W200 Python Fundamentals for Data Science - Syllabus

Univ. of California-Berkeley, Information School, MIDS Program, Summer 2018

Developers:
Paul Laskowski: paul@ischool.berkeley.edu
William Chambers
Kay Ashaolu

Section Instructors:
Gerald Benoit: gbenoit@berkeley.edu
Gunnar Kleemann: gunnarkl@berkeley.edu
Christopher Llop: christopher.llop@ischool.berkeley.edu

Office Hours:
Chris: Monday: 2:00-3:00pm PST

Gerry: Tuesday: 5:30-6:30pm PST
Wednesday 3:00-4:00 PM PST
Others by appointment

Gunnar: Tuesday: 12:00-1:00 PM PST
Thursday: 8:00-9:00 PM PST

Prerequisites: There are no prior courses required, but due to the fast pace of the course material, previous experience in a general-purpose programming language is required.

Course Description:
The Python programming language is an increasingly popular tool for the manipulation and analysis of data. This fast-paced course aims to give students the fundamental Python knowledge necessary for more advanced work in data science. The course structure provides students with frequent opportunities to practice writing code, gradually building to an advanced set of skills focused on data science applications. We begin by introducing a range of Python objects and control structures, then build on these with classes and object-oriented programming. A major programming project will reinforce these concepts; give students insight into how a large piece of software is built, and give students experience in managing a full-cycle development project. The last section of the course is devoted to two popular Python packages for data analysis, Numpy and Pandas. The course ends with an exploratory data analysis, in which students apply a script-style of programming to describe and understand a dataset of their own choosing. Aside from Python, the course also spends time on several other technologies that are fundamental to the modern practice of data science, including use of the command line, Jupyter notebooks, and source control with Git and GitHub.

Learning Objectives:
After completing this course, students will:

- Be able to navigate a file system, manipulate files, and execute programs using a command line interface
- Understand how to manage different versions of a project using Git and how to collaborate with others using Github
- Be fluent in Python syntax and familiar with foundational Python object types
- Be able to design, reason about, and implement algorithms for solving computational problems
- Understand the principles of object-oriented design and the process by which large pieces of software are developed
- Be able to test and effectively debug programs
- Know how to use Python to extract data from different type of files and other sources
- Understand the principles of functional programming
- Know how to read, manipulate, describe, and visualize data using the NumPy and Pandas packages
- Be able to generate an exploratory analysis of a data set using Python
- Be prepared for further programming challenges in more advanced data science courses

Methods of Instruction:
Students log into the Data Science homepage (https://learn.datascience.berkeley.edu/) and then select their course section. From there, select Coursework to view the *asynchronous* lessons. Each week there are a number of videos, references to required readings, and additional readings/videos. After the asynchronous activities, students attend an online video chat "live" section to review the materials and participate in breakout sessions to work with other students on activities. There are a number of weekly activities, projects, and in-class breakouts that require a computer, Python3, and Jupyter.

Required Textbook and other Readings:

- Lubanovic, B. (2014). *Introducing Python: Modern computing in simple packages*. Sebastopol, CA: O'Reilly Media. [can be viewed using the UC Berkeley library VPN]
- Goodrich (2013). *Data Structures and Algorithms in Python* (chapter 3). [Available from study.net]
- Recommended but Optional Textbook:
  - McKinney, W (2012). *Python for Data Analysis*. Sebastopol, CA: O'Reilly Media. [can be viewed using the UC Berkeley library VPN]

Online Tools needed for class:

- **Study Net**: Study.net has some of the reading materials required for the course: http://www.study.net/r_mat.asp?crs_id=30133588
- **GitHub:** Load and configure GIT: github.com
- **Anaconda with Jupyter Notebooks:** download here; python3 is used in this course
- **I School Virtual Campus:** https://learn.datascience.berkeley.edu/
  - On the left bar is a calendar icon, click on this to bring up the Meeting page. Below the *Upcoming Meetings* heading, mouseover the section and click on the video icon on the right to start Zoom for the session's live class.
- **Google group:** primary way to talk to all class sections for this term. Use the google group to talk with your classmates and ask questions (please don't share code) w200-python-2018-summer@googlegroups.com
- **Slack** - use Slack to communicate with your classmates, ask questions, share problems and your successes. https://ucbischool.slack.com/messages/C5AL99BU6/

Section Dates:

Week	Date	Topic & Activities
Week 1:	May 8-10	Introduction to Programming, Command Line, Source Code
	Readings	
	Homework	Assignments are made available via GitHub after the synchronous video session
Week 2:	May 15-17	**Starting out with Python**
	Readings	Lubanovic - Chapter 2 and the beginning of chapter 4, up to but not including the section labeled "cancel with break" [pp. 69-75]. https://mkaz.blog/code/python-string-format-cookbook/
	Homework 1 due	11:59 Pacific Daylight Time - the day before your section meeting
Week 3:	May 22-23	**Sequence Types & Dictionaries**
	Readings	Chapter 3
	Homework 2 due	11:59 Pacific Daylight Time - the day before your section meeting
Week 4:	May 29-31	**More about control and algorithms**
	Readings	Chapter 4, start with "cancel with break" (p. 75) through "comprehensions" (p. 85)
	Homework 3 due	11:59 Pacific Daylight Time - the day before your section meeting
Week 5:	Jun 5-7	**Functions**
	Readings	Chapter 4, start with Functions, p. 85.
	Homework 4 due	11:59 Pacific Daylight Time - the day before your section meeting
Week 6:	Jun 12-14	**Complexity**
	Readings	See chapter 3 from the Goodrich et al. (2013) *Data structures* (available from study.net)
	Project 1 assigned	
	Homework 5 due	11:59 Pacific Daylight Time - the day before your section meeting
Week 7:	Jun 19-21	**Classes**
	Readings	Chapter 6 through "Define a Class with class" and next "In self Defense" through "Method Types"
	Homework - Scrabble	11:59 Pacific Daylight Time - the day before your section meeting

Week 8:	Jun 26-28	**Object-Oriented Programming**
	Readings	The rest of Ch 6: "inheritance" through "In self defense" and "Duck typing" to the end of the chapter. https://learnpythonthehardway.org/book/ex43.html
	Exam 1	Window starts; distributed on ISVC; you have 24 hours to complete from when you start the exam
Week 9:	July 3-5	[National Holiday in the US; a make-up class will be scheduled]
	Readings	Chapters 7 and 8 through JSON
	Exam 1	Due before class!
	Homework 7 due	11:59 Pacific Daylight Time - the day before your section meeting
Week 10:	July 10-12	**Working with text and binary data**
	Readings	There are none assigned this week.
	Project 1	Presentations – project 1 turn-in at 11:59 Pacific time the day AFTER class
Week 11:	July 17-19	**NumPy**
	Readings	There are none assigned this week.
	Project 2 assigned	
	Homework 9 due	11:59 Pacific Daylight Time - the day before your section meeting
Week 12:	July 24-26	**Data analysis with Pandas**
	Readings	There are none assigned this week.
	Project 2 Proposal	Due 11:59 Pacific Daylight Time - the day before your section meeting
Week 13:	July 31-Aug 2	**More analysis with Pandas**
	Readings	Chapter 12, just the "Test with Unitest" section
	Exam 2	Window starts; distributed on ISVC; you have 24 hours to complete from when you start the exam
Week 14:	Aug 7-9	**Testing**
	Readings	There are none assigned this week.
	Project 2	Presentations – final project turn-in at 11:59 Pacific time the day AFTER class
	Exam 2	Due before class!

Congratulations!

Course Outline:

1. **Programming Languages, the Command Line, and Version Control (1 lecture)**: The course begins with an overview of programming languages and an introduction to some of the lower-level tools that support the work of a data scientist.

    - Programming Language Characteristics: Interpreted Versus Compiled, Low Level Versus High Level, General Purpose Versus Specialized
    - Using the Command Line
    - Version Control with Git
    - Collaboration with GitHub

2. **Python Objects and Basic Control Structures (5 lectures)**: We continue with a vocabulary-building survey of basic Python syntax, object types, and control structures. These elements are common to virtually all programming applications. Students will gain experience designing algorithms and organizing code logically into functions and modules.

    - Important Python Object Types
    - Iteration and Conditionals
    - Functions
    - The Design of Algorithms
    - Writing and Presenting Code in Jupyter Notebooks
    - Python Modules and Packages
    - Big-O Notation

3. **Classes and Object-Oriented Programming (3 lectures)**: We will spend three weeks discussing classes, as well as the larger practice of object-oriented programming. This section of the course will give students a view into how large production systems are organized and developed. At the end, students will complete a significant coding project that will reinforce their understanding of object-oriented development.

    - Classes and Attributes
    - Class Inheritance
    - Object-Oriented Programming
    - Project 1

4. **Using Python's Packages for Data Analysis (6 lectures):** We introduce the basics of data analysis using Python's system of scientific programming packages. Students learn the common tools that form the basis of the PyData ecosystem and gain experience with programming in a functional style. The final two weeks of the course give students time to work on a final data analysis project, while emphasizing good practices for developers.

    - File Input-Output
    - Text Encoding
    - Common Structure File Formats
    - Functional Programming
    - NumPy Arrays
    - Pandas Series and DataFrames

- Basic Data Set Manipulation With pandas
- Plotting with Matplotlib
- Descriptive Statistics
- Test-Driven Development
- Resources for Further Development
- Project 2

Grading:

1. 10 Weekly Assignments - 30% (3% each)
2. 2 Projects - 40% (20% each)
3. Midterm Exam - 10%
4. Comprehensive Final Exam - 10%
5. Participation - 10%

Weekly Assignments:
The weekly assignments are designed to reinforce and extend the programming concepts in each live session. A typical assignment consists of several programming exercises of varying difficulty. While some exercises can be completed in a single line of code, others may require students to combine commands in innovative ways, to design their own algorithms, and to navigate common sources of documentation. Students may consult with each other about the assignment but must write their own code and list their collaborators in their submissions. The expectation is that these assignments will take around 10 hours to complete each week. Depending on your experience with coding, this time estimate might be more or less. The weekly assignments are due at 11:59 PM PST (or PDT) the day before the next live session (e.g. if you have live sessions on Tuesday then the assignments are due the following Monday night at 11:59 PM PST). Weekly assignments will be released via GitHub.

Group Projects:
There are two large coding projects. The first is an individual project that comes at the end of the discussion of object-oriented programming and allows students to practice designing a multi-class program using best coding practices. The second is a group project that comes at the end of the course and involves the analysis of an actual data set using Python's system of data analysis packages. Further details about the projects will be given during the school term.

Exams:
The midterm and final exams are cumulative. Unlike the weekly assignments, students must do all of their work independently. Both exams include multiple-choice and short-answer questions, as well as short programming tasks, designed to test each student's grasp of important programming concepts. Further details about the exams will be given during the school term.

Participation:
Students are expected to participate in class activities, to contribute to discussions held in live session and on other platforms, to behave professionally towards classmates, and to help maintain a supportive atmosphere for education. Participation scores will be assigned based on these criteria

Academic Integrity:
Please read UC Berkeley's policies around academic integrity: http://sa.berkeley.edu/conduct/integrity

Avoiding Plagiarism:
Plagiarism is a serious academic offense, and students must take care not to copy code written by others. Beginning students sometimes have trouble identifying exactly when plagiarism takes place. Remember that it is generally fine to search for examples of code (for example, on forums such as stackexchange). This is a normal part of programming and can help you learn. However, it is important that you understand the code you find and use what you learn to write your own statements. It is ok if a single line of code happens to match an example found on the internet, but you should not copy multiple lines at once. If in doubt, simply document the place you found your example code and ask your instructors for further guidance.

Accommodations:
We make every effort to address the needs of students with physical, medical, and learning disabilities. See https://disabilitycompliance.berkeley.edu and https://dsp.berkeley.edu. UC Berkeley's Center for Teaching & Learning maintains the academic calendar and student accommodations policies and guidelines. Here you will find the policy for accommodating religious creeds, religious holidays calendar, the honor code, in which all students will "act with honesty, integrity, and respect for others."
[https://teaching.berkeley.edu/academic-calendar-and-student-accommodations-campus-policies-and-guidelines]

Late/Extension Policy: We cannot accept late assignments. If an assignment is turned-in late a 50% reduction per day will be applied. No assignments will be accepted after the live session that that assignment is discussed.

Assignment Help:
If you are stuck on a problem for more than an hour please reach out to get some help. There are many avenues to aid students:

1) Search for the error message or problem - stack overflow is a good coding help resource.
2) Google Group email with fellow students - please don't share code though!
3) Use Slack to chat among yourselves.
4) Come to Instructor Office Hours – times and appointments will be posted on the ISVC wall.
5) Email the instruction team – please include all of us on your email so we can respond faster; you can also email us your code so we can have a look.