DSC 201: Data Analysis & Visualization

Python and Notebooks

Dr. David Koop
http://www.cis.umassd.edu/~dkoop/dsc201
Course Material

• Course Website
  - http://www.cis.umassd.edu/~dkoop/dsc201-2018fa
  - **All** material will be posted there
  - myCourses for turning in assignments

• Textbook: *Python for Data Analysis*
  by Wes McKinney, 2nd ed., 2017
  - Good reference for data science topics in Python
  - McKinney created the Pandas package
Course Material

- Anaconda Python Distribution (https://www.continuum.io/downloads): makes installing python and python packages easier
- Jupyter Notebook: Web-based interface for interactively writing and executing Python code
- JupyterLab: An updated web-based interface that includes the notebook and other cool features
- JupyterHub: Access everything through a server
Course Material

• Pandas:
  - Python library for data analysis
  - Many operations available
  - Efficient

• Tableau:
  - Desktop (or web) application
  - Create visualizations quickly

• Other Visualization Tools:
  - Python libraries: Matplotlib, Altair, Bokeh, folium
  - Don't have to move between applications
Grading

• Assignments (5): 40%
• Quizzes: 2 in-class: 7.5% each
• Midterm: 17.5%
• Final: 22.5%
• Class Participation: 5%
• Late Policy
Important Dates

• **Check these now!**
• Quiz 1: October 2 (in class)
• Midterm: October 23 in class
• Quiz 2: November 20 (in class)
• Final Exam: December 12, 11:30am-2:30pm
• Quizzes and exams may not be rescheduled and can only be made up in case of a documented emergency.
Accommodation Policy

• Please contact me at the **beginning** of the semester and provide the appropriate paperwork from the Center for Access and Success.

• Please update me if anything changes during the semester.

• Center for Access and Success: Pine Dale Hall Room 7136, x8711, access_success@umassd.edu
Academic Honesty

• Do not cheat!

• You will receive a zero for any assignment/exam/etc. where cheating has occurred. Repeat offenders will fail the course.

• You may discuss problems and approaches with other students

• You may not copy or transcribe code from another source
Course Registration Reminder

• Add/drop deadline is tomorrow
• Make sure you are signed up for the course
Chicago Food Inspections Exploration

- Based on David Beazley's PyData Chicago talk
- YouTube video: https://www.youtube.com/watch?v=j6VSAasKAj98
- Our in-class exploration:
  - Don't focus on the syntax
  - Focus on:
    - What is information is available
    - Questions are interesting about this dataset
    - How to decide on good follow-up questions
    - What the computations mean
Chicago Food Inspections Exploration
Chicago Food Inspections

• Do not focus on code syntax!
• What questions did we analyze?
Chicago Food Inspections

• Do not focus on code syntax!
• What questions did we analyze?
• Sample answers:
  - How many inspections are there?
  - What types of ratings do facilities get?
  - Which restaurant chains fail inspections most?
  - What address has the most inspection failures?
  - What is unique about that address?
  - How often is a facility checked?
  - Which violation is most common?
  - Where are failures most common?
Chicago Food Inspections

• What *tasks* did we do?
Chicago Food Inspections

• What **tasks** did we do?

• Sample answers:
  - Reading data
  - Organizing data
  - Extracting unique values
  - Counting occurrences
  - Cleaning names
  - Searching data
  - Transforming data (string -> list)
  - Visualizing data
Python

- Started in December 1989 by Guido van Rossum
- “Python has surpassed Java as the top language used to introduce U.S. students to programming…” (ComputerWorld, 2014)
- Python and R are the two top languages for data science
- High-level, interpreted language
- Supports multiple paradigms (OOP, procedural, imperative)
- Help programmers write **readable** code
- Use less code to do more
- Lots of libraries for python
  - Designed to be extensible
  - Easy to wrap code from other languages like C/C++
- Open-source with a large, passionate community
Python Compared to Java

• Dynamic Typing
  - A variable does not have a fixed type
  - Example: `a = 1; a = “abc”`

• Indentation
  - Braces define blocks in Java, good style is to indent but not required
  - Indentation is critical in Python
  - Example:
    ```
    z = 20
    if x > 0:
      if y > 0:
        z = 100
      else:
        z = 10
    ```
Learning Python Resources

- https://software-carpentry.org/lessons.html
- http://www.pythontutor.com
- http://www.python-course.eu
- http://thepythonguru.com
- https://en.wikibooks.org/wiki/A_Beginner%27s_Python_Tutorial
- https://learnpythonthehardway.org
- learnpython.org
Jupyter Notebook

- Display rich representations and text
- Uses Web technology
- Cell-based
- Built-in editor
- GitHub displays notebooks
Jupyter Notebook

- An interactive programming environment
- Runs in your web browser
- Displays results (even interactive maps) inline
- Originally designed for Python
- Supports other languages, too
- You decide how to divide code into executable cells
- Shift+Enter to execute a cell
Using Python & Jupyter Notebook at UMassD

- [https://rps.cscvr.umassd.edu:8000/](https://rps.cscvr.umassd.edu:8000/)
- Just log in and everything should be setup
- Advanced:
  - Can add your own conda environments in your user directory
Using Python & Jupyter Notebook Locally

- www.anaconda.com/download/
- Anaconda has Jupyter Notebook
- Use Python 3.6 version (not 2.7)
- Anaconda Navigator
  - GUI application for managing Python environment
  - Can install packages
  - Can start Jupyter Notebook
- Can also use the shell to do this:
  - $ jupyter-notebook
  - $ conda install <pkg_name>
Jupyter Notebook Tips

• Starts with a directory view

• Create new notebooks using "New" menu on the right
  - Originally has name "Untitled"
  - Click on "Untitled" to change the name (do this!)

• Save a notebook using the command under the File menu

• Shutting down the notebook requires quitting the kernel
  - Web browser is interface to display code and results
  - Kernel actually runs the code: usually see messages in a console/terminal window
  - Ctrl+C in this window ask you if you want to shut down the kernel
Jupyter Notebook

- Open a notebook by going back to the main screen and clicking on it like you would in a desktop view
- Past results are displayed—does not mean they are loaded in memory
- Use "Run All" or "Run All Above" to re-execute past work
  - If you shut down the kernel, all of the data and variables you defined need to be redefined (so you need to re-run all)
  - **Watch Out—Order Matters:** If you went back and re-executed cells in a different order than they are shown, doing "Run All" may not produce the same results!
- Edit mode (green) versus Command mode (blue == **Be Careful**)
Jupyter Notebook

• Can write code or plain text (can be styled Markdown)
  - Choose the type of cell using the dropdown menu

• Cells break up your code, but all data is global
  - Defining a variable $a$ in one cell means that variable is accessible in any other cell
  - This includes cells above the cell $a$ was defined in!

• Remember Shift+Enter to execute

• Enter just adds a new line

• Use ?<function_name> for help

• Use Tab for auto-complete or suggestions

• Tab also indents, and Shift+Tab unindents
Jupyter Notebook Documentation

- Notebook Quick Start Guide
- Notebook Basics
Open a CSV file using Pandas

```python
In [5]:
1. import pandas
2. df = pandas.read_csv('data/iris.csv')
3. df.head(10)
```

Output:
```
sepal_length  sepal_width  petal_length  petal_width  species
0            5.1          3.5          1.4          0.2     setosa
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     versicolor
6            4.6          3.4          1.4          0.3     setosa
7            5.0          3.4          1.5          0.2     versicolor
8            4.4          3.0          1.4          0.2     setosa
9            4.9          3.1          1.5          0.1     versicolor
```

JupyterLab Demo

JupyterLab: The next generation user interface for Project Jupyter

https://github.com/jupyter/jupyterlab

It has been a collaboration between:
- Project Jupyter
- Rhomborg
- Anaconda

1) Building blocks of interactive computing

[JupyterLab Documentation]
JupyterLab

- Notebooks work almost identically as the "classic" notebook
- More than just notebooks:
  - Text editor
  - Console
  - Custom components (Many extensions)
- Arrange multiple documents and views
- Documentation
- Tutorial:
  - https://www.youtube.com/watch?v=Gzun8PpyBCo
Using JupyterLab

• Installed on rps
  - https://rps.cscvr.umassd.edu:8000/
  - Click the JupyterLab button after logging in
• Can install locally using conda
  - conda install -c conda-forge jupyterlab
Advanced: Python 2 and 3

- [https://docs.python.org/3/whatsnew/3.0.html](https://docs.python.org/3/whatsnew/3.0.html)

- Key Differences:
  - print as a function: print "Hello" vs. print("Hello")
  - Views and iterators instead of lists
  - Integer division: 5/2 = 2.5, 5//2 = 2
  - Unicode as standard
  - String formatting:
    - Py2: "Hello %s. You are %d years old" % (name, age)
    - Py3: "Hello {}. You are {} years old".format(name, age)
    - Py3.6: f"Hello {name}. You are {age} years old"