BUDT 758X
Data Processing and Analysis in Python
Spring 2018

Prof. Sean Barnes
Dept. of Decision, Operations, and Information Technologies
Robert H. Smith School of Business
University of Maryland, College Park

Agenda – January 23

- Course Logistics
  - Syllabus, Canvas, Piazza, DataCamp
- Introduction to Python
- Brief Tour of Python Tools

COURSE LOGISTICS

Academic Integrity

	Class Notes/Slides	**Textbook**	**Online Resources**	**Peers**	**Instructor/ TAs**
Class Participation	**OK**	**OK**	LIMITED	LIMITED	**OK**
In-Class Exercises	**OK**	**OK**	LIMITED	**OK**	**OK**
DataCamp	**OK**	**OK**	LIMITED	**LIMITED**	LIMITED
Homework Assignments	**OK**	**OK**	**LIMITED**	**LIMITED**	**OK**
Exam	NOT OK	NOT OK	NOT OK	**NOT OK**	LIMITED
Project	**OK**	**OK**	**OK**	LIMITED	**OK**

Blueprint for Success

Actively prepare for class (e.g., reading the sections listed in the Syllabus in advance, complete DataCamp modules)

Regularly attending class AND actively participating (e.g., ask questions, follow along with examples)

Regularly attend office hours

Study efforts outside of class (e.g., reviewing text and slides, practice examples, DataCamp, homework)

Online engagement (e.g., asking questions via email, Canvas, and/or Piazza)

Tasks for This Week

- **Read the syllabus thoroughly!!**
- Purchase the text (and begin reading!)
- Explore Canvas, DataCamp, and Piazza
- Install/access and test software
- Begin working through DataCamp modules

Class Dynamics

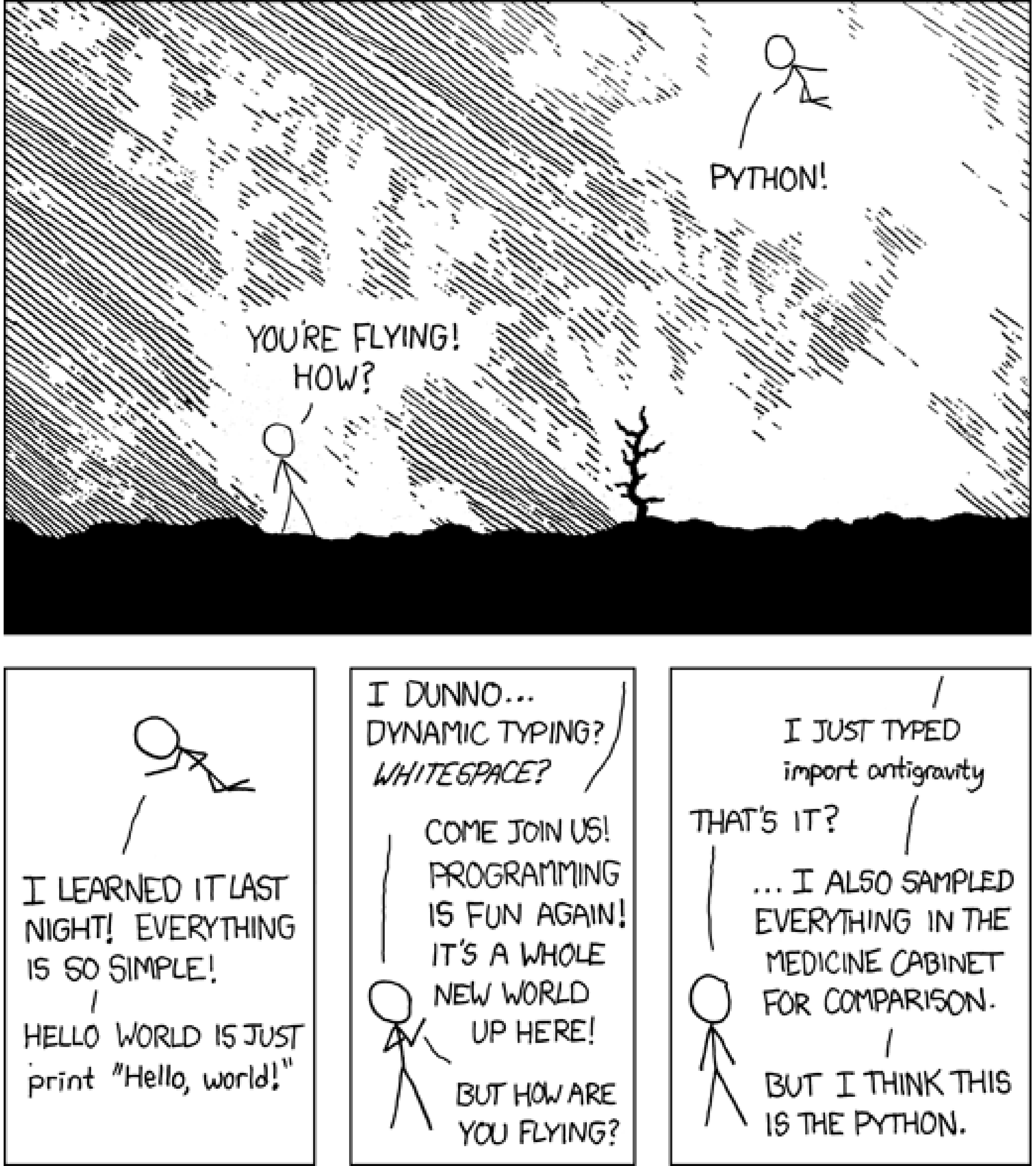
- Programming is a difficult subject to learn, and a difficult subject to teach
  - We will need to find a balance between addressing Q&A and progressing through our course plan
  - When following along with in-class demonstrations, figure out what works best for your learning style
    - In the end, you will learn by doing more than by observing
- Classroom
  - Not ideal, but we will manage!

ANY QUESTIONS??

INTRODUCTION TO PYTHON

What is Python?

- Open-source, object-oriented programming language with extensive capabilities
- User-oriented language – Relatively easy to learn
- Well documented, broad user base
  - Lots of resources for learning
- Runs very fast (for an interpretable language)
- Interpretable language (i.e., no compiling)
  - "Execute-explore" vs. "Edit-compile-run"
- Integration with other tools (e.g., R, C/C++, databases)


YOU'RE FLYING! HOW?
PYTHON!
I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"
I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?
I JUST TYPED import antigravity
THAT'S IT?
... I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.

xkcd

What can Python be used for?

Processing numerical or textual data

File management and conversion

Automate work flows

Quantitative and statistical analysis

Modeling (e.g., optimization, simulation)

Data visualization

Network analysis

Web development

What we’re going to use Python for

Processing data sets

- Importing external data
- Cleaning data
- Merging and transforming data
- Natural language processing

Data exploration

- Descriptive statistics
- Data visualization

Modeling

- Statistical modeling and machine learning
- Optimization
- Simulation

What kind of data?

- Tabular or spreadsheet-like data
  - Excel (.xlsx) or tab- (.txt) or comma-delimited (.csv) files
- Multiple tables of data interrelated by key columns
- Web-based data (i.e., scraping)
- Unstructured text files
- Structured data files (e.g., .html, .xml, .json)
- Multidimensional numerical arrays
- Evenly or unevenly spaced time series

Some (Realistic) Goals

- A semester is probably not long enough to become a Python expert…
- …BUT, by the end of this course, you should:
  - Become comfortable reading, writing, and executing Python code
  - Gain intermediate skill working with several powerful Python modules (i.e., libraries)
  - **Learn how to be resourceful**

Being Patient

- Resilience
- Learning from mistakes
- Break your code into parts

Being the Code

Being Resourceful

Don't recreate the wheel! Check references for desired functionality or code

- Text and online documentation are full of examples
- "Google It"
  - Search python (or specific module) + short description (e.g., sort list, scatter plot)
  - Tutorials, Forums, Videos

Learn how to understand function/method descriptions

- Required vs. optional arguments

"Essential" Python Modules

NumPy

- Foundational package for scientific computing
- Multidimensional array objects and computational functions

matplotlib

- Data visualization

SciPy

- Collection of packages for performing linear algebra, statistics, optimization, and more

pandas

- Rich data structures and functions to facilitate data processing and analysis

Other Common Add-On Modules

Seaborn: Pretty data visualization package built on matplotlib
statsmodels: Statistical analysis package
scikit-learn: Machine learning/data mining package
nltk: Natural language tool-kit -- text analytics
SimPy: Discrete-event simulation package
NetworkX: Graph/network generation and social network analysis
Parallel Python: Parallel computing
Cython: Integration with C/C++
rpy2: Integration with R
Django, webpy: Web development
Big Data: Numba, Blaze, wiseRF, PySpark, Pydoop, Theano
Modules available to interact with various data sources

- MySQL, Microsoft SQL Server, PostgreSQL, mongoDB, JSON, XML, Hbase, HDF5

Python Resources

- Data files, code from text
  - http://github.com/wesm/pydata-book
- Python Package Index (PyPI)
  - https://pypi.python.org/pypi
- The Python Tutorial
  - https://docs.python.org/3/tutorial/
- Numpy, Scipy, IPython, matplotlib, pandas
  - http://www.scipy.org/docs.html
- Data Camp
  - https://www.datacamp.com/
- PyData
  - https://www.youtube.com/user/PyDataTV
- pyvideo
  - http://pyvideo.org/
- YouTube
  - http://www.youtube.com
- Communities
  - Google Groups (see text, pg. 12)
  - Conferences: PyCon, EuroPython, SciPy, EuroSciPy, PyData

Python Tools

- Python/IPython Interpreter
- Jupyter Notebooks
- spyder Integrated Development Environment
  - Comes with Anaconda distribution, many other options for IDEs or text editors

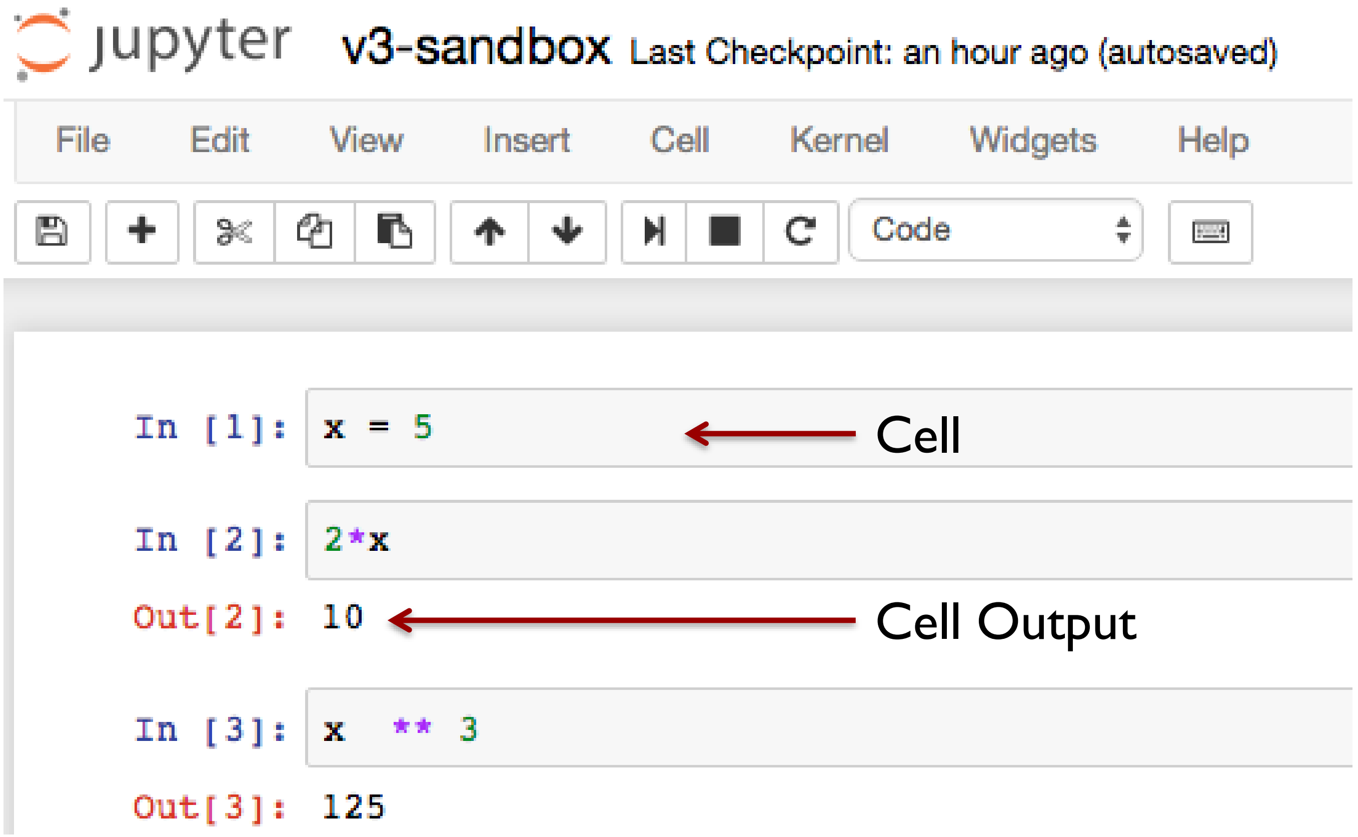
IPython

- Interactive Python
- Robust and productive environment for interactive and exploratory computing
  - Much more user-friendly than the standard Python interactive environment
- Additional features
  - Incorporated into Jupyter Notebook environment
  - GUI console with inline plotting, multiline editing, and syntax highlighting
  - An infrastructure for interactive parallel and distributed computing

Jupyter Notebook

- Convenient web-based environment
  - Leverages IPython functionality
  - Iterative code development
  - Cell-wise execution of code
  - Rich and mathematical text support
    - Markdown, LaTeX
  - Excellent communication tool
  - Easy to share (e.g., download as HTML or PDF)
- Compatible with recent version of Chrome (v13+), Firefox (v6+), or Safari (v5+)
  - Older versions are not supported
  - Internet explorer, Opera not supported

Jupyter Notebook



```
In [1]: x = 5

In [2]: 2*x
Out[2]: 10

In [3]: x  ** 3
Out[3]: 125
```

To execute cell in place: Ctrl + Enter

To execute cell and move to next cell: Shift + Enter

- Creates new cell if necessary

To execute and insert new cell: Option/Alt + Enter

For more keyboard shortcuts, see Help > Keyboard shortcuts

Jupyter Notebook Cell Types

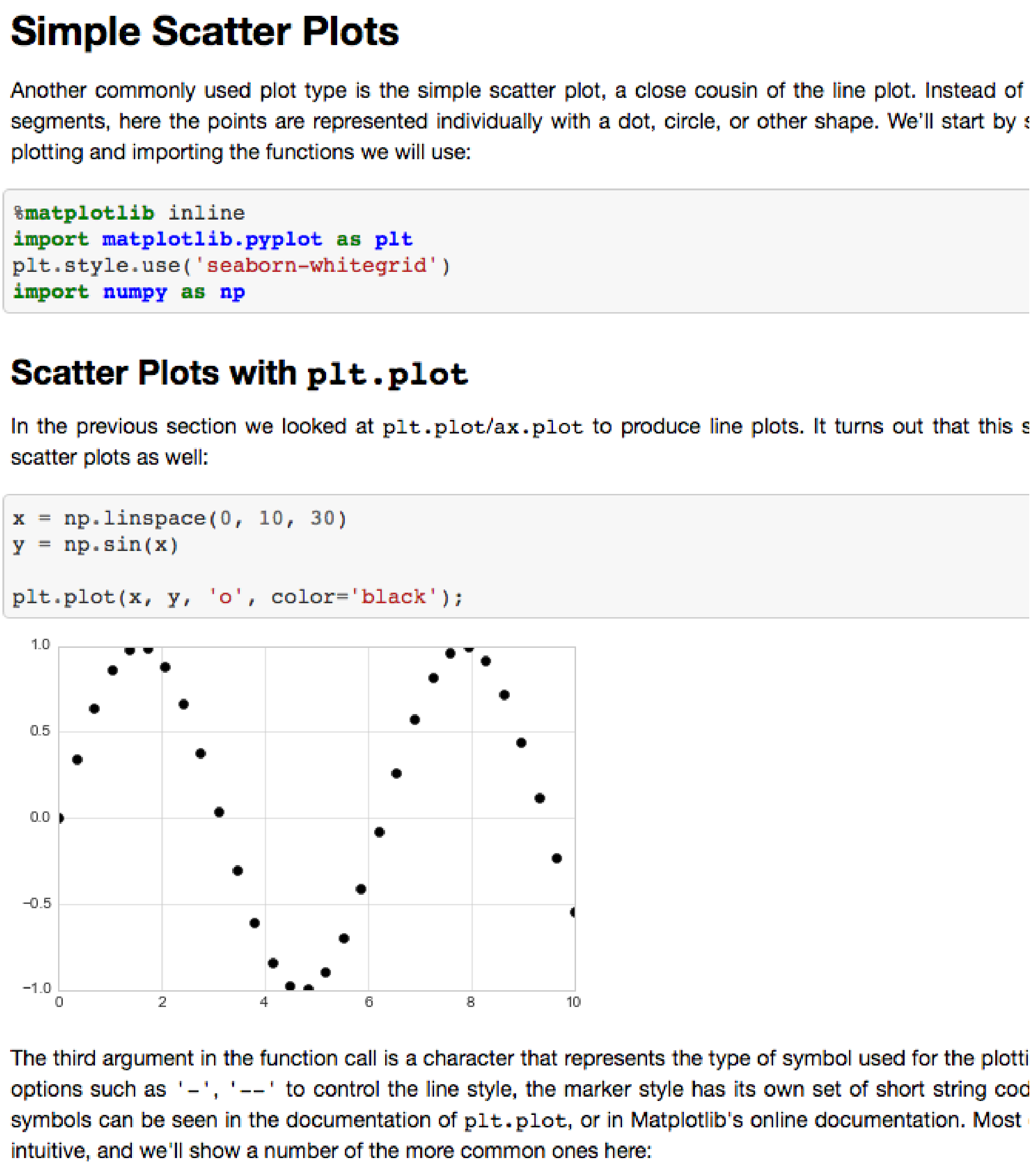
- Code cells
  - – Edit and execute cells inline, generates output (if applicable) as text, figures, HTML tables
  - – Syntax highlighting, tab completion, introspection
  - – Default type for inserted cells
- Markdown cells
  - – Rich text input, including HTML and LaTeX
    - Headings (begin with ##….#) or body text (as is)
  - – Double click to edit once executed

Jupyter Notebook Example

Good example of how to integrate code, markdown cells, and visualization into a single, shareable document

Extracted from GitHub page for Python Data Science Handbook:

- https://github.com/jakevdp/PythonDataScienceHandbook/tree/master/notebooks
- Section 04.02 – Simple Scatter Plots



Simple Scatter Plots

Another commonly used plot type is the simple scatter plot, a close cousin of the line plot. Instead of
segments, here the points are represented individually with a dot, circle, or other shape. We'll start by
plotting and importing the functions we will use:

```
%matplotlib inline
import matplotlib.pyplot as plt
plt.style.use('seaborn-whitegrid')
import numpy as np
```

Scatter Plots with `plt.plot`

In the previous section we looked at `plt.plot`/`ax.plot` to produce line plots. It turns out that this
scatter plots as well:

```
x = np.linspace(0, 10, 30)
y = np.sin(x)

plt.plot(x, y, 'o', color='black');
```

The third argument in the function call is a character that represents the type of symbol used for the plotti
options such as `'-'`, `'--'` to control the line style, the marker style has its own set of short string cod
symbols can be seen in the documentation of `plt.plot`, or in Matplotlib's online documentation. Most
intuitive, and we'll show a number of the more common ones here:

How to launch the Python tools

After proper installation (via Anaconda Distribution), you can run Python in several ways

- Command Line Initialization
  - Enter 'python', 'ipython', 'jupyter notebook', or 'spyder' into command line
    - Windows: Start > Run > Enter 'cmd'
    - Mac: Open Terminal application
- Anaconda Navigator – qtconsole (IPython), Jupyter Notebook, spyder IDE
  - Windows: Start > Program > Anaconda
  - Mac: Applications > Anaconda Navigator

How to run Python code

- Interactively
  - Write and execute code in real-time
    - Similar to R/MATLAB/Mathematica command window
  - Accessible via Python/IPython shell (from command prompt/terminal or within spyder IDE), or Jupyter Notebook environment
- Scripting
  - Write code via text editor, then execute code separately
    - Python scripts (text files with .py extension)
    - Choice of text editor is important (e.g., auto indent, syntax highlighting)
  - Execute code in several ways
    - In command prompt: **python script.py**
    - In interactive mode: **import script**
      - Entire script will execute as written, including any class definitions and functions, which will be loaded into the namespace for interactive use

BRIEF TOUR OF PYTHON TOOLS