Setup

Introduction to Data Science with Python

Course Information

- Book we are using
  - Python Data Science Handbook

  https://tanthiamhuat.files.wordpress.com/2018/04/pythondatasciencehandbook.pdf
- Learn to use online references
- Python is extensive, much more than me ☺
  - There may be questions I cannot answer (or remember)
  - Fortunately there are great resources on the Internet
- In General - Learn the basic language/syntax, and learn how to use Python resources to accomplish your goals

What is Python?

- Started as an easy to learn programming language but adopted for Data Science by the "Machine Learning" crowd
- Introduction of Numpy, Pandas and other modules has brought main stream data science to Python
- Often contrasted with R these days for data science (you might end up using both)
- "Slow" compared to lower level languages, but easy to code in…

What is Python?

- Open Source "High Level" programming language
- Looks like Pseudo-Code…
- Object Oriented, has a "module" system for adding extensions
- Started in 1989/1990
  - 2.0 released in 2000
  - 3.0 released in 2008
- 3.0 broke backwards compatibility, but removed some duplication and added structural improvements

Why Python?

- Big Debate here…why should we use Python for data science?
- From your book:
  - "Python as Glue"
  - "The Two Language Problem"
- Bottom Line: Python works as a data science tool, and also as a general programming language.
- Often compared to R:
  - Less focused on statistics/data science, but catching up quickly.

Which Python?

- 2.0 versus 3.0
  - We will use 3.6.xx
- These are often not compatible, some basic functions differ (like “print”, integer division)
- Most modules have been migrated up to version 3 so you should as well
  - 2.7 release is supposed to be the last major release for version 2
- Overall, the learning experience will be very similar.

Anaconda Python

- We will be using the Anaconda Python Distribution, Python 3.6.XX
  - This has everything we need, is open source, and is easy to install.
  - This includes…
  - The Python kernel
  - IPython (enhanced interpreter? More on this later)
  - Spyder (IDE)
  - Scientific Stack of libraries (packages?)
- We could build a Python stack from scratch, compiling all libraries we need…this is not trivial.

Anaconda Installation

Download Installation Files from here:

https://www.continuum.io/downloads

-We are using 3.6.x

-Cross Platform

-Free basic setup, charges for “Add-Ons”

Does anyone not have this done? If not, begin the download now

Python Data Science Libraries

- IPython
  - Enhanced interpreter
- NumPy
  - Matlab type stuff
  - Ndarray object
- Matplotlib
  - Plotting in 2D originally (now some 3D)
- Pandas
  - Dataframe class, Time series analysis
- Stat Models
  - Basic Regression
- Scikit Learn
  - Machine learning classification

Spyder IDE

- We will use the IDE called Spyder in Class
- This comes with the Anaconda distribution.
- Let’s open up this IDE and take a quick tour.

Code in the slides

- You will receive session code similar to what is in the slides so you can follow along in the demonstrations
  - Or, if you prefer, just watch while I do it
- Note that some code you can run as is from the slide, but others will be snippets, of which the larger part is in the session code.
- I suggest you type/retype the code (time permitting) as we go through the demonstrations to help with your Python "memory"
- Just let me know if we are moving too quickly or you have other questions

IPython: Enhanced Interpreter

- Spyder uses an enhanced interpreter called IPython
- "Execute – Explore"

  Supports iterative nature of data science, different from typical programming workflows.
- Tab Complete
- Introspection

```
# Ipython console within Spyder
x1 = 5
x2 = 8

x <tab>

?x1
```

Let's Get Started

- Add comments to code to add clarity
- Be concise, comments add clutter

```
# Use a hash mark for a comment

# Use comments to highlight important or unintuitive decisions

a_piece_of_code = "is a string"

print("Output " + a_piece_of_code)


> Output is a string
```

Multiline Comments & Help

- Multiline comments start and end with three quotes (single or double – same kind)
- help() or ? Or Object Inspector will get you the details about methods

  Use quotes to get help for an object not yet loaded

```
# Single line Comment
""" This is a
Multiline Comment, also known as a docstring
"""
help(str)
var = "hello"
print(var.capitalize())
help(str.capitalize)
str.capitalize?
```

Magic Commands

- IPython supports a number of built-in commands to do a variety of tasks
  - Some of these you will want to use as part of your scripts
  - They aren't technically Python code, but you can embed them in your scripts to perform some special functions that would be more work otherwise
  - Some examples
    - **%cd** – change directory
    - **%debug** – activate the interactive debugger
    - **%history** – print command history
    - **%pwd** – return current working directory
    - **%time**, **%timeit** – Time the execution of Python statement or expression

Python Basics

- Python is high level
- Python supports object oriented Programming
  - Everything is an object, including numbers and operators
  - Every object has a type
- Python is focused on readability
  - Looks like pseudo-code

Modules

- Modules allow us to extend Python with new functionality
  - Math
  - Statistics
  - Date and Time Functions
  - Graphics
  - Machine Learning, etc.

- Modules can also be user defined as simply scripts ending in .py
  - More on this later.

Modules

- We import modules with the import statement

  import *module* *(import module)*

  import *module* as *alias* *(give module short alias name)*

  from *module* import * *(Load all module objects into local namespace…don't do this)*

```
# Import module
import keyword as kw
kw.kwlist
from keyword import kwlist
```

Working Directory

- We can change the working directory through Spyder, or by submitting code with os module

```
import os
# Get Working Directory
os.getcwd()
# Set Working Directory
os.chdir(r“C:\Project1”)

help(os)
```

Lab 2 – Anaconda Install (30 min)

- By now, you should have downloaded the Anaconda installation
- Install the software if you haven’t done so already

```
Test your installation with the next exercise
```

Exercise – 5 minutes

- Validate your lab download by spending a few minutes exploring Python and Spyder
  - Create a variable
  - View it in the variable explorer
  - Run a command in the console
  - Get help for an object
  - Explore menus and toolbar

  - In the next lesson we start coding…