# Python Fundamentals for Data Science
# Syllabus


| | | | |
|---|---|---|---|
| **Developers:** | Paul Laskowski | William Chambers | Kay Ashaolu |
| **Email:** | paul@ischool.berkeley.edu | wchambers@ischool.berkeley.edu | kay@ischool.berkeley.edu |
| **Place:** | Virtual/Online | | |


**Office Hours:** TBD

**Course Description:**
The Python programming language is an increasing 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. We begin by introducing a range of Python objects and control structures, then builds on these with classes and object-oriented programming. The last section of the course is devoted to two popular Python packages for data analysis, Numpy and Pandas, and how these can be applied to explore and understand a dataset. Students will gain experience in different styles of programming, including scripting, object-oriented design, test-driven development, and functional programming. Weekly programming exercises are designed to reinforce each programming concept, while two larger projects give students experience in developing a larger program and in manipulating a data set. 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.

**Course Format:**
The course is organized as an inverted classroom. During each week, students first work through a set of asynchronous materials, including video lectures, readings, and other activities. Once a week, students meet for a 2-hour live session, in which they connect with an instructor and other students in an online classroom. A functioning webcam and an audio headset are required to participate in the live sessions. Students must complete all assigned asynchronous material before the scheduled live session begins.

**Prerequisites:** Due to the fast pace of the course material, previous experience in a general-purpose programming language is strongly recommended.

**Required Textbook:**
Lubanovic, B. (2014). *Introducing Python: Modern computing in simple packages.* Sebastopol, CA: O'Reilly Media.

**Course Outline:**

The asynchronous content for the course is broken up into 13 units. Because the summer sessions semester is only 10 weeks long, you will need to work through more than 1 unit per week. Your instructors will supply you with a detailed schedule of which units correpond to which weeks of the course.

The units of the course are organized as follows.

1. Introduction to Programming and Fundamental Data Science Tools (1 unit). The course begins with an overview of programming languages. Students then learn about some of the underlying tools that will be used through the course and which are indispensable to the practice of data science. These include the command line interface, source control with Git, and the sharing of projects with GitHub.

    - Programming Language Characteristics: Interpreted Versus Compiled, Low Level Versus High Level, General Purpose Versus Specialized
    - Basic Command Line Tools
    - Source Control With Git
    - Managing Projects with GitHub

2. Python Objects and Basic Control Structures (3 units). The next section of the course is a vocabulary-building survey of important Python syntax and object types. Students gain experience writing short segments of code in a script style using Jupyter notebooks.

    - Important Python Object Types
    - Iteration and Conditionals
    - Functions
    - The Design of Algorithms
    - Writing and Presenting Code in Jupyter Notebooks

3. Modules, Classes, and Object-Oriented Programming (4 units). We will spend 4 units on structures for modularizing larger pieces of code, culminating in a discussion 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.

    - Python Modules and Packages
    - Classes and Attributes
    - Class Inheritance
    - Object-Oriented Programming
    - File Input-Output
    - Using Common Structure File Formats
    - Text Encoding

4. Using Python's Packages for Data Analysis (3 units). 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.

    - Functional Programming
    - NumPy Arrays
    - Pandas Series and DataFrames
    - Basic Data Set Manipulation With pandas

- Plotting With Matplotlib
- Descriptive Statistics

5. Final Project Work and Best Coding Practices (2 units). The final part of the course gives students time to work on a final data analysis project, while emphasizing good practices for developers.

   - Final Project Work
   - Test-Driven Development
   - Resources for Further Development

**Grading:**

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

The weekly assignments are due six days after the corresponding live session. Students will have at least two weeks to work on each group project.

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.

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 multiclass 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.

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 projects and 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.

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 like 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.