

Lecture 2: Python Programming

Course: Biomedical Data Science

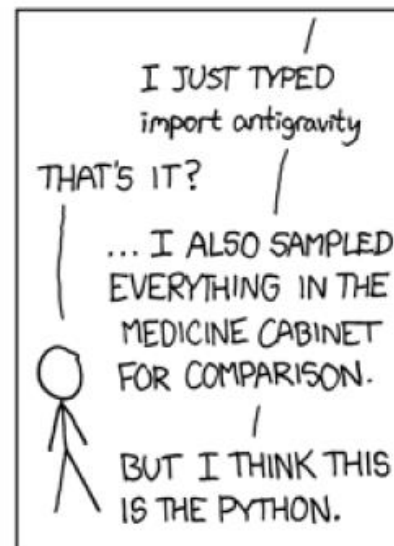
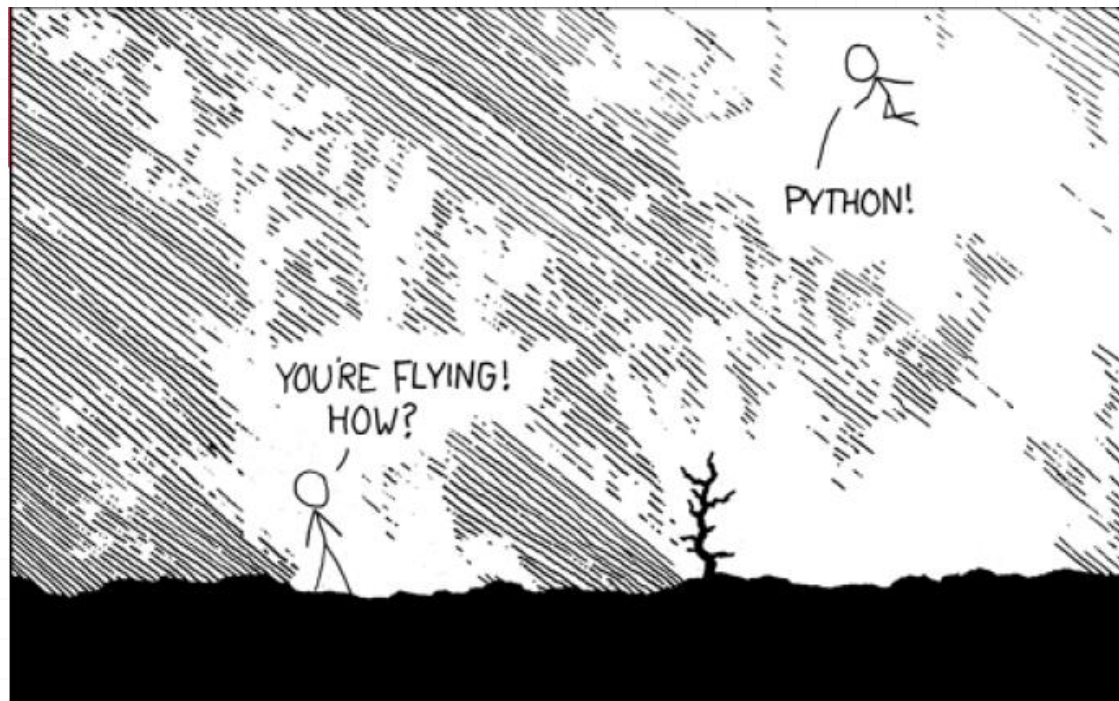
Parisa Rashidi

Fall 2018

Disclosure

The Slides are partially based on the following tutorial:
A Complete Tutorial to Learn Data Science with Python from Scratch
[Link](#)

and
PHY 546: Python for Scientific Computing
Instructor: [Michael Zingale](#)



Why Python?

- Open Source – free to install (compare to Matlab)
- Great online community
- Very easy to learn
- Can be used for both data science as well as the complete software stack.
- Needless to say, it still has few drawbacks too:
 - It is an interpreted language rather than compiled language – hence might take up more CPU time.
 - It is getting faster

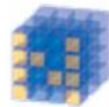
Why Python

- Read [this](#) Nature article on why you should pick up python



Scientific Python Stack

Most scientific libraries in python build on the Scientific Python stack:



NumPy
Base N-dimensional
array package



SciPy library
Fundamental library
for scientific
computing



Matplotlib
Comprehensive 2D
Plotting



IPython
Enhanced
Interactive Console



Sympy
Symbolic
mathematics



pandas
Data structures &
analysis

(scipy.org)

Object Oriented

- Python is an object-oriented language
- Think of an object as a container that holds both data and functions (methods) that know how to operate on that data.
 - Objects are built from a datatype called a class—you can create as many instances (objects) from a class as memory allows
 - Each object will have its own memory allocated
- Objects provide a convenient way to package up data
- Everything, even integers, etc. is an object
 - `1 + 2` will be interpreted as `(1).__add__(2)` in python

Your Computer: Installing Anaconda

- Two Options:
 - Standalone Python [Link](#)
 - Scientific Bundles such as Anaconda [Link](#)
- We suggest downloading [Anaconda](#), which contains most of the relevant libraries

Cloud

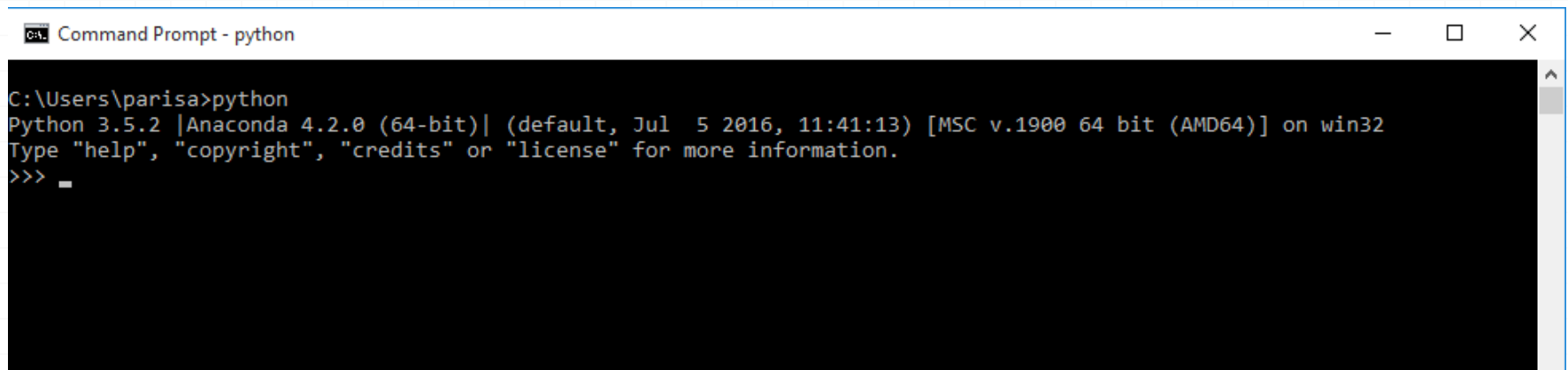
- Amazon AWS, fee-based, [Link](#)
- Google Colab, free, [Link](#)

Python: A start

- If you have python installed properly, you can start python simply by typing `python` at your command line prompt
 - The python shell will come up
 - Our hello world program is simply:
`print Hello, World!`
 - Or, in python 3.x (more on this later...):
`print(Hello World)`

Python Shell

- This is the standard (and most basic) interactive way to use python
 - Runs in a terminal window
 - Provides basic interactivity with the python language
- Simply type `python` at your command prompt
 - You can scroll back through the history using the arrows



```
Command Prompt - python

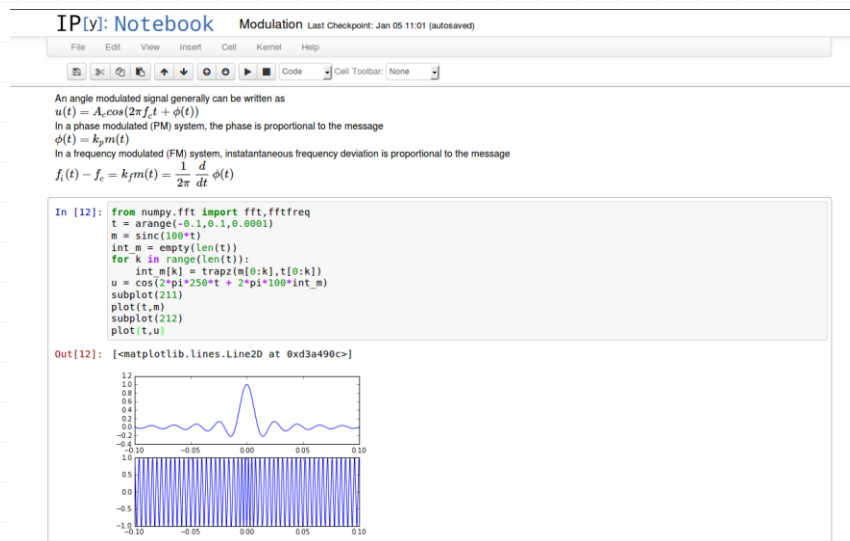
C:\Users\parisa>python
Python 3.5.2 |Anaconda 4.2.0 (64-bit)| (default, Jul 5 2016, 11:41:13) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> _
```

IPython Shell

- Type `ipython` at your prompt
- Like the standard shell, this runs in a terminal, but provides many conveniences (type `%quickref` to see an overview)
 - Scrollback history preserved between sessions
 - Build-in help with `?`
 - `function-name?`
 - `object?`
 - Magics (`%lsmagic` lists all the magic functions)
 - `%run script`: runs a script
 - `%timeit`: times a command
 - Lots of magics to deal with command history (including `%history`)
 - Tab completion
 - Run system commands (prefix with a `!`)

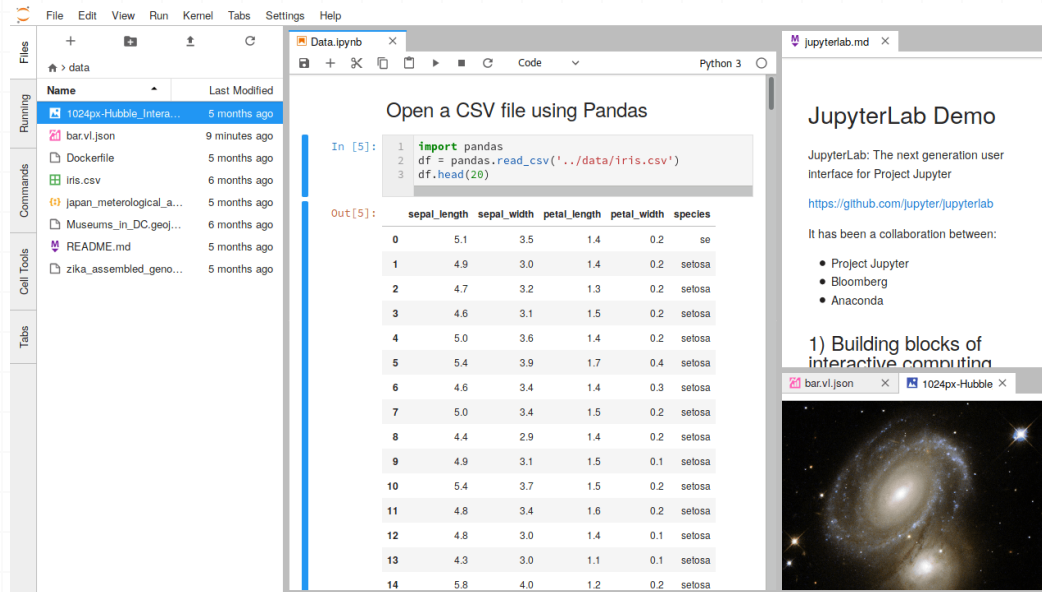
Jupyter Notebook

- A web-based environment that combines code and output, plots, plain text/stylized headings, LaTeX (later versions), ...
 - Notebooks can be saved and shared
 - Viewable on the web via: <http://nbviewer.ipython.org/>
 - Provides a complete view of your entire workflow



Jupyter Lab

- The next-generation user interface for Project Jupyter
- A more flexible and powerful user interface
- Stable version was released in February 2018



The screenshot displays the JupyterLab user interface. On the left is a file explorer sidebar showing a directory structure with files like '1024px-Hubble_Inter...', 'bar.vl.json', 'Dockerfile', 'iris.csv', 'japan_meterological_a...', 'Museums_in_DC.geoj...', 'README.md', and 'zika_assembled_geno...'. The main area is divided into two panes. The left pane, titled 'Data.ipynb', shows a code cell with the following code:

```
In [5]: 1 import pandas
2 df = pandas.read_csv('../data/iris.csv')
3 df.head(20)
```

The output of the code cell is a table with 20 rows and 5 columns: 'sepal_length', 'sepal_width', 'petal_length', 'petal_width', and 'species'. The data is as follows:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	se
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
5	5.4	3.9	1.7	0.4	setosa
6	4.6	3.4	1.4	0.3	setosa
7	5.0	3.4	1.5	0.2	setosa
8	4.4	2.9	1.4	0.2	setosa
9	4.9	3.1	1.5	0.1	setosa
10	5.4	3.7	1.5	0.2	setosa
11	4.8	3.4	1.6	0.2	setosa
12	4.8	3.0	1.4	0.1	setosa
13	4.3	3.0	1.1	0.1	setosa
14	5.8	4.0	1.2	0.2	setosa

The right pane, titled 'JupyterLab Demo', contains text describing JupyterLab as the next generation user interface for Project Jupyter, with a link to the GitHub repository. It also lists collaborators: Project Jupyter, Bloomberg, and Anaconda. Below this, it mentions '1) Building blocks of interactive computing' and shows a small image of a galaxy.

Python 2.7, 3.5, 3.6, 3.7

- The choice of python version depends on your application
- 3.* mostly backward compatible
- 2.7: some libraries still work with 2.7
- 3.4/3.5/3.6/3.7: cleaner, faster, the future
- 3.7 released in June 2018

Python 2 vs. 3

- See <https://wiki.python.org/moin/Python2orPython3>
- Mostly about cleaning up the language and making things consistent
 - e.g. `print` is a statement in python 2.x but a function in 3.x
- Some trivial differences
- It's possible to write code that works with both python 2 and 3—often we will do so by importing from `__future__`

Using Python

- Different ways to use Python
 - Terminal / Shell based
 - IDLE, installed with standalone Python
 - Jupyter notebook
 - Jupyter Lab (to replace Jupyter Notebook)
 - Spyder

Python Scripts

- Scripts are a non-interactive means of writing python code
 - `filename.py`
- This is also the way that you write python modules that can be `import`-ed into other python code (more on that later...)

Demo

- Demo of Jupyter notebook will be shown in the class

Some Notes on Notebook

- You can start Jupyter notebook by writing “jupyter notebook” on your terminal / cmd, depending on the OS you are working on
- You can name a iPython notebook by simply clicking on the name
- The interface shows In [*] for inputs and Out[*] for output.
- You can execute a code by pressing “Shift + Enter” or “ALT + Enter”, if you want to insert an additional row after.

Data Structures

Data Structures

- Lists
- Strings
- Tuples
- Dictionary

Lists

- Lists are one of the most versatile data structure in Python.
- A list can simply be defined by writing a list of comma separated values in square brackets.
- Lists might contain items of different types, but usually the items all have the same type.
- Python lists are mutable and individual elements of a list can be changed.

Strings

- Strings can simply be defined by use of single ('), double (") or triple ("'') commas.
- Strings enclosed in tripe quotes ("'') can span over multiple lines and are used frequently in docstrings (Python's way of documenting functions).
- \ is used as an escape character.
- Please note that Python strings are immutable, so you can not change part of strings.

Tuples

- A tuple is represented by a number of values separated by commas.
- Tuples are immutable and the output is surrounded by parentheses so that nested tuples are processed correctly.
- Additionally, even though tuples are immutable, they can hold mutable data if needed.
- Since Tuples are immutable and can not change, they are faster in processing as compared to lists. Hence, if your list is unlikely to change, you should use tuples, instead of lists.

Dictionary

- Dictionary is an unordered set of key: value pairs, with the requirement that the keys are unique (within one dictionary).
- A pair of braces creates an empty dictionary: {}.

Loops

- For loops have a simple structure

```
for i in [Python Iterable]:  
    expression(i)
```

- Here “Python Iterable” can be a list, tuple or other advanced data structures

Conditional Statement

- The most commonly used construct is if-else, with following syntax:

```
if [condition]:  
    __execution if true__  
else:  
    __execution if false__
```


Question

- Now that you are familiar with Python fundamentals, let's take a step further. What if you have to perform the following tasks:
 - Multiply 2 matrices
 - Find the root of a quadratic equation
 - Plot bar charts and histograms
 - Make statistical models
 - Access web-pages

Libraries

- If you try to write code from scratch, its going to be a nightmare and you won't stay on Python for more than 2 days! But lets not worry about that. Thankfully, there are many libraries with predefined which we can directly import into our code and make our life easy. e,.g.

```
math.factorial(N)
```

Libraries

- The first step is obviously to learn to import them into our environment. There are several ways of doing so in Python:

```
import math as m
```

```
from math import *
```

Some Libraries

- **NumPy** stands for Numerical Python. The most powerful feature of NumPy is n-dimensional array. This library also contains basic linear algebra functions, Fourier transforms, advanced random number capabilities and tools for integration with other low level languages like Fortran, C and C++
- **SciPy** stands for Scientific Python. SciPy is built on NumPy. It is one of the most useful library for variety of high level science and engineering modules like discrete Fourier transform, Linear Algebra, Optimization and Sparse matrices.
- **Matplotlib** for plotting vast variety of graphs, starting from histograms to line plots to heat plots.. You can use Pylab feature in ipython notebook (ipython notebook -pylab = inline) to use these plotting features inline. If you ignore the inline option, then pylab converts ipython environment to an environment, very similar to Matlab. You can also use Latex commands to add math to your plot.

Some Libraries

- **Pandas** for structured data operations and manipulations. It is extensively used for data munging and preparation. Pandas were added relatively recently to Python and have been instrumental in boosting Python's usage in data scientist community.
- **Scikit Learn** for machine learning. Built on NumPy, SciPy and matplotlib, this library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction.
- **Statsmodels** for statistical modeling. Statsmodels is a Python module that allows users to explore data, estimate statistical models, and perform statistical tests. An extensive list of descriptive statistics, statistical tests, plotting functions, and result statistics are available for different types of data and each estimator.

Some Libraries

- **Seaborn** for statistical data visualization. Seaborn is a library for making attractive and informative statistical graphics in Python. It is based on matplotlib. Seaborn aims to make visualization a central part of exploring and understanding data.
- **Bokeh** for creating interactive plots, dashboards and data applications on modern web-browsers. It empowers the user to generate elegant and concise graphics in the style of D3.js. Moreover, it has the capability of high-performance interactivity over very large or streaming datasets.
- **Blaze** for extending the capability of Numpy and Pandas to distributed and streaming datasets. It can be used to access data from a multitude of sources including Bcolz, MongoDB, SQLAlchemy, Apache Spark, PyTables, etc. Together with Bokeh, Blaze can act as a very powerful tool for creating effective visualizations and dashboards on huge chunks of data.

Some Libraries

- **Scrapy** for web crawling. It is a very useful framework for getting specific patterns of data. It has the capability to start at a website home url and then dig through web-pages within the website to gather information.
- **SymPy** for symbolic computation. It has wide-ranging capabilities from basic symbolic arithmetic to calculus, algebra, discrete mathematics and quantum physics. Another useful feature is the capability of formatting the result of the computations as LaTeX code.
- **Requests** for accessing the web. It works similar to the the standard python library urllib2 but is much easier to code. You will find subtle differences with urllib2 but for beginners, Requests might be more convenient.

Some Libraries

- **os** for Operating system and file operations
- **networkx** and **igraph** for graph based data manipulations
- **regular expressions** for finding patterns in text data
- **BeautifulSoup** for scrapping web. It is inferior to Scrapy as it will extract information from just a single webpage in a run.

Pandas



Let's get Started with Pandas!

- 2 key data structures in Pandas:
 - Series
 - DataFrames
- The data sets are first read into these structures and then various operations can be applied very easily.

Series

- Series can be understood as a 1 dimensional labelled indexed array.
- You can access individual elements of this series through these labels.

Data Frame

- A dataframe is similar to Excel workbook – you have column names referring to columns and you have rows, which can be accessed with use of row numbers. The essential difference being that column names and row numbers are known as column and row index, in case of dataframes.

Practice on your own

- Academy python track: a step-by-step tutorial through the basics of the language
- Project Euler: a set of increasingly complex programming tasks to try out with python