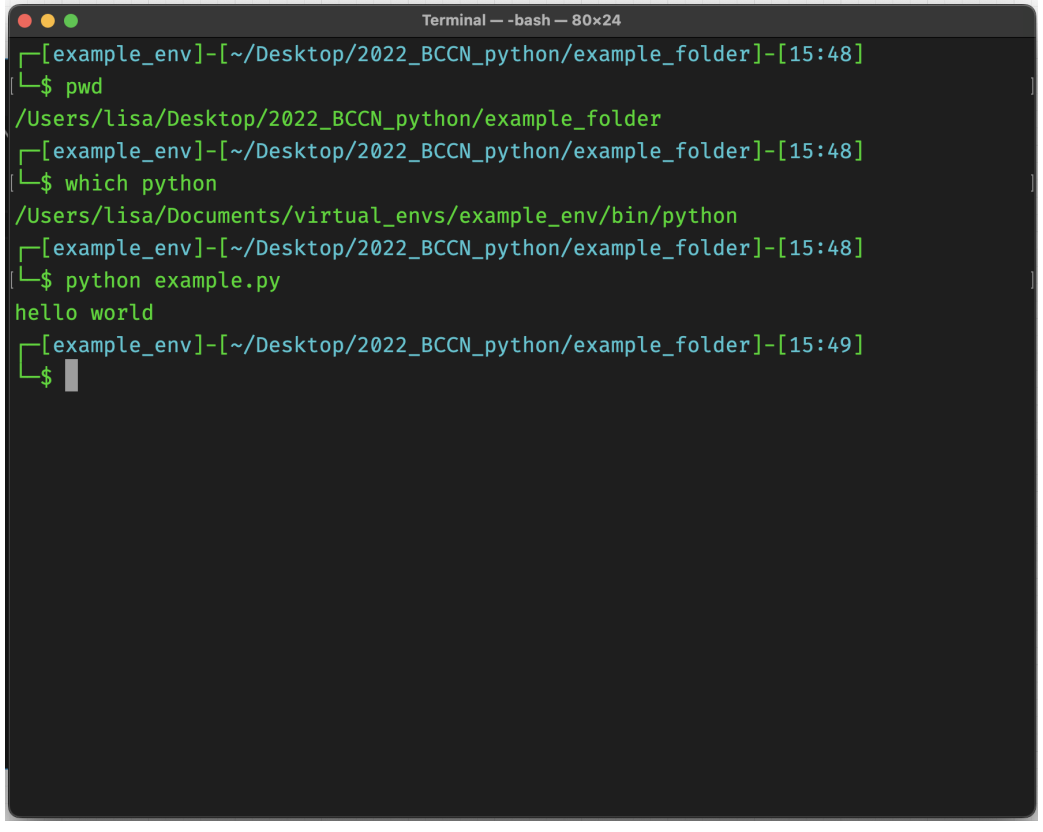


Notebook

Terminal



Editor



Terminal

Terminal

- Let's say you're running python from the Terminal
- You can find out what program is called when you type "python"

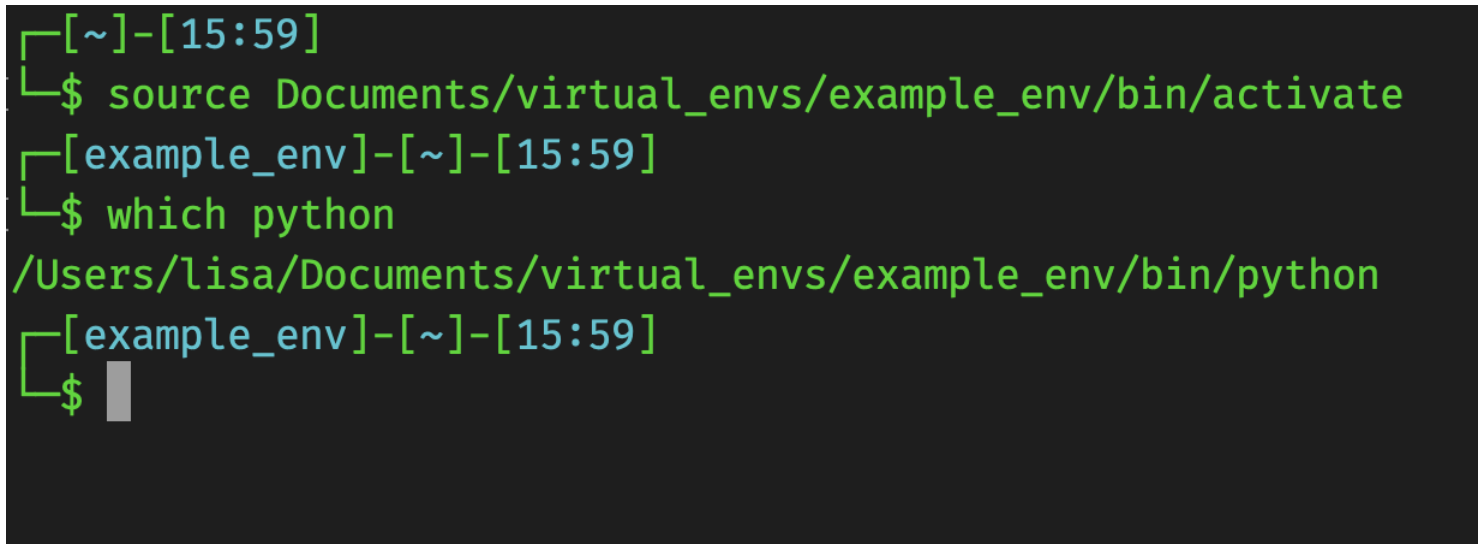
```
Lisas-MBP-4:~ root# which python
/usr/local/bin/python
```

- You can find out what program is called when you type "pip"
- Those should be in the same directory

```
Lisas-MBP-4:~ root# which pip
/usr/local/bin/pip
```

Terminal

- If you are inside an environment instead it will look like this

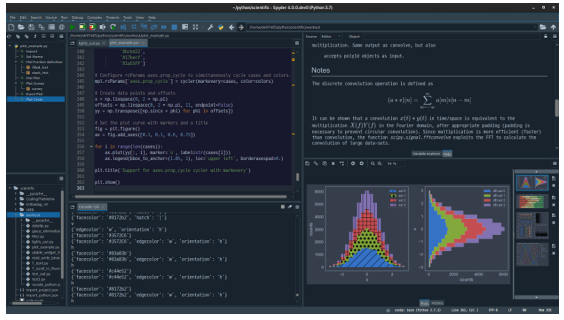


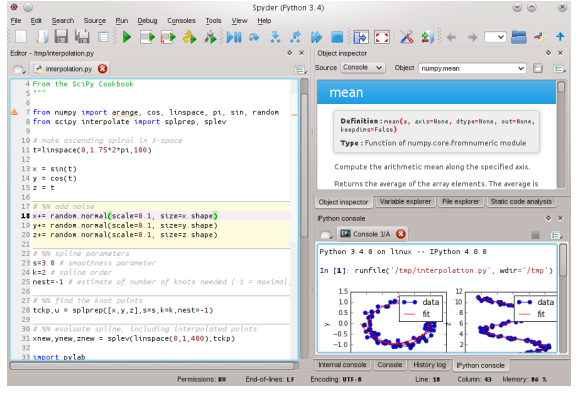
Terminal

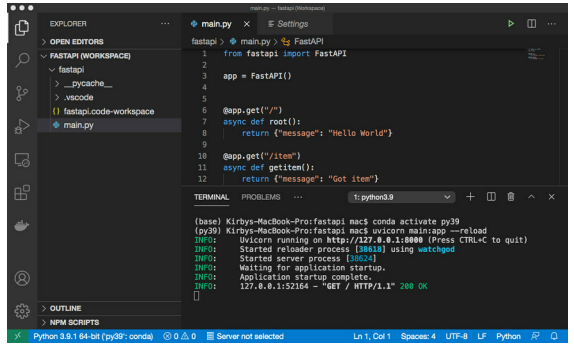
- In the terminal (and in all other programs) you can and should **deliberately set up which version it uses**
  - i.e. which Python the terminal calls when you type `python`
- depending on which Terminal shell you are using it may be set up in
  - bashprofile
  - .bashrc
  - .zshprofile
  - .profile
  - ...

Editors

- There are any number of editors you might use. Notably
  - VScode
  - pycharm
  - Spyder
- Most have **IDE capabilities**
  - set it up to run your code **directly** from the editor
  - tell the IDE **which version of python** you want to use *(usually which **path** to use)*

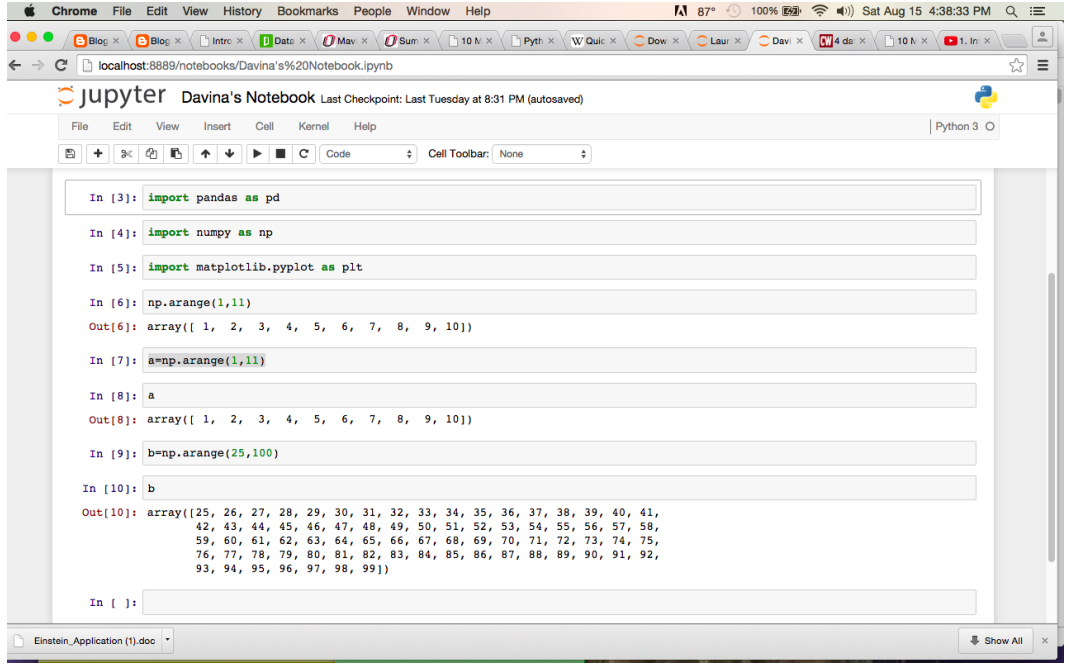






Notebooks

- Ipython/Jupyter Notebooks are really nice for data exploration
  - start it from inside the environment you want to use
    - `pip install jupyter`
    - `jupyter notebook`
  - (OR set up a **kernel)**



How to stay sane

- **Pick one system and stick with it**!
  - conda OR pip
- Identify which is your **base Python**
  - Do not install any packages directly there!
  - Do set up the base Python version as the **default** in the terminal and other programs like IDEs
- Whenever you want to start a project, make a **virtual environment** using the base Python
- In that environment install all your dependencies

Now let's get started!

Other Problems

- But I NEED Multiple pythons!
  - You could get really familiar with using the terminal and defining your own commands
  - Look into using pyenv or something similar
-