HARVARD T.H. CHAN | SCHOOL OF PUBLIC HEALTH

BST 273
Introduction to Programming
Fall 2 2019
MW, 11:30 - 1:00pm, FXB G13

Version: 2019-10-29

Instructor Information

Faculty

Dr. Eric Franzosa
Research Scientist
Office: HSPH, Building 1, Room 412 (*office hours will be held at another location*)
E-mail: franzosa@hsph.harvard.edu (*please include "BST 273" in the subject line*)

Office hour:
Friday, 11-12pm, HSPH, Building 2, Room 428

Teaching Assistants

Yuri Ahuja
E-mail: yuri_ahuja@hms.harvard.edu
Office hour: Weds, 2-3pm, HSPH, Building 2, Room 434

Luli Zou
E-mail: lulizou@fas.harvard.edu
Office hour: Weds, 4-5pm, HSPH, Building 2, Room 434

Kareem Carr
E-mail: kareemcarr@fas.harvard.edu
Office hour: Thurs, 1-2pm, HSPH, Building 2, Room 401

Credits

2.5 credits

Course Description

This is an introductory course on computer programming in Python 3. Students will begin by exploring the fundamental elements of a computer program (e.g. variables, data objects, functions/methods, and flow of control) along with their associated Python syntax. In the second part of the course, students will gain familiarity with modules in the Python Standard Library for reading and writing data, interacting with external programs, and writing command-line interfaces. The course concludes with an introduction to a number of advanced topics, including scientific computing and Object-Oriented Design. Interspersed with programming content, students will be introduced to important related skills, such as debugging and command-line navigation.

The course consists of two 90-minute meetings per week, divided into lecture/discussion sections and hands-on activities. Students will complete weekly assignments to reinforce their new programming skills, starting with basic tests of new syntax and leading toward the completion of functional Python scripts. Each student will complete a final project (of their own design, or based on a template) that involves developing, testing, and documenting a Python script to solve a problem in data analysis.

Note: *This course is NOT intended as an introduction to the Python language for experienced programmers. Students with prior programming experience (excluding MATLAB and R) should consult with the instructor before enrolling.*

Pre-Requisites

- None

Learning Objectives

Upon successful completion of this course, you should be able to:

- Design, write, and test Python scripts to solve problems in data analysis
- Identify and invoke appropriate (existing) software modules to aid in script design
- Enroll in more advanced courses requiring a programming background
- Learn additional programming languages through self-guided study

Course Readings

The required (*free, online*) textbook for this course is Think Python 2e by Allen B. Downey (Green Tea Press). The textbook is available for browsing or download at https://greenteapress.com/wp/think-python-2e/. Make sure you are looking at the second edition (which is updated to Python 3). The website also provides a link to purchase a hard copy of the book (*not required*).

Author: Allen B. Downey
Title: Think Python: How to Think Like a Computer Scientist 2nd Edition
Publisher: O'Reilly Media; 2 edition (December 28, 2015)
ISBN-10: 1491939362
ISBN-13: 978-1491939369

Recommended books:

Get Programming: Learn to code with Python
Author: Ana Bell
Title: Get Programming: Learn to Code with Python
Publisher: Manning Publications; 1 edition (April 19, 2018)
ISBN-10: 1617293784
ISBN-13: 978-1617293788

Students will also complete reading assignments from the official Python documentation: https://docs.python.org/3/.

Course Structure

Canvas Course Website: *https://canvas.harvard.edu/courses/62213*

Technical Information: This course is suitable for students with no programming experience. The course will also be beneficial for students whose exposure to programming has been informal or limited to numerical computing environments (such as MATLAB or R). Students are expected to bring a laptop computer to class to participate in in-class programming activities. Students who do not have access to a laptop should contact the instructors to make alternative arrangements.

Grading, Progress and Assessment

This course assumes student participation. General discussion of theory and practice is encouraged and expected of all students. At a minimum, being informed requires class attendance, completion of assigned readings and homework. Class attendance and thoughtful participation are important and will be reflected in part in the final grade. Please notify the instructor of an anticipated absence at least 24 hours before the lecture meeting.

The final grade for this course will be based on:

- Four homework assignments (15% × 4 = 60%)
- Final project (30%)
- Participation (10%)

Homework assignments (60%)

After the first week, students will complete weekly homework assignments to reinforce the previous week's material (four in total). Homework will be posted by class time on Monday and will be due the following Friday (11:59pm) via Canvas hand-in. Homework assignments are designed to teach Python syntax and encourage computational thinking around algorithmic challenges. Points will be scored based on the syntactic and semantic correctness of the submitted results, with students expected to match pre-set input, output, and behavioral specifications for written code. Especially early on, assignments will also include a small number of written answers to assess conceptual understanding.

Final Project (30%)

Students will work on final projects during the final two weeks of the course. Final projects will involve an open-ended implementation of a Python script to solve a problem in data analysis. Students will be able to choose between completing a default final project specification (introduced during the third-to-last week of the course) or proposing their own final project. (Students electing the latter option MUST seek and receive instructor approval before proceeding.) In addition to working code, students will turn in sample input and output data along with documentation in the form of a README file (as a single Canvas hand-in). Projects will be graded based on the correctness of the submitted code and completeness of the accompanying documentation.

Participation (10%)

Students are expected to attend and participate in lecture and in-class activities. Participation includes physical presence in class, asking and answering questions, sharing viewpoints in constructive and respectful ways, working diligently on in-class assignments, and otherwise actively engaging with other students and the course instructors. The floor of the participation grade will be set by the completion of ungraded in-class surveys roughly every-other lecture and will fall off exponentially after the first missed survey.

Collaboration Policy

Students are expected to complete their out-of-class assignments individually. While it is tempting to compare code with other students to check correctness and/or move past a stumbling block, within the context of homework assignments and the final project, this is strictly not allowed. Deviations from this policy will be considered a violation of the school's Academic Integrity policy and treated accordingly.

Outside of comparing assignment code, students are allowed and encouraged to help each other in and outside of class. This includes working together or comparing code during in-class programming assignments. In addition, discussing general concepts (e.g. "could you explain the difference between a list and a set?") is also allowed, as long as it does not involve specific review of assignment code. (As a general rule, if the people in the discussion are not looking at the current assignment, then you are probably not reviewing assignment code.)

To help with checking code correctness, assignments will be bundled with expected inputs and outputs for testing. To help with stumbling blocks, students are encouraged to reach out to the instructors and/or TAs during the week (preferably during scheduled office hours). Searching for help online (e.g. "what does 'ZeroDivisionError' mean?") is also allowed and encouraged. However, posing specific homework questions online (e.g. via StackExchange) is not permitted.

Late Work Policy

The maximum score for late work will fall rapidly: 90% if one day late, 75% if two days late, 50% if three days late, and all credit lost if four or more days late. Extensions may be granted if requested with reason at least 24 hours in advance of the assignment deadline.

Final Grades

Final letter grades will be curved based on the percentiles of total scores received by students in the class.

Lecture & Assignment Schedule

Lec	Date	Day	Unit	Lecture Part I	Assignment
01	2019-10-28	M	Orientation	Welcome and setup	#
02	2019-10-30	W	Fundamentals	Data, transformations, and variables	#
03	2019-11-04	M	Fundamentals	Collections (Lists) and iteration	HW1
04	2019-11-06	W	Fundamentals	Collections (Dicts, Sets) and helper functions	HW1
#	2019-11-11	M	#	HOLIDAY (Veterans Day)	HW2
05	2019-11-13	W	Fundamentals	Conditional logic and flow of control	HW2
06	2019-11-18	M	Fundamentals	Writing functions, references vs. data	HW3
07	2019-11-20	W	Fundamentals	Introduction to modules and file I/O (sys, csv)	HW3
08	2019-11-25	M	Review	Testing, debugging, getting online help	#
#	2019-11-27	W	#	HOLIDAY (Thanksgiving)	#
09	2019-12-02	M	Special topics	OS and software interaction (os, subprocess)	HW4
10	2019-12-04	W	Special topics	Command-line interfaces (argparse)	HW4
11	2019-12-09	M	Special topics	Regular expressions (re)	Final Project
12	2019-12-11	W	Special topics	Scientific computing with Python (numpy, scipy, pandas)	Final Project
13	2019-12-16	M	Special topics	Object-oriented programming	Final Project
14	2019-12-18	W	Wrap-up	Next steps for developing as a programmer	Final Project

**Note: lecture topics, especially those in the second half of the course, are subject to change pending class progress and interests.*