IAHR & IWRA Student Chapters
University of Illinois at Urbana Champaign

October 5, 2019

Urbana, IL

IAHR & IWRA UIUC Python Workshop: Basic Python

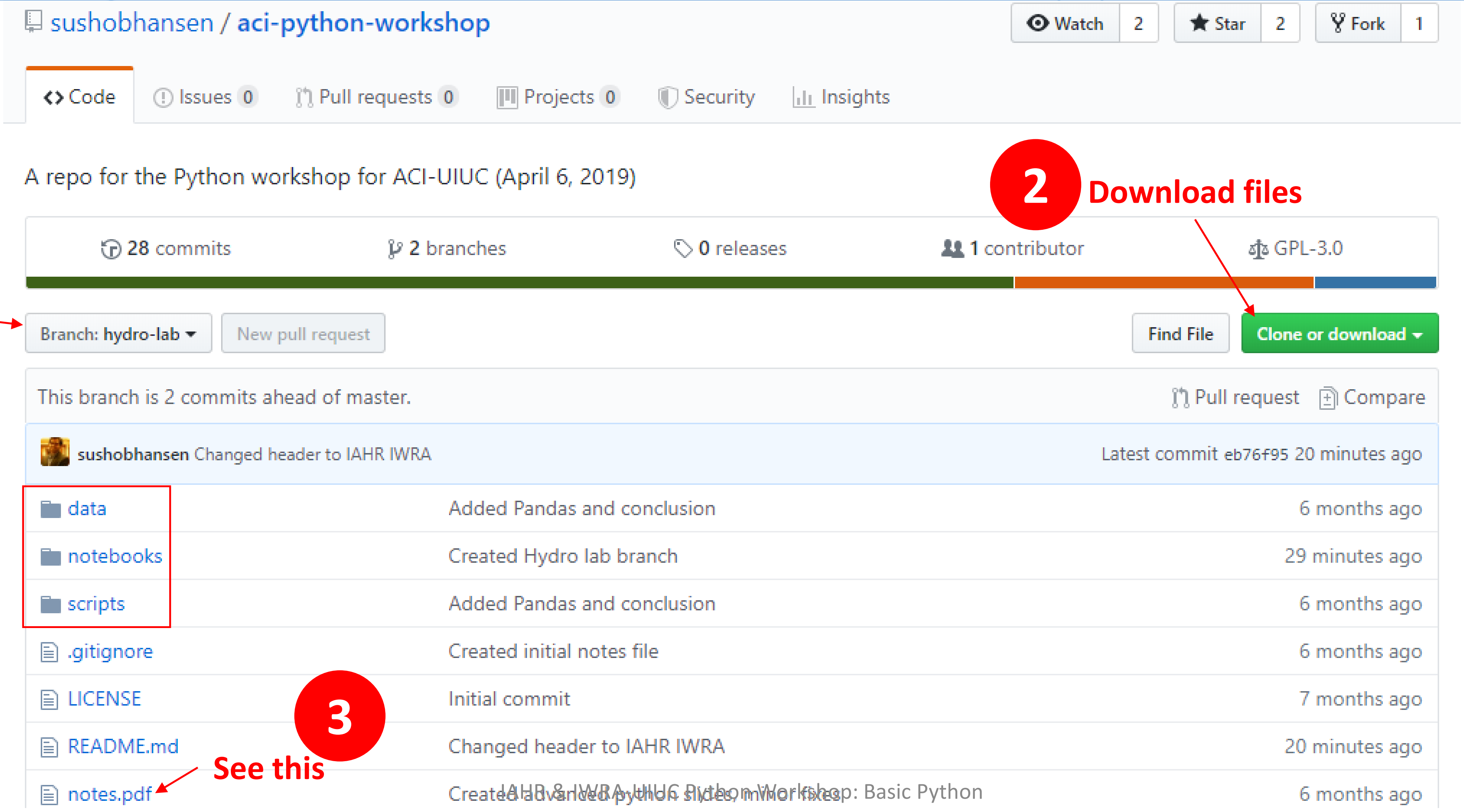
Before we begin…

- If you haven't already, go to https://github.com/sushobhansen/aci-python-workshop for course materials
- Download and install Anaconda: https://www.anaconda.com/distribution/
- Or use an online IDE: https://repl.it/languages/python3 (won't work too well though, using the terminal is better)

Who am I?

- Sushobhan Sen
- Doctoral candidate in Civil Engineering (Transportation) – I study stratified urban boundary layer flows
- Languages I know: FORTARN, C/C++/C#, Python, Visual Basic, HTML, CSS, JavaScript, Latex, Bash
  - (I also speak a few human languages)
- More at: https://sushobhansen.github.io/
- E-mail: sen6@illinois.edu

Workshop Website: https://github.com/sushobhansen/aci-python-workshop



sushobhansen / aci-python-workshop

Watch 2 | Star 2 | Fork 1

Code | Issues 0 | Pull requests 0 | Projects 0 | Security | Insights

A repo for the Python workshop for ACI-UIUC (April 6, 2019)

28 commits | 2 branches | 0 releases | 1 contributor | GPL-3.0

Branch: hydro-lab | New pull request | Find File | Clone or download

This branch is 2 commits ahead of master. Pull request | Compare

sushobhansen Changed header to IAHR IWRA — Latest commit eb76f95 20 minutes ago

data	Added Pandas and conclusion	6 months ago
notebooks	Created Hydro lab branch	29 minutes ago
scripts	Added Pandas and conclusion	6 months ago
.gitignore	Created initial notes file	6 months ago
LICENSE	Initial commit	7 months ago
README.md	Changed header to IAHR IWRA	20 minutes ago
notes.pdf	Created advanced python slides, minor fixes	6 months ago

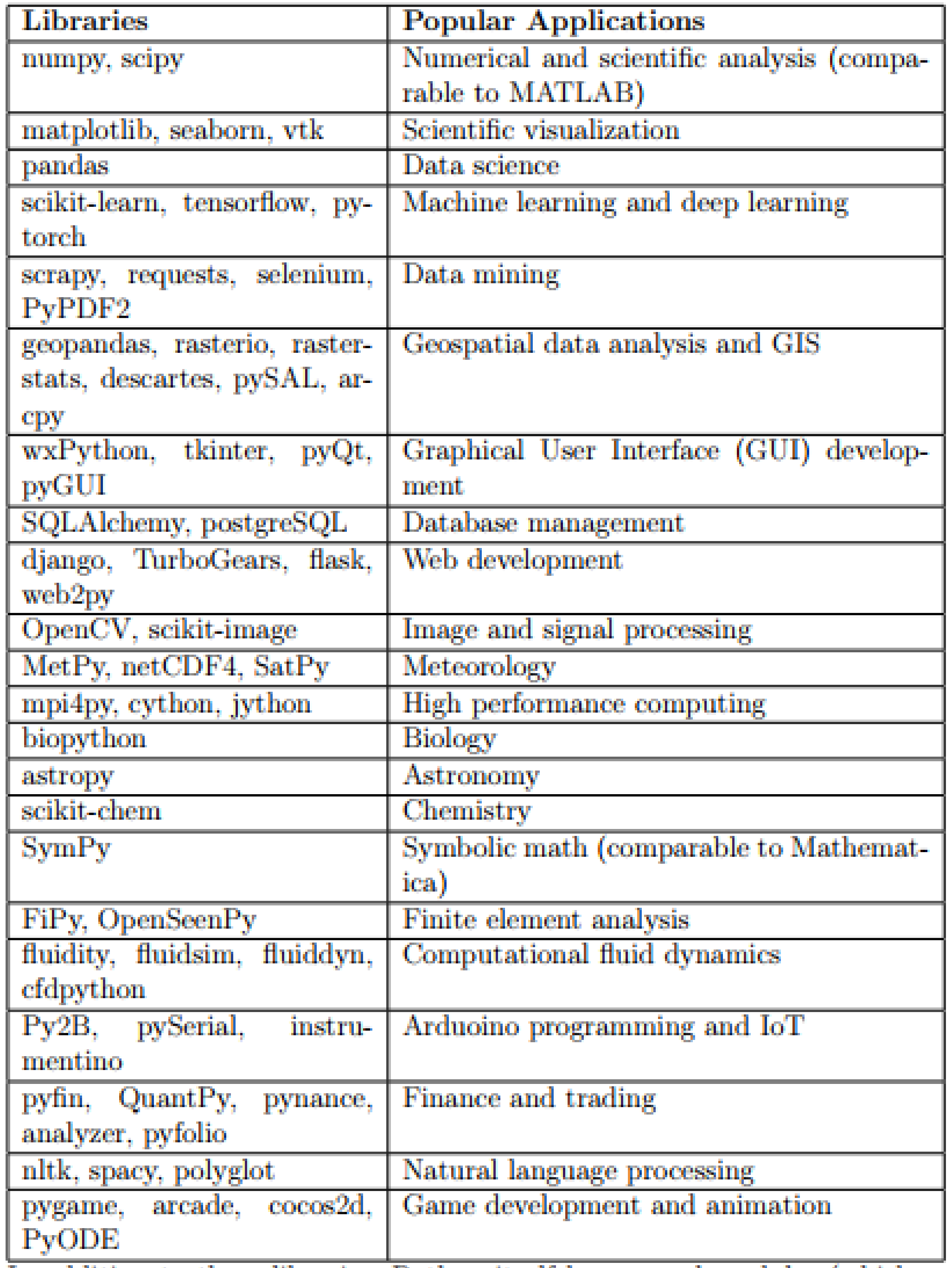
Learning Objectives: Advanced Python

- At the end of this workshop, participants will be able to:
  - Use numpy to define and use matrices, and read and process data from files
  - Use matplotlib to visualize data
  - Use scipy to process an image, analyze a sound wave, and fit data to a curve
  - Use pandas to read and analyze data (if we have time)
- We'll do all of this by solving several problems

Python Libraries

- You already know how to use Python (you should)
- Libraries extend Python’s functionalities
- Almost all of them are open-source and free
- Some have great documentation, others iffy

Python Libraries



Libraries	Popular Applications
numpy, scipy	Numerical and scientific analysis (comparable to MATLAB)
matplotlib, seaborn, vtk	Scientific visualization
pandas	Data science
scikit-learn, tensorflow, pytorch	Machine learning and deep learning
scrapy, requests, selenium, PyPDF2	Data mining
geopandas, rasterio, rasterstats, descartes, pySAL, arcpy	Geospatial data analysis and GIS
wxPython, tkinter, pyQt, pyGUI	Graphical User Interface (GUI) development
SQLAlchemy, postgreSQL	Database management
django, TurboGears, flask, web2py	Web development
OpenCV, scikit-image	Image and signal processing
MetPy, netCDF4, SatPy	Meteorology
mpi4py, cython, jython	High performance computing
biopython	Biology
astropy	Astronomy
scikit-chem	Chemistry
SymPy	Symbolic math (comparable to Mathematica)
FiPy, OpenSeenPy	Finite element analysis
fluidity, fluidsim, fluiddyn, cfdpython	Computational fluid dynamics
Py2B, pySerial, instrumentino	Arduoino programming and IoT
pyfin, QuantPy, pynance, analyzer, pyfolio	Finance and trading
nltk, spacy, polyglot	Natural language processing
pygame, arcade, cocos2d, PyODE	Game development and animation

Using libraries in Python

Import the whole library

```python
import numpy as np
```

Alias

Import only some functions

```python
from scipy.optimize import curve_fit
```

Numpy

- Numerical Python library
  - Implements vectorization
  - Implements fast linear algebra libraries (BLAS and LAPACK)
- Very similar to using MATLAB – many of the functions have the same name!
- Numpy works on arrays or matrices as a whole – you shouldn’t have to manipulate element by element

Numpy arrays

- Main data structure when using numpy
- A continuous block of memory

```
import numpy as np
a = np.array([[1,2,3],[4,5,6],[7,8,9]])
print(a)
print(type(a))
print(a.shape)
```

Go to Notebook

Array slicing and broadcasting

- **Slicing:** Using only parts of an array [row:column]
    - Numpy arrays are zero indexed, exclusive of second number
- **Broadcasting**: Converting one array dimension to another compatible one

Go to Notebook ➡

Vectorization and Masking

- **Vectorization:** Manipulate all elements of an array simultaneously
- **Masking:** A Boolean array that can be passed to another array to get only some elements (vectorized version of an if-condition)

Go to Notebook

Challenge: Linear Algebra

- Solve: $\begin{cases} x + y + z = 6 \\ 2y + 5z = -4 \\ 2x + 5y - z = 27 \end{cases}$

$$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 2 & 5 \\ 2 & 5 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 6 \\ -4 \\ 27 \end{pmatrix}$$

A*x = b

Go to Notebook

One library to rule them all

- Numpy is considered a fundamental library, so much so it’s like a part of Python itself
- Almost all major numerical Python libraries are built on top of numpy
- We’ll see more numpy functions later

Matplotlib

- Matrix Plotting Library
- PyPlot – Python API
- Great for scientific data visualization, comparable to MATLAB
- Easy to make publication-quality charts

Plotting 1D Data

- Usually categorical variable on x-axis, continuous variable on y-axis
- E.g., Bar chart, Pie chart

```
%matplotlib inline
import numpy as np
from matplotlib import pyplot as plt

barnumber = np.arange(4)+1
barheight= np.random.uniform(5.0,10.0,(4,))

plt.bar(barnumber,barheight)
plt.xlabel('Bar Number')
plt.ylabel('Bar Height')
plt.title('Random Bar Heights')
plt.ylim([0,10])
plt.xticks(barnumber)
plt.show()
```

Go to Notebook

Plotting 2D Data

- Typical type of plotting
- Usually plot a function of the form $y = f(x)$
- Plot **numerically** i.e., plot points, maybe join with a curve
- E.g., line plot, scatter plot

Plotting a trigonometric function

- Plot $f(x) = \frac{\sin x + \cos x}{2}$ for $x \in [0°, 360°]$

```
x = np.linspace(0,360,1000)
y = (np.sin(x*np.pi/180.) + np.cos(x*np.pi/180.))/2.0
plt.plot(x,y)
plt.grid()
plt.xlabel('x (degrees)')
plt.ylabel('f(x)')
plt.show()
```

Go to Notebook

Multiple series in a single plot

- Plot $\begin{cases} e^{-x} \\ e^{-2x} \end{cases}$ for $x \in [0,2]$

```python
x = np.linspace(0.,2.,1000)
y1 = np.exp(-x)
y2 = np.exp(-2.0*x)

plt.semilogy(x,y1,label='$e^{-x}$')
plt.semilogy(x,y2,label='$e^{-2x}$')
plt.xlabel('x')
plt.ylabel('f(x)')
plt.legend()
plt.grid(which='both')
plt.show()
```

Go to Notebook

Subplot & Data from a File

- Plot data from data/weather.csv as a line and bar chart

```
hour,temperature,precip_prob =
        np.loadtxt('../data/weather.csv', dtype=np.float,
        delimiter=',',skiprows=1,unpack=True)

plt.subplot(1,2,1)
plt.plot(hour,temperature)
plt.xlabel('Hour')
plt.ylabel('Temperature ($^o$F)')
plt.title('Temperature')

plt.subplot(1,2,2)
plt.bar(hour,precip_prob)
plt.xlabel('Hour')
plt.ylabel('Probability (%)')
plt.title('Probability of Precipitation')
plt.show()
```

Go to Notebook

Plotting 3D Data

- Two ways:
  - Actual 3D isometric view (good for web apps, not good for publication): Not directly supported, available from mpl_toolkits
  - Contour plots (good for publication)
- Always plot over a *mesh* of points

Plot a 3D function

- Plot $f(x,y) = \sin(\sqrt{x^2 + y^2})$ for $(x,y) \in [-6,6]^2$

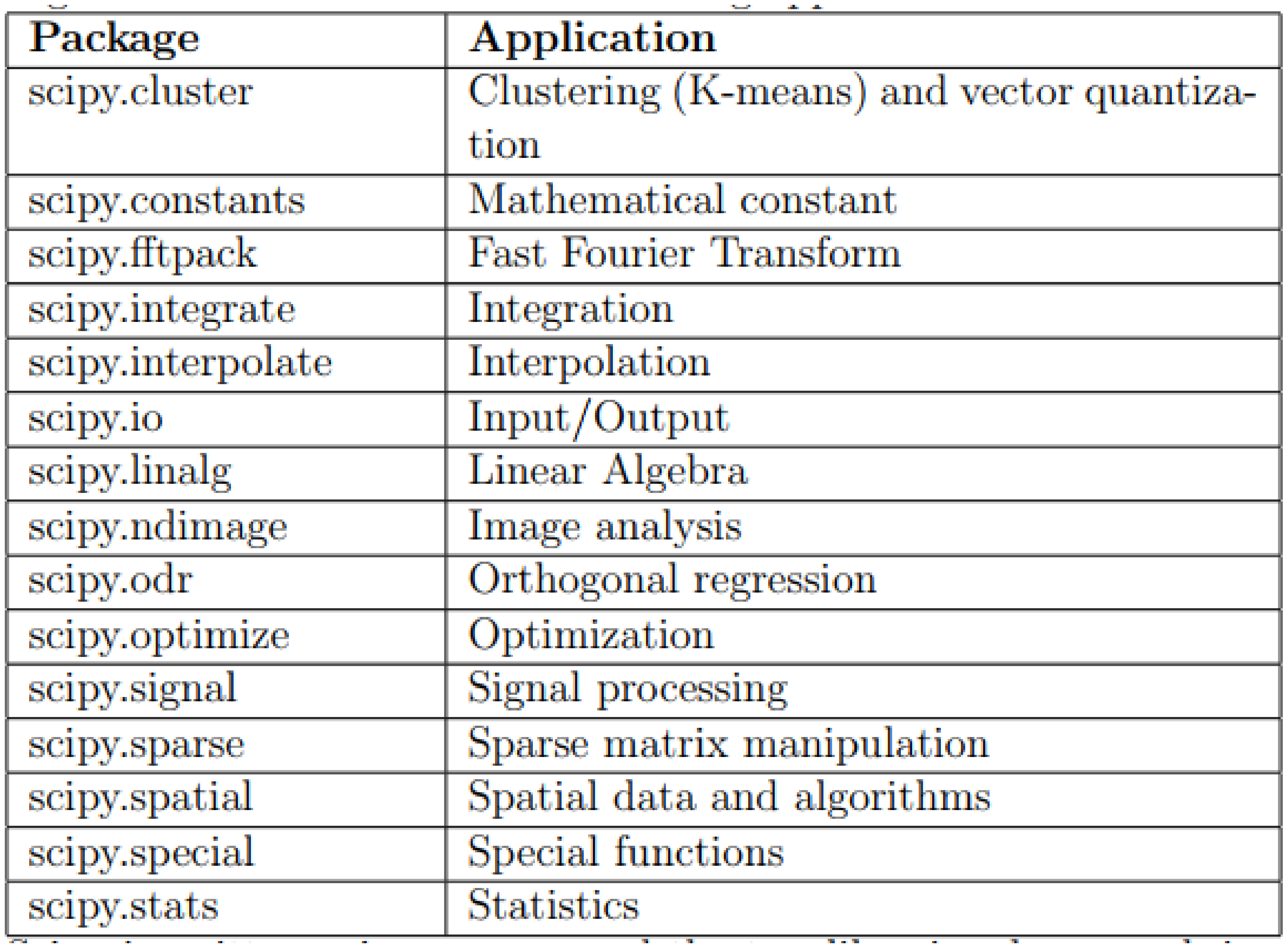
```python
from mpl_toolkits import mplot3d

x = np.linspace(-6,6,100)
y = np.linspace(-6,6,100)

X,Y = np.meshgrid(x,y)
Z = np.sin(np.sqrt(X**2+Y**2))

fig = plt.figure()
ax = plt.axes(projection='3d')

ax.contour3D(X, Y, Z, 50, cmap='spring')
ax.set_xlabel('x')
ax.set_ylabel('y')
```

```python
plt.contourf(X,Y,Z,50,cmap='spring')
plt.xlabel('x')
plt.ylabel('y')
plt.colorbar(label='f(x,y)')
plt.show()
```

Go to Notebook

Scipy

- Scientific Python Library
- Built on top of numpy, and follows it very closely (many common functions too)



Package	Application
scipy.cluster	Clustering (K-means) and vector quantization
scipy.constants	Mathematical constant
scipy.fftpack	Fast Fourier Transform
scipy.integrate	Integration
scipy.interpolate	Interpolation
scipy.io	Input/Output
scipy.linalg	Linear Algebra
scipy.ndimage	Image analysis
scipy.odr	Orthogonal regression
scipy.optimize	Optimization
scipy.signal	Signal processing
scipy.sparse	Sparse matrix manipulation
scipy.spatial	Spatial data and algorithms
scipy.special	Special functions
scipy.stats	Statistics

Challenge: Curve Fitting

- Use scipy.optimize.curve_fit() to fit an Intensity-Duration-Frequency (IDF) curve using data from data/idf.csv of the following form:
  - $I(t) = \frac{a}{t^n + c}$ (Chow et al)
  - $I(t)$ is intensity (in/hr) as a function of storm duration (t) in hours.
  - a, n, c are empirical constants that are a function of the return period (frequency, in years)

Go to Notebook

Challenge: Image Processing

- Use scipy.ndimage to perform the following manipulations on the data/otter.png image:
  - View it
  - Rotate it by $10°$
  - Sharpen it (use a Gaussian filter)
  - Detect its edges (use a Sobel filter)

Go to Notebook

Challenge: Fourier Transform

- Use scipy.fftpack.fft() to find the dominant frequency in the data/cat.csv file (it’s a cat purring, listen to data/cat.wav)

Go to Notebook

Pandas

- A high-level data analysis package built on top of numpy, with further abstractions
- Most popular package for data science
- Basic data types: Series (1D) and Data Frames (2D)

Challenge: 2013 College Football Drives

- Read the data/drive.csv file in a pandas data frame, examine the data, and clean it

```
import numpy as np
import pandas as pd

df = pd.read_csv('../data/drive.csv')
df.head()
```

```
print(df.shape, df.columns, df.dtypes)
df['Game Code'][10:25]
df.describe()
```

```
df.isna().sum()
```

```
df = df.dropna()
```

Go to Notebook

Challenge: 2013 College Football Drives

- Using the pandas data frame, analyze the data and answer the following questions:
  - What was the distribution of the ways that drives ended?
  - Of the drives that ended in a touchdown, what was the distribution of the time of possession?
  - Which team made the most yards per play on average?

Go to Notebook ➡

Conclusion

(EW.com)

Further Resources

- Numpy and Scipy documentation: https://docs.scipy.org/doc/
- Matplotlib documentation: https://matplotlib.org/contents.html
- Pandas documentation: https://pandas.pydata.org/pandas-docs/stable/
- Free online courses:
    - Python for research: https://www.edx.org/course/using-python-for-research
    - Python for data science: https://www.edx.org/course/python-for-data-science-1