

# Setting up Python

EE599 Deep Learning

Kuan-Wen (James) Huang

Spring 2020



**USC**University of  
Southern California

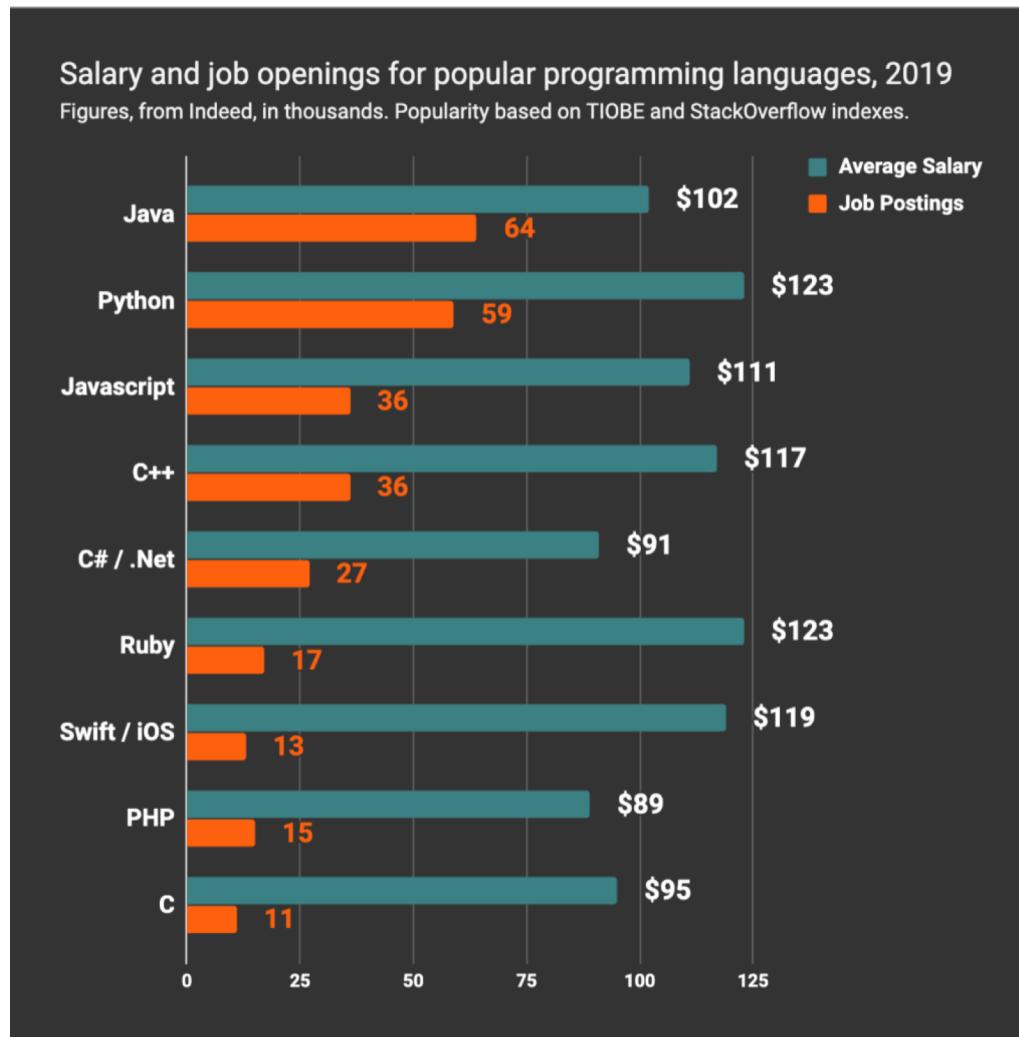
# About TAs

- Kuan-Wen (James) Huang
  - Office hours: Thursday 1-3pm at EEB522
  - Email: [kuanwenh@usc.edu](mailto:kuanwenh@usc.edu)
- Olaoluwa (Oliver) Adigun
  - Office hours: Tuesday, and Thursday 5-6:30pm at EEB420
  - Email: [adigun@usc.edu](mailto:adigun@usc.edu)
- Jiali Duan
  - Office hours: Friday 4-5pm at EEB B10
  - Email: [jialidua@usc.edu](mailto:jialidua@usc.edu)
- Arnab Sanyal
  - Office hours: Monday 8-10am at PHE320
  - Email: [arnabsan@usc.edu](mailto:arnabsan@usc.edu)

# Why Python but not Matlab?

- Python does nearly everything Matlab does, but it is open source and free.
  - Interpreted language (script) like Matlab
  - Vectorized operations
  - Plotting
  - Data handling, web-accessing
  - Extensive libraries for numerical computation and ML/DL
    - numpy, scipy, scikit-learn, NLTK, Keras, Tensorflow, Pytorch, etc.
  - Full-featured language
    - Classes/Objects, Inheritance, Modules

# Why Python but not Matlab?



- Source: <https://www.codeplatoon.org/the-best-paying-and-most-in-demand-programming-languages-in-2019/>

## 10 Best Programming Language to Learn in 2020

With time old programming languages become obsolete while new programming languages are launched, but they never gain traction. A common question amongst beginners (and coders alike) is the programming language they should invest learning in, that is in demand, stable outlook, and plenty of jobs.

Here, is a list of top 10 languages that you should learn -

### 1) Python



**Created:** Python language developed by Guido van Rossum. It was first released in 1991.

#### Pros:

- Supports multiple systems and platforms
- Object-Oriented Programming (OOPs) driven.
- Helps to improve Programmer's Productivity
- Allows you to scale even the most complex applications with ease
- Extensive Support Libraries

#### Cons:

- Note ideal for Mobile Computing
- Python's database access layer is bit underdeveloped and primitive.

- Source: <https://www.guru99.com/best-programming-language.html>

# Getting Started with Python

**piazza** EE 599 CHUGG ▾ Q & A Resources Statistics Manage Class Search Companies 1 Kuan-Wen Huang

Drafts | lecture logistics other discussion-session python  
Unread Updated Unresolved Following

New Post Search or add a post...

PINNED

- Instr Lecture Topics and Coverage 1/13/20 I plan to update this each lecture so that we can track what was covered when.  
1/13/2020: Class overview.Syllabus. Int
- Instr Syllabus 1/13/20 syllabus\_599\_deep\_learning.pdf – this is a mirror to the file uploaded on the syllabus tab under "Resources"
- Instr SLIDES (Lecture and Discus... 1/13/20 This post will contain all of the slide decks used in lectures and discussions. Updates will be placed inline and dated.
- Private Search for Teammates! 1/13/20

YESTERDAY

- How to downgrade Python from 3.7 to ... 10:12PM I have tried using command "conda install python==3.6" or just downgrading the version in Anaconda Navigator, bu
  - An instructor thinks there are good followups
- Instr MS Data Sciences and ML (...) 7:08PM on weds, 1/14, there will be an informational meeting on the new MS ECE in Data Sciences and Machine Learning. This wil
  - An instructor thinks this is a good note
- Instr controlling the flow of piazz... 9:29AM If you are getting too many emails from piazza (the default is every two hours) you can customize it. Go to the upper r
  - An instructor thinks this is a good note
- Instr Python resources 9:18AM This is a running list of resources for learning and using python. Feel free to comment and add your inputs. Note: W
  - An instructor thinks this is a good note

note ★ 65 views

**Python resources**

This is a running list of resources for learning and using python. Feel free to comment and add your inputs.

**Note:** We are standardizing on Python 3 -- preferably 3.6. Python 3.7 does not have support for Keras and Tensorflow yet. (Updated 1/9)  
Please do not use Python 2.

- Setting up Python:
  - Anaconda distribution
  - pyenv and pyenv-virtualenv
    - minimal Jupyter install file: [jupyter\\_min\\_reqs.txt](#)
- Popular editors and IDEs:
  - MS Code (my preference)
  - Sublime
  - PyCharm
- Getting started quickly:
  - [NumPy for Matlab users](#) and another [NumPy for Matlab users](#) <--- key reference
  - [Stanford CS231 tutorial](#)
  - [Gallery of sample plots using matplotlib](#)
- Full courses/books:
  - [Numerical Python by Robert Johansson](#) <-- recommended (has Jupyter Notebooks with many examples)
  - [Harrington's Python Tutorial](#)
  - [CS61a from UC Berkeley](#) -- freshman programming. Great full class, but slow going and not focused on numerical computing.
- Examples for keras:
  - [Github for Deep Learning with Python, by Chollet](#)
  - [Keras examples from keras.io](#)

python

~ An instructor (Arnab Sanyal) thinks this is a good note ~

edit good note 1 Updated 1 day ago by Keith Chugg

Getting Started with Python 8:17AM

EE599 will use  
Python 3.6

# Setting up Python Environment

## Anaconda

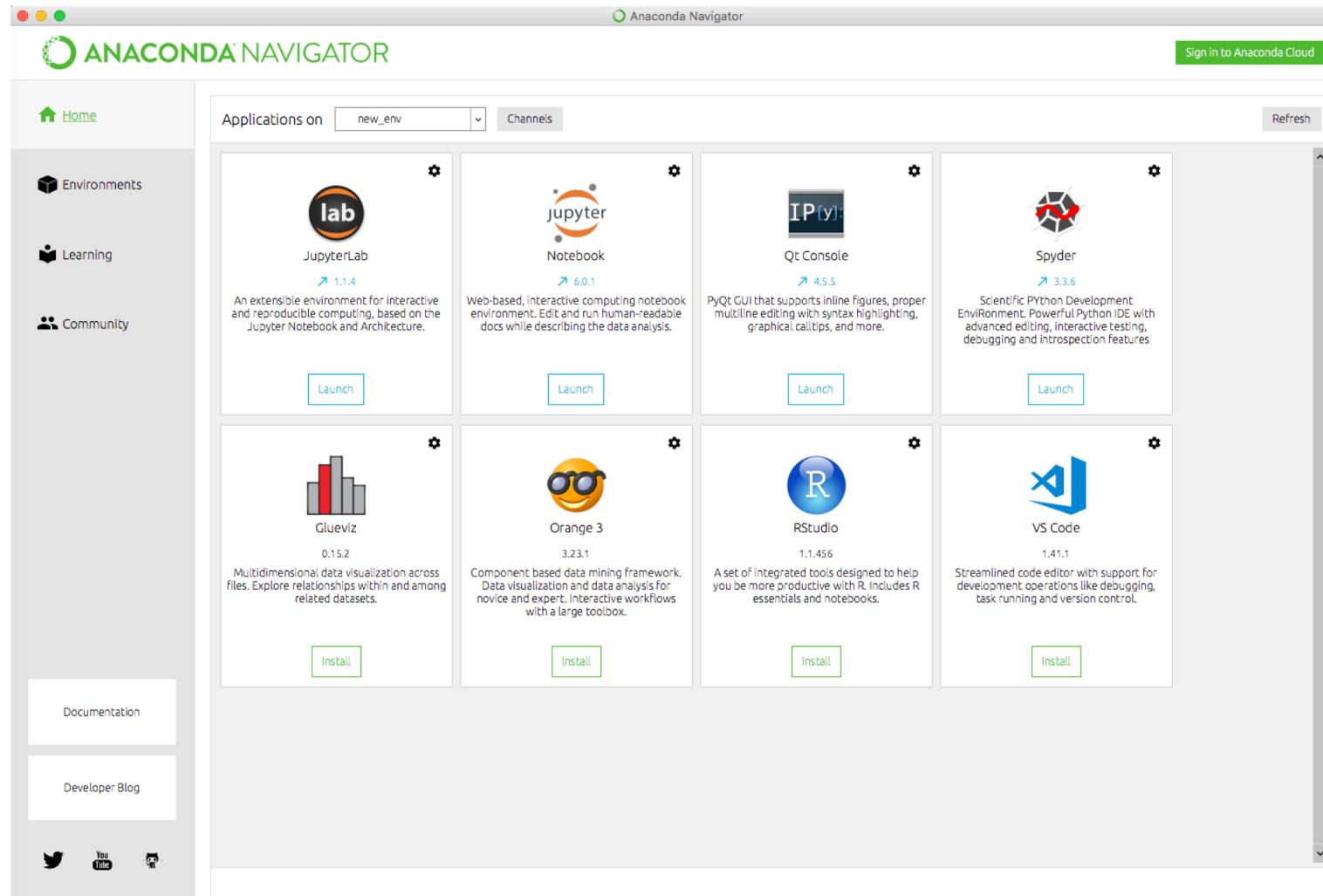
- Install just about everything you ever need (usually more than that..)
- Comes with some IDEs and Jupyter tools:
  - Spyder
  - Jupyter notebook
  - Qt Console

## pyenv

- Install the minimum you need for the projects
- Best professional option when working with others
  - Manages library versions w/o ambiguity

*you may have a system Python, best practice is not to modify that and instead do a separate install using one of the above*

# Python Workflows: Anaconda



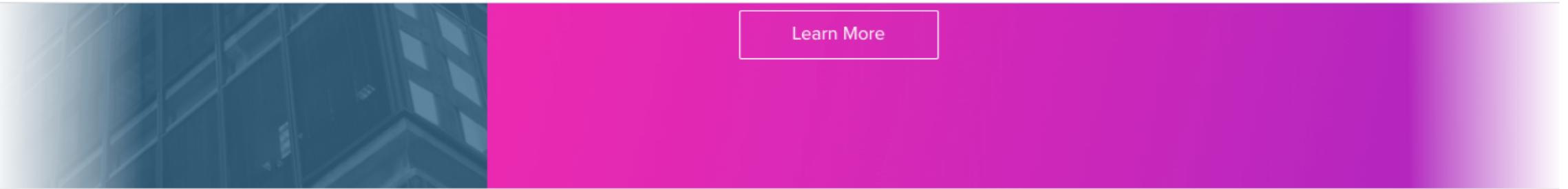
## Positives

Installs almost everything you will ever need  
conda package manager and update system  
has some useful packages — starts with navigator

## Negatives

takes about 8 GB  
Installs a lot stuff you may never use  
Not as simple to control package/library versions when working in teams

# Python Workflows: Anaconda



Learn More

## Solutions for Data Science Practitioners and Enterprise Machine Learning

### Anaconda Distribution

*The industry standard for open-source data science*

Supported by a vibrant community of open-source contributors and more than 18 million users worldwide, Anaconda Distribution is the tool of choice for solo data scientists who want to use Python or R for scientific computing projects.

[Download Now](#)

### Anaconda Team Edition

*Harness open-source building blocks for real data science*

Team Edition consists of a repository of over 7,500 data science and machine learning packages curated by Conda experts for security and reliability, mirrored on your enterprise infrastructure.

[Learn More](#)

### Anaconda Enterprise

*A full-featured platform for the machine learning life cycle*

Doing data science is hard, but getting machine learning models into the light of day, where they move a business forward – that's even harder. That's where Anaconda Enterprise (AE) comes in.

[Learn More](#)

# Python Workflows: Anaconda

After installing Anacoda, open terminal

```
conda create -n new_env python=3.6 anaconda
```

wait... type y and wait...

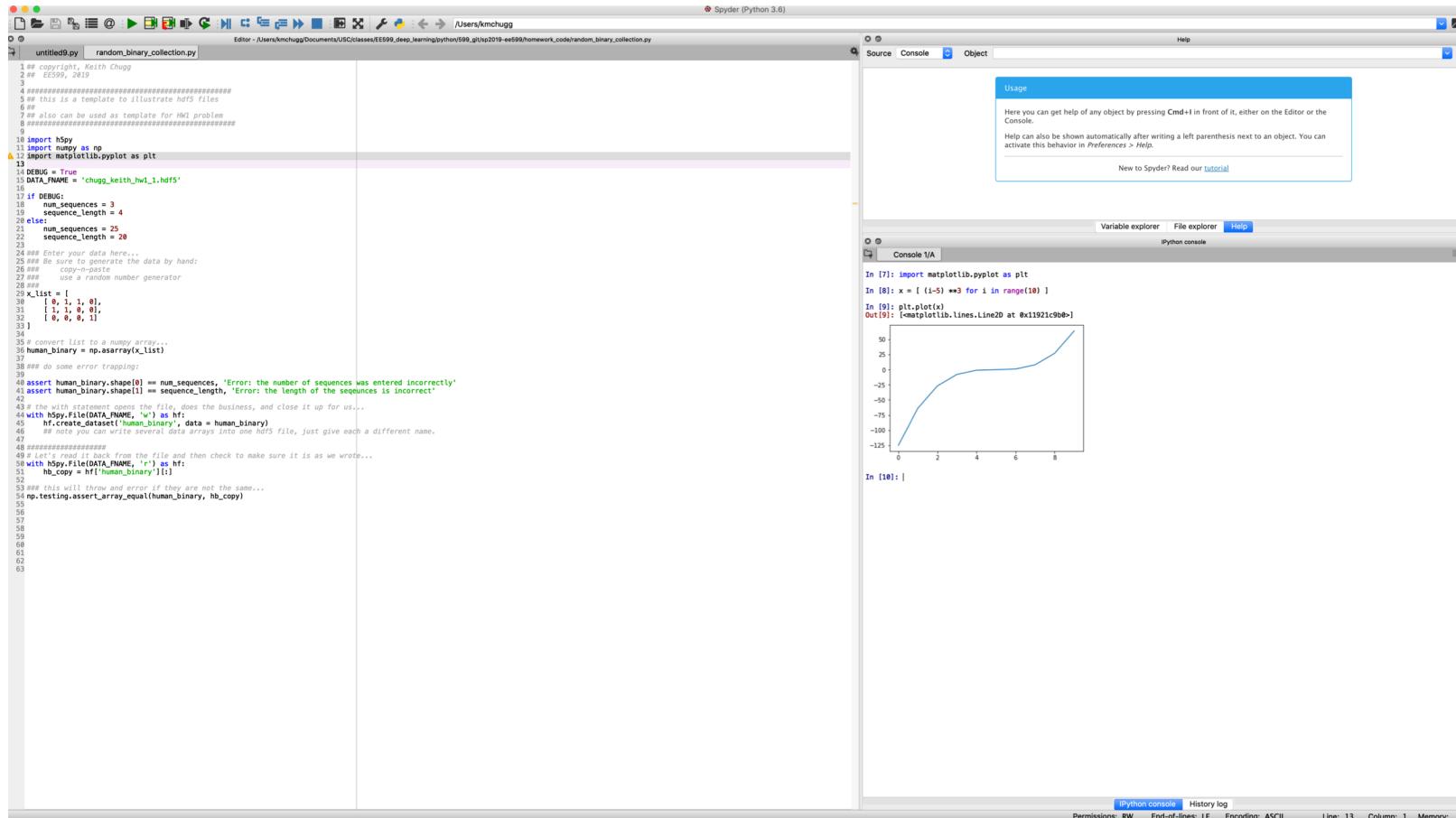
```
conda activate new_env
```

```
python --version should say 3.6.9
```

You can now work inside virtual environment “new\_env”

```
conda deactivate to quit virtual environment
```

# Python Workflows: Anaconda-Spyder



*Spyder IDE included with Anaconda:*

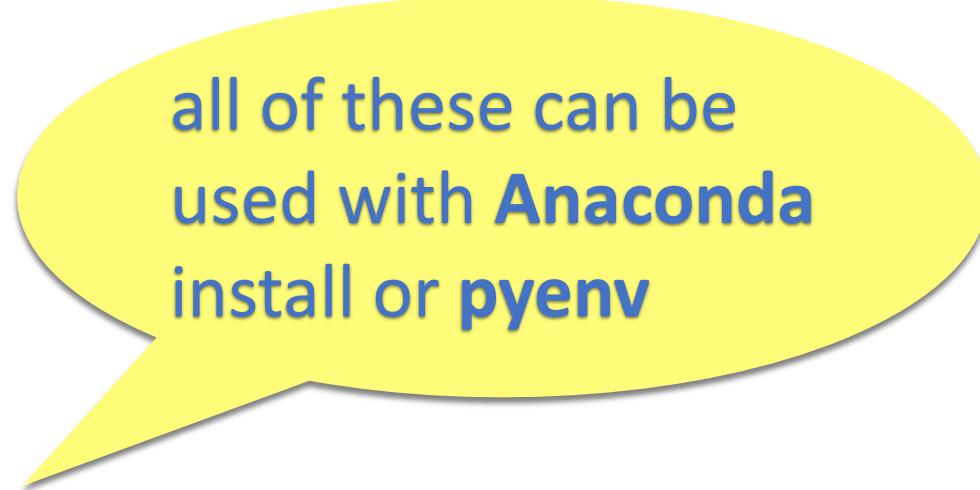
- + integrated with Jupiter qtConsole, includes debugger, simple, closest to Matlab
- editor is slow for larger files, similar flow can be replicated with editor of choice

# External Editors/Debuggers

- Language sensitive text editors
  - Sublime text, MS Code, BBEdit, etc.
- Some of these have additional features
  - Debugger for Python – e.g. breakpoints, variable inspection, etc.
  - Git integration, extension system, latex, support for other languages, etc.
- All of these can be used with Anaconda and/or pyenv
  - qtConsole, CLI, jupyter notebook

# Other Set-ups

- Jupyter Notebook
  - Web-based interactive development
  - Good for teaching or showing results to others
- Jupyter qtConsole
  - “Smart terminal” where you can plot
  - Good command history and completion
- CLI
  - Useful for “headless” operation – e.g. AWS, google cloud
  - Plots have to be saved and viewed elsewhere



all of these can be used with **Anaconda install** or **pyenv**

# Python Workflows: Editor + qtConsole

The image shows a desktop environment with two windows open. On the left is a code editor window titled "random\_binary\_collection.py" containing Python code. The code uses h5py and numpy to generate and manipulate binary sequence data. It includes sections for generating data by hand or using a random number generator, and for reading and writing to an HDF5 file named "chugg\_keith\_hw1\_1.hdf5". On the right is a "Jupyter QtConsole" window showing a plot of a mathematical function. The plot shows a curve starting at approximately (-10, -120) and increasing to about (10, 50). The console history shows the imports, the definition of the x-axis, and the resulting plot.

```
random_binary_collection.py
1  ## copyright, Keith Chugg
2  ## EE599, 2019
3
4  ##### this is a template to illustrate hdf5 files
5  ##
6  ## also can be used as template for HW1 problem
7  ##
8  #####
9
10 import h5py
11 import numpy as np
12 import matplotlib.pyplot as plt
13
14 DEBUG = True
15 DATA_FNAME = 'chugg_keith_hw1_1.hdf5'
16
17 if DEBUG:
18     num_sequences = 3
19     sequence_length = 4
20 else:
21     num_sequences = 25
22     sequence_length = 20
23
24 ### Enter your data here...
25 ### Be sure to generate the data by hand:
26 ### copy-n-paste
27 ### use a random number generator
28 ###
29 x_list = [
30     [ 0, 1, 1, 0 ],
31     [ 1, 1, 0, 0 ],
32     [ 0, 0, 0, 1 ]
33 ]
34
35 # convert list to a numpy array...
36 human_binary = np.asarray(x_list)
37
38 ### do some error trapping:
39
40 assert human_binary.shape[0] == num_sequences, 'Error: the number of sequences was entered incorrectly'
41 assert human_binary.shape[1] == sequence_length, 'Error: the length of the sequences is incorrect'
42
43 # the with statement opens the file, does the business, and close it up for us...
44 with h5py.File(DATA_FNAME, 'w') as hf:
45     hf.create_dataset('human_binary', data = human_binary)
46     # note you can write several data arrays into one hdf5 file, just give each a different name.
47
48 #####
49 # Let's read it back from the file and then check to make sure it is as we wrote...
50 with h5py.File(DATA_FNAME, 'r') as hf:
51     hb_copy = hf['human_binary'][:]
52
53 ### this will throw and error if they are not the same...
54 np.testing.assert_array_equal(human_binary, hb_copy)
55
56
57
58
59
60
61
```

In [4]: import matplotlib.pyplot as plt  
In [5]: x = [ (i-5) \*\*3 for i in range(10) ]  
In [6]: plt.plot(x)  
Out[6]: [

In [7]:

*not tied together, simple text editor + qtConsole*

# Python Workflows: MSCode + qtConsole

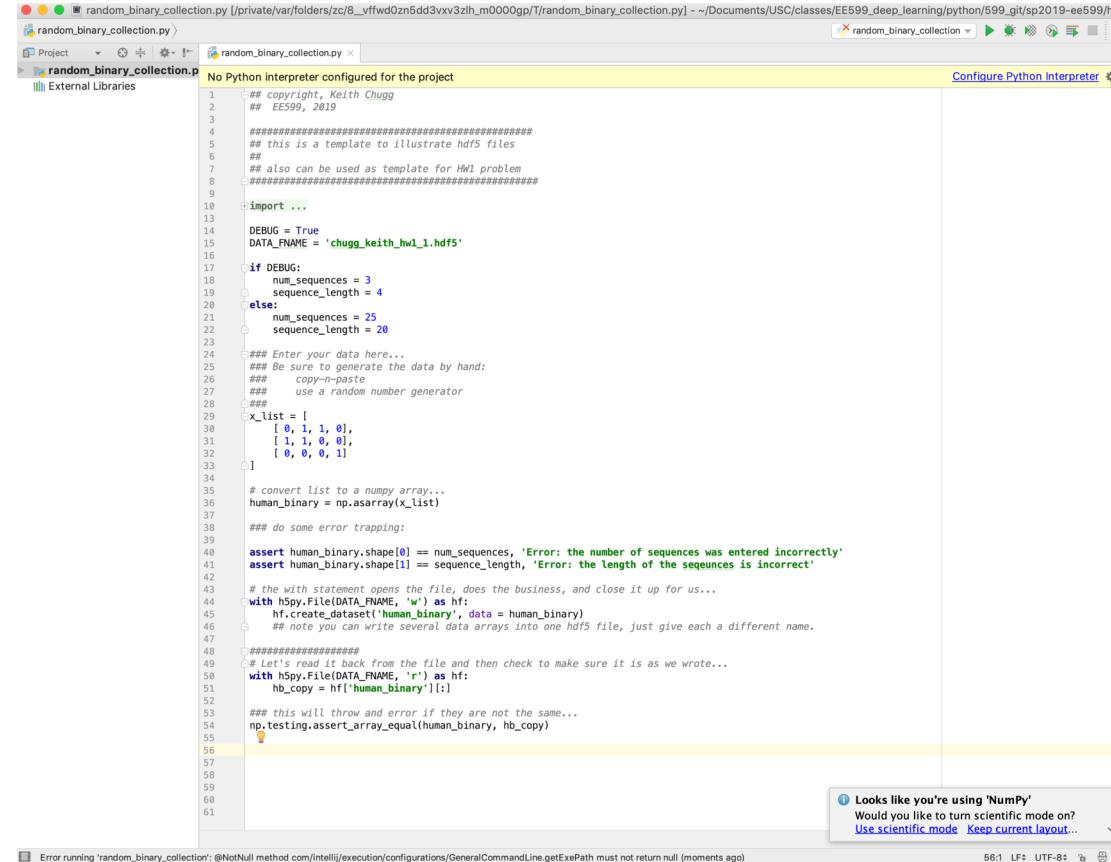
The screenshot shows a desktop environment with two windows open. On the left is the Microsoft Code editor window, which displays two open files: `simple_linear_regression.py` and `random_binary_collection.py`. The `simple_linear_regression.py` file contains Python code for performing linear regression and plotting the results. On the right is the Jupyter QtConsole window, which shows the execution of code in a notebook cell. The console output includes the imports, the creation of a list `x`, the plotting command `plt.plot(x)`, and the resulting plot showing a curve. Below the plot, the text `In [7]:` is visible.

```
simple_linear_regression.py — sp2019-ee599
simple_linear_regression.py random_binary_collection.py
1 ## copyright, Keith Chugg
2 ## EE599, 2019
3
4 from scipy import stats
5 import numpy as np
6 import matplotlib.pyplot as plt
7
8 x = np.arange(10)
9 y = 3*x+4
10 y = y + np.random.normal(0,2,10)
11 slope, intercept, r_value, p_value, std_err = stats.linregress(x,y)
12 y_hat = intercept + slope * x
13
14 fig = plt.figure()
15 plt.plot(x,y_hat, color='r')
16 plt.scatter(x,y)
17 plt.xlabel("x")
18 plt.ylabel("estimate of y")
```

```
In [4]: import matplotlib.pyplot as plt
In [5]: x = [ (i-5) **3 for i in range(10) ]
In [6]: plt.plot(x)
Out[6]: [matplotlib.lines.Line2D at 0x10e06fa90>]
```

*MS Code can be used a text editor in this workflow and/or you can run and debug inside MS Code — i.e., it is an IDE*

# Python Workflows: PyCharm



```
## copyright, Keith Chugg
## EES99, 2019

#####
## this is a template to illustrate hdf5 files
##
## also can be used as template for HW1 problem
#####

import ...

DEBUG = True
DATA_FNAME = 'chugg_keith_hw1_1.hdf5'

if DEBUG:
    num_sequences = 3
    sequence_length = 4
else:
    num_sequences = 25
    sequence_length = 20

### Enter your data here...
### Be sure to generate the data by hand:
### copy-n-paste
### use a random number generator
###

x_list = [
    [ 0, 1, 1, 0],
    [ 1, 1, 0, 0],
    [ 0, 0, 0, 1]
]

# convert list to a numpy array...
human_binary = np.asarray(x_list)

### do some error trapping:
assert human_binary.shape[0] == num_sequences, 'Error: the number of sequences was entered incorrectly'
assert human_binary.shape[1] == sequence_length, 'Error: the length of the sequences is incorrect'

# the with statement opens the file, does the business, and close it up for us...
with h5py.File(DATA_FNAME, 'w') as hf:
    hf.create_dataset('human_binary', data = human_binary)
    # note you can write several data arrays into one hdf5 file, just give each a different name.

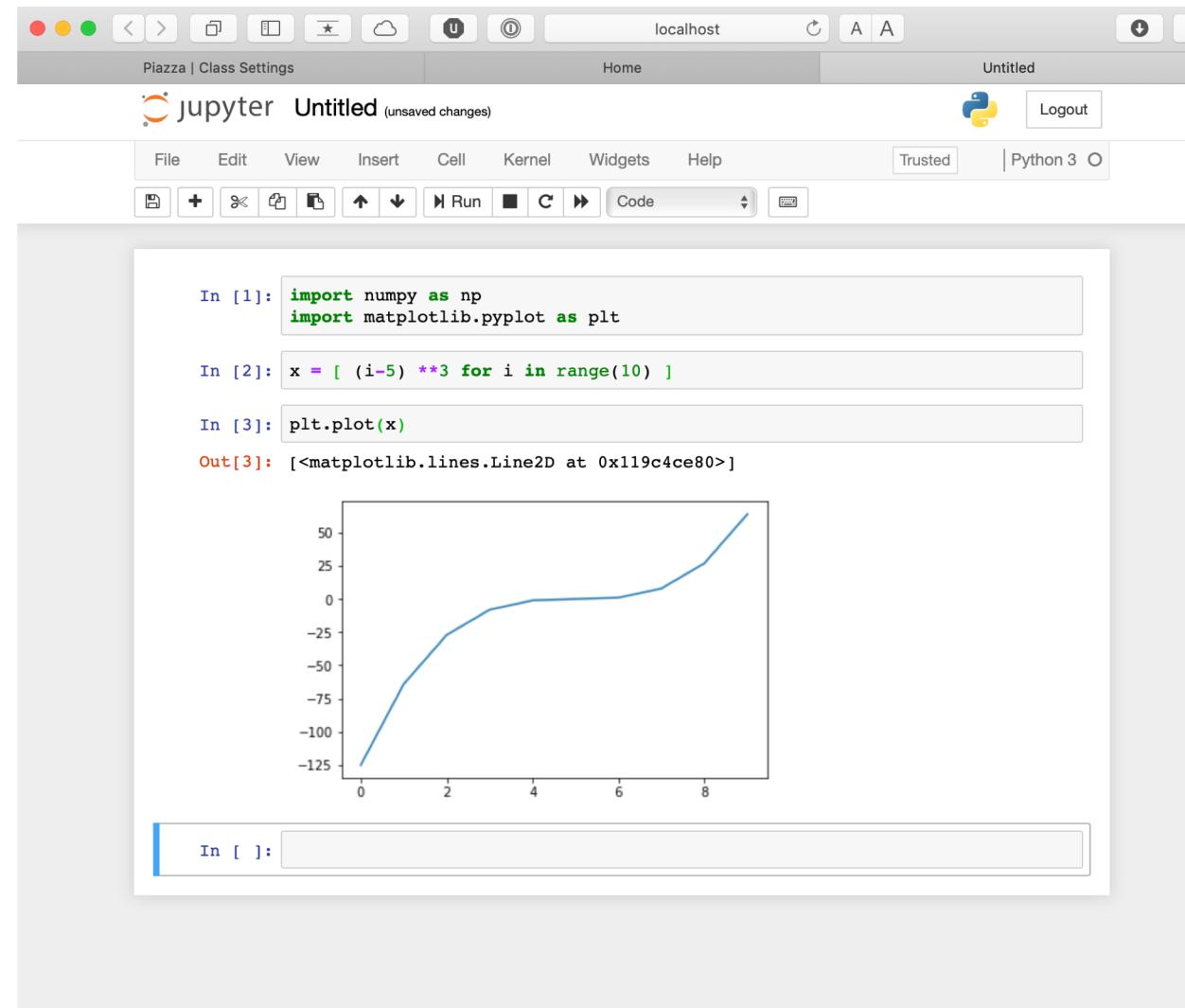
#####
# Let's read it back from the file and then check to make sure it is as we wrote...
with h5py.File(DATA_FNAME, 'r') as hf:
    hb_copy = hf['human_binary'][()]

### this will throw and error if they are not the same...
np.testing.assert_array_equal(human_binary, hb_copy)

Looks like you're using 'NumPy'
Would you like to turn scientific mode on?
Use scientific mode Keep current layout...
```

- + *more polished than Spyder, more widely used. Supports pyenv and Anaconda*
- *Slows down when using Anaconda install because it indexes libraries*

# Python Workflows: Jupyter Notebooks



# Python Workflows: pyenv Virtual Environment

- To install pyenv, check its websites
  1. <https://github.com/pyenv/pyenv>
  2. <https://github.com/pyenv/pyenv-virtualenv>

*The second link has a good description of how to:*

- *install various Python versions (e.g., 2.7.x, 3.6.9, 3.7.1)*
- *create, manage and delete virtual-environments on top of these base Python installs*

# pyenv Virtual Environment

pyenv install -l	list available python base python versions
pyenv install 3.6.9	install python 3.6.9 as a base version
pyenv virtualenv 3.6.9 new_env2	create a new virtualenv new_env2
pyenv activate new_env2	activate new_env2— this is your current python environment
pip install -r requirements.txt	install libraries from a specified list
pip install numpy	add other libraries as needed
pip freeze	list the libraries installed in new_env2
pip freeze > new_requirements.txt	save the new set-up to replicate elsewhere

pyenv virtualenvs	list all your virtaulenvs
pyenv uninstall new_env2	deletes the virtual environment

Issue with many open source solutions is minor incompatibilities by version

This approach allows you to set up and replicate on multiple machines the same python set-up (including all versions)

# Final Word

Be proactive and try to figure things out

