Fundamentals of Programming With Python
Syllabus

Spring 2016

Instructors:	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:
A fast-paced introduction to the Python programming language geared towards students of data science, the course begins 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 Python's system of packages for data analysis. 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, coding, and presentation with Jupyter notebooks, and source control with Git and GitHub.

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:

1. Python Objects and Basic Control Structures (4 lectures) The course begins with an overview of programming languages and 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.

    - Programming Language Characteristics: Interpreted Versus Compiled, Low Level Versus High Level, General Purpose Versus Specialized
    - Important Python Object Types
    - Iteration and Conditionals
    - Functions
    - The Design of Algorithms
    - Basic Command Line Tools
    - Source Control With Git
    - Writing and Presenting Code in Jupyter Notebooks

2. Modules, Classes, and Object-Oriented Programming (5 lectures) We will spend 4 weeks 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
    - Test-Driven Development

3. Using Python's Packages for Data Analysis (3 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.

    - Functional Programming
    - NumPy Arrays
    - Pandas Series and DataFrames
    - Basic Data Set Manipulation With pandas
    - Plotting With Matplotlib
    - Descriptive Statistics

4. Final Project Work and Best Coding Practices (2 lectures) The final two weeks of the course give students time to work on a final data analysis project, while emphasizing good practices for developers

    - Final Project Work
    - Code Hygiene
    - Resources for Further Development

Grading:

1. Weekly Assignments - 30%
2. 2 Projects - 40% (20% each)
3. Midterm Exam - 15%
4. Comprehensive Final Exam - 15%

The weekly assignments and two exams are due one week after they are assigned. Students will have at least two weeks to work on each group project.

Weekly Assignments:
The weekly assignments are integrated throughout each asynchronous lesson. Most consist of short exercises to practice using a new Python object or coding technique immediately after it is introduced. Students are expected to finish the exercises before continuing with the asynchronous video. Some weeks include a somewhat larger exercise that occurs at the end of the asynchronous material. 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 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.