Software Prototyping for Data Science and Information Management

Info 206

Niall Keleher

24 August 2017

Today's Outline

1. Introductions
2. Course Overview
3. Easy Quiz!
4. Setup - command line, bash, git
5. Collaborative exercise: exploring Github repositories

Introductions


HELLO
my name is
NIALL

About me

- What to call me: Niall ("Neil")
- Ph.D. Student (4th year) in UC Berkeley School of Information
- My research focuses on applying methodologies from data science and machine learning to the measurement of social and economic indicators
- I have worked on survey data collection and field experiments (primarily in Africa and Asia) since 2006 with the World Bank and Innovations for Poverty Action.

What I've been up to this summer

Course Overview

Course website and syllabus

http://nkeleher.com/info206.html

Office hours:

Thursdays 1:00 - 3:00 PM

6A South Hall

Course Objectives

At the end of the course, a successful student should be able to:

- work collaboratively using source control
- understand and operationalize core Python objects
- create user-written functions in Python
- work with classes in Python
- clearly explain and execute good practices in software prototyping

What this course will provide you

- A foundation for computer programming to use in future classes and work.
- A familiarity with tools of software development, data science, and collaborative programming.
- A thorough introduction to the use of core methods of programming languages with a heavy focus on Python.

What this course will not provide

- A computer science or algorithms course.
- Everything you need to become a Data Scientist.
- A guaranteed job.
- Spam [*]

[*] See Lutz for Monty Python humor.

Course Structure

- 'Flipped classroom' model
- Materials and videos produced specifically for UC Berkeley iSchool students
    - Originally developed for MIDS students.
    - First year of delivering online materials to MIMS students
- Intensive reading and practical application to compliment online resources
- Online video resources found through: https://learn.datascience.berkeley.edu
- Much of our work will be done through Github [more on this later in the meeting]

https://learn.datascience.berkeley.edu

- Developed by Professor Paul Laskowski for W18/INFO 200
- In this course, we will concentrate on materials that cover core programming lessons in Python[1]
- Topics in applied data science will not be covered (available in subsequent courses)
- Additional materials shared through Github class site[2]

[1] Instructors in videos include: Dr. Paul Laskowski, Kay Ashaolu, and Bill Chambers.

[2] Additional materials were initially developed by Professor Laskowski, Gunnar, Kleeman, and Chris Llop.

Paul


Niall

Course Structure

- The course is divided by in-class and out-of-class activities.
- Watch instructional videos as well as read the relevant in advance of each in-class session.
- Classroom sessions are intended to review key information covered in the videos and readings, introduce new topics (e.g. algorithms and applications), and provide teams with time to work on their group projects.
- Each session is (loosely) structured as follows:
    - Quiz on videos and reading for the day's session (10 minutes)
    - In-class review plus applications (40 minutes)
    - Collaboratiive assignments and group project activities (30 minutes)

Main Texts for Course

- Mark Lutz. *Learning Python: Powerful Object-Oriented Programming*, 5th Edition, O'Reilly Media, Inc., 2013.
    - An online version is freely available at UCB through Safari publications: http://proquest.safaribooksonline.com/9780596513986
    - Code from the book is available through the book website: http://learning-python.com/about-lp5e.html
- Kent D. Lee and Steve Hubbard. *Data structures and algorithms with Python*, Springer, 2015.
    - A pdf and ebook version is freely available at UCB through Springer: https://link.springer.com/book/10.1007%2F978-3-319-13072-9
    - Code from the book is available through: http://knuth.luther.edu/~leekent/CS2Plus
- Scott Chacon and Ben Straub. *Pro Git*, 2nd Edition, Apress, 2014.
    - An online version is freely available at: https://git-scm.com/book/en/v2.

Course Schedule

Course Schedule

08/24 Course overview, Command Line, Bash, and Git

08/29 Git Review, Teams, Python overview, Installing Python

08/31 Types, Polymorphism, Numerics, and Strings

09/05 Sequences, Lists, and Dictionaries

09/07 Statements and Syntax, Part 1

09/12 Statements and Syntax, Part 2

09/14 Functions

09/19 Recursion and List Comprehension

09/21 Modules, Part 1

09/26 Modules, Part 2

09/28 Object-Oriented Programming & Classes, Part 1

10/03 Object-Oriented Programming & Classes, Part 2

10/05 Complexity

10/10 Exceptions & Testing, Part 1

10/12 Exceptions & Testing, Part 2

10/17 Project Presentations

Course Expectations

- Treat our class meetings as if they were work meetings.
- Attendance is expected.
- Arrive on time.
- Respect the time and space of your colleagues.
- Learn through experience. Your hard work will pay off.

Course Grading

- **10%** of your grade will be determined by your **attendance** and participation in class. Generally, ask questions and answer them
- **20%** of your grade will be determined by **quizzes**, which will occur at the beginning of every in-class meeting
- **10%** of your grade will be determined by **individual assignments** *
- **10%** of your grade will be determined by **group and team assignments**
- **50%** of your grade will be determined by a **group project** that culminates in a presentation at the end of the course

[*] Assignments should be submitted by the due date indicate. In general, assignments are completed during class meetings - thus assignments are due at 12:30 PM. If you need additional time to complete the assignment, you can submit the assignment within 24 hours of the due date (e.g. by 12:30 PM on the day following a class meeting) with a 25% penalty. Assignments submitted between 24 and 48 hours of the due date will be docked a 50% penalty. Assigments submitted more than 48 hours after the due date will be docked a 75% penalty.

Python

Your mission: to become an adept programmer in Python.

Why Python?

- Object-oriented and Functional - good for learning core programming paradigms
- Powerful programming language - 'smart' memory allocation and quick to type code
- Dynamically typed - no need to declare object types
- Portability - works across platforms
- Extensive libraries and support
- Relatively accessible to learn - focus on readability, reusability. and accessibility
- Good entry point for working with other languages - Java, C++, Ruby, R
- It's free!

Requisites

Computer: mac/windows/linux

Software: text editor, terminal emulator, git, Python

Brain: skepticism, curiosity, organization, communication

Logistics

- Class listserve (Google group)
- ISVC accounts
- Office hours
- Project groups [more on this next week]

Today's Quiz: https://goo.gl/forms/n9HgEaHMdObvAlAl2

Setup - command line, bash, git

Terminal emulator and Command Line

- 'Working in the shell'
- Unix machines (Mac/Linux) - typically built in terminal
- Windows - Powershell, GitBash, Anaconda Prompt

First video

The Command Line

Key commands

- pwd
- hostname
- mkdir
- cd
- ls
- rmdir
- cp
- mv
- head
- tail
- less
- cat
- find
- Which
- man
- env
- echo
- exit

Bash

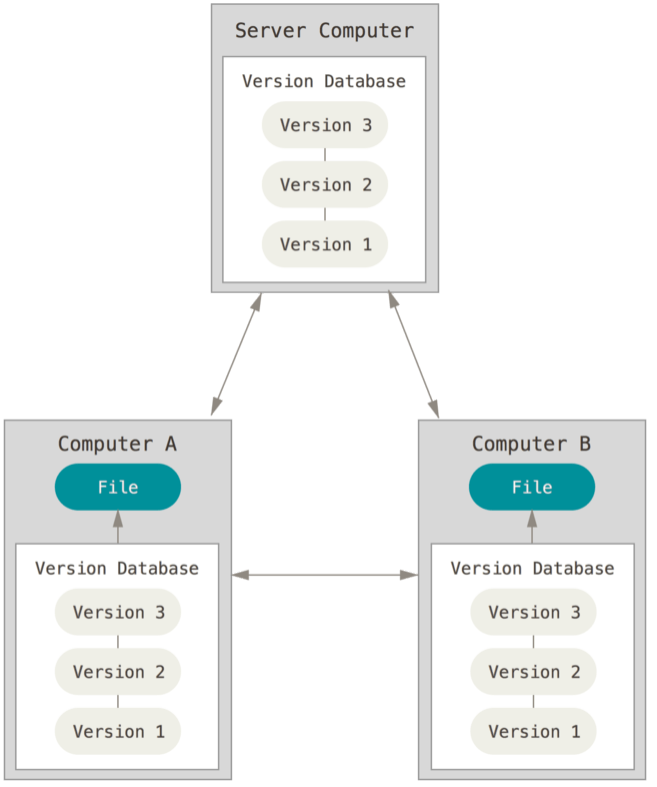
- Can be used in interactive shell or as scripting language

Command line tutorial

https://learnpythonthehardway.org/python3/appendixa.html

Source Control

- A system for organizing different versions of a project under development
    - It could be a document, a picture file, almost anything.
    - In our case, we want to keep track of different versions of the Python code we're writing.
- Version Control Systems are ubiquitous in the software development industry



Source Control


git

Installing Git

- Most Mac and Linux users should have git preinstalled.
- Windows users are less likely to have Git preinstalled
    - download Github Desktop: https://desktop.github.com/
- In the terminal emulator, run:

```
which git # Shows where git binary/executable is stored

git --help # Help information for git command
```

Git demo


GitHub

Github

- Create a Github account if you do not already have one.
  - https://github.com/ >> Sign-up
- On your machine, follow the instructions here:
  - https://help.github.com/articles/connecting-to-github-with-ssh/

Join class Github organization

https://github.com/INFO206-Fall2017

- Send me an email (nkeleher@berkeley.edu) with your Github account ID

Readings

- Command Line tutorial https://learnpythonthehardway.org/python3/appendixa.html
- Chacon & Straub -- Chapter 1 & 2 - https://git-scm.com/book/en/v2
- Lutz Chapter 1: A Python Q\&A Session - http://cdn.oreillystatic.com/oreilly/booksamplers/9781449355739_sampler.pdf

Additional Resources

- http://rogerdudler.github.io/git-guide/
- https://guides.github.com
- http://ohshitgit.com/

Collaborative exercise: Exploring Github repository

Work in groups of 2-3

- Explore the following public Github repository that uses python
- With your group members, discuss what you see in the repositories.
  - How are files organized?
  - Who contributes to the repository?
  - Does the repositories have any branches? Has it been forked?
  - Are the repositories well documented?
  - Are there any commonalities across multiple repositories?
- Write up your notes as a group and send to the class email.
- Repos:
  - https://github.com/fogleman/Minecraft
  - https://github.com/facebook/pyaib
  - https://github.com/cherrypy/cherrypy

End of Meeting #1

For next meeting

- Videos:

    1. Introducing Python (5 mins)
    2. Programming Language Characteristics (10 mins)
    3. Starting Python (4 mins) [optional]
    4. The Jupyter Notebook Page (14 mins) [optional]

- Readings:

    - Lutz Chapter 2: How Python Runs Programs
    - Lutz Chapter 3: How You Run Programs