Master of Information and Data Science (MIDS)

DATASCI W200 Introduction to Data Science Programming

Course Description

This course is a fast-paced introduction to computer programming tailored to the needs of data science professionals. It uses Python as its main language and incorporates frequent coding exercises to build each student’s capabilities. The course begins by introducing foundational Python objects, then explains how these are combined to form useful functions and classes. The first of two projects is an object-oriented exercise, giving students insight into how large-scale software systems are developed. The last section of the course is devoted to two popular Python packages for data analysis, NumPy and pandas. The final project consists of a data analysis, giving students the chance to explore 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 the command line, Jupyter notebooks, and source control with Git and GitHub.

Course Goals and Objectives

By the completion of 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 types 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

Required Textbooks:

- Lubanovic, B. *Introducing Python: Modern computing in simple packages.* Sebastopol, CA: O’Reilly Media. Ed 1 or Ed 2
- Goodrich et al - chapter 3 (2016) *Data structures* (available from study.net)
- McKinney, W. (2017). Python for data analysis: Data wrangling with Pandas, NumPy, and IPython. O'Reilly Media, Inc. Chapter 4

Methods of Instruction

Each unit of the course includes asynchronous materials, consisting of pre-recorded video lectures and required readings. After completing these materials, students attend a synchronous live session over video chat. Each live session is overseen by an instructor and includes group discussions, coding using Jupyter notebooks, and other activities. Homework for each unit is assigned after live session and due the following week.

Grading

Assignment	Weight
Unit Homework (x10)	40% (4% each)
Project 1	15%
Project 2	15%
Midterm Exam	10%
Comprehensive Final Exam	10%
Participation	10%

Unit Homework Assignments:

The unit homework assignments are designed to reinforce and extend the programming concepts covered in the asynchronous content and live section. A typical homework 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 10-20 hours to complete each week. Depending on your experience with coding, the time required might be much higher or lower. Each unit homework assignment is due by 11:59 PM PST (or PDT) the day before the next live section (e.g., if you have your live section on Tuesday, then the assignments are due the following Monday night at 11:59 PM PST). Please upload your completed assignments to GitHub as well as Gradescope. The official version of your assignment for grading purposes will be the one uploaded to Gradescope.

Projects:

There are two large coding projects. The first comes at the end of the discussion of object-oriented programming and prompts students to design 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. Students must do all of their work independently on each exam. 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:

We believe in the importance of the social aspects of learning. Knowledge is built on an individual level, but also by social activity involving peers and instructors. Participation and communication are key aspects of this course that are vital to the learning experiences of you and your classmates.

Students are required to join live sessions with video turned on and with a headset for clear audio, without background movement or background noise, and with an internet connection suitable for video streaming. Microphones should be muted when not talking. In exceptional circumstances, if you are unable to meet in a space with no background movement, or if your connection is poor, make arrangements with your section's instructor (beforehand if possible) to explain your situation.

Students are expected to participate in class activities, to contribute to discussions held in the live section 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.

General Grading Philosophy

The course will be graded on an absolute scale, and the grades will not be fitted to a specific curve. This is a graduate-level course, and we trust that different students will have varying levels of interests in the different subjects in the course. As such, the grading scheme is designed to acknowledge this intellectual diversity.

Late Submission Policy

We recognize that sometimes things happen in life outside the course, especially in MIDS where many students have full time jobs and family responsibilities. To help with these situations, we are giving you 6 "late days" to use throughout the term as you see fit. Each late day gives you a 24 hour (or any part thereof) extension to any assignment in the course except the project presentations or exams. (UC Berkeley needs grades submitted very shortly after the end of classes.) Once you run out of late days, each 24 hour period (or any part thereof) results in a 10 percentage point deduction on the grade for that assignment. You can use a maximum of 2 late days on any single assignment. We will not accept any submissions more than 48 hours past the original due-date, even if you have late days (for example on the assignments due 11:59 PM PST Monday, the latest that assignment will be accepted is 11:59 PM PST Wednesday). If you have very extenuating circumstances (for example, an emergency medical situation), please reach out to your section instructor.

Tutoring

If you have difficulty with programming concepts in the course, please reach out to the tutor TAs to set up a 1-on-1 (or small group) tutoring session. This is a free resource available to all students who might need extra help with the course material.

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. Post questions on Slack, but don't share code.
3. Come to Instructor office hours.

4. Email the instruction team at mids-python-instructors@googlegroups.com. Please use this email address as it contains all of the instructors so we can respond faster; you can also email us your code so we can have a look.
5. Request a 1-on-1 meeting with a tutor TA.

Collaboration Policy/Academic Integrity

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

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 (e.g., 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 okay 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 instructor for further guidance.

Diversity and Inclusion

Every one of our students has the potential to be an amazing computer programmer, regardless of their skin color, religion, sexual orientation, or gender identity. It is vital for our educational mission that we create a classroom environment in which all of our students are welcome and respected. We recognize that the field of computing has evolved to include language and practices that are biased against certain groups of people. It is our aim to identify and address these biases wherever we can, so that we can better support all of our students. If we say something that makes you feel uncomfortable or excluded, please reach out to your instructor. You may also contact the student affairs team at studentaffairs@ischool.berkeley.edu.

Disability Services & 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 policies for student accommodations. Here you will find the policy for accommodating religious creeds, the religious holidays calendar, and the honor code. [https://teaching.berkeley.edu/academic-calendar-and-student-accommodations-campus-policies-and-guidelines]