ENGL-GA.1972-001 Introduction to Programming: Python for Humanists
Professor David Hoover (English) and Deena Engel (Computer Science)
New York University, Fall 2015

Course description:

This course introduces humanities students to the fundamentals of computer programming as students design, write, and debug computer programs using the programing language Python. No knowledge of programming is assumed. The approach in this course is to focus on text and textual manipulation while building useful applications in a variety of disciplines.

Overview:

In the first unit of the course, students will study the fundamentals of computer programming using Python. The course would begin by tapping into the familiarity of Humanities students with language – along with its textual forms and linguistic structures – by discussions and an opening assignment that draws analogies between programming and natural languages. Topics in semantics, grammar, and syntax will be discussed to help Humanities students understand the basics of writing computer programs. Topics in Python will include data types; selection statements; iteration; functions and modules; lists and dictionaries; and working with text and data files, among others. These topics will be taught in a traditional classroom/lecture discussion format, followed by weekly programming assignments in hands-on lab sessions.

The second unit of the course will be project-based as students hone their programming skills to build useful programming applications to support scholarly research in the Humanities, such as textual analysis tools to examine word frequency and collocation; programs to "scrape" data from the web and prepare data and text files for further research and analysis; and related topics.

Case studies of programming projects in the Humanities will be presented in class and evaluated from both scholarly and technical perspectives.

Grading:

Graded work will consist of the following:

- 10 homework assignments at 7% each. **(70%)**
- Final project **(15%)**

  Each student will design and implement a textual analysis research project using Python to manage a selected corpus and produce original and meaningful research results of interest to the student's discipline. Students in literature and language, history, and other fields have access to a wide variety of digitized documents and textual resources that are appropriate for analysis using the skills and methods they will have learned in this course. Starting in Week 6, the students will be working with texts that are relevant to their fields of study, so that by Week 11 they will have both a suite of programs and a corpus of texts to use as a foundation for the project. The project will address a significant research question, such as studies in authorship attribution; stylometry; gender studies; genre; and other areas of inquiry that employ textual analysis and are relevant to the student's content field. The projects will be discussed in class and class time will be spent working

with the students on both the programming and the research aspects during class weeks 11, 12, and 13 (see below). Like many of the weekly assignments, the central pedagogical goal of the final project is to push students to more serious consideration of how the nature of the problem should guide the construction of projects and the writing of the programs that support their research. This approach insures that students focus early and often on designing projects that are not just computationally accurate but are also persuasive and satisfying to the contemporary humanities scholar.

- Final exam. (15%)

Readings:

Weekly readings will be assigned from a central course textbook, *Think Python* by Allen B. Downey (O'Reilly Media 2012) as well as from scholarly books and articles in the Digital Humanities.

Weekly Class Plan:

Week	Course Content
1	**Definition of a programming language; types of programming languages; comparisons between programming languages and natural languages; and an introduction to working in Python.** Reading (Python): *Think Python* Chapter 1 Reading (Theory): Historical Background Hockey, Susan. 2004. "The History of Humanities Computing." In Susan Schreibman, Ray Siemans, and John Unsworth, eds. *A Companion to Digital Humanities*. Oxford: Blackwell.
2	**Introduction to manipulating text in Python.** Introduction to working with characters and text ("strings") as well as textual user input and string manipulation. Relevant string methods in Python are introduced. Reading (Python): *Think Python* Chapter 2, Chapter 3 (Pages 23-25), Chapter 8 (Pages 85-86) Reading (Theory): General Introduction Hoover, David L. 2013. "Text Analysis." In Ken Price and Ray Siemens, eds. *Literary Studies in the Digital Age: An Evolving Anthology*. New York: MLA. Craig, Hugh. 2004. "Stylistic Analysis and Authorship Studies." In Susan Schreibman, Ray Siemans, and John Unsworth, eds. *A Companion to Digital Humanities*. Oxford: Blackwell.

3	**Selection statements and additional datatypes** Students will study selection statements ("if" and "if/else" structures), and additional data types (integers, floating point numbers, and Boolean values) will be introduced, along with the ASCII chart. Students continue to explore the Python API and become familiar with built-in string operations for string manipulation (e.g. capitalization, identifying characters within a string, formatting strings, etc.). Reading (Python): *Think Python* Chapter 5 (Pages 49-52), Chapter 8 (Pages 90-92) Reading (Theory): Seminal Early Work Burrows, John F. 1992. "Not Unless You Ask Nicely: The Interpretative Nexus Between Analysis and Information." *Literary and Linguistic Computing* 7: 91-109.
4	**Iteration (repetition) structures ("for" and "while" loops)** Students implement iteration to work with text. Reading (Python): *Think Python* Chapter 7 and Chapter 8 (Pages 86-90) Reading (Theory): Provocative Recent Challenge to DH, Involving Alliteration Fish, Stanley. 2012. "Mind Your P's and B's: The Digital Humanities and Interpretation." *The New York Times*, January 23.
5	**Working with Python's list data structure.** Reading (Python): *Think Python* Chapter 10 Reading (Theory): Provocative Recent Approach Ramsay, Stephen. 2011. "An Algorithmic Criticism." In *Reading Machines: Toward an Algorithmic Criticism*. Champaign, IL: University of Illinois Press: 1-17.
6	**File manipulation:** Students write Python scripts to open text files and process the data. Students will be asked to download out-of-copyright files containing historical documents and literary texts from both publically available archives (e.g. http://www.gutenberg.org/) and online archives available through the Bobst library. Students are encouraged to select texts and build a corpus consisting of works that are relevant to each student's field of study for this and future assignments. Students will be introduced to additional character sets beyond ASCII. Reading (Python): *Think Python* Chapter 14 (pages 160-163) Reading (Theory): Classic Study with Significant Disciplinary Interest Jordan, Ellen, Hugh Craig, and Alexis Antonia. 2006. "The Brontë Sisters and the Christian Remembrancer: A Pilot Study in the Use of the 'Burrows Method' to Identify the Authorship of Unsigned Articles in the Nineteenth-Century Periodical Press." *Victorian Periodicals Review* 39: 21-45 Dridan, Rebecca, and Stephan Oepen. 2012. "Tokenization: Returning to a Long Solved Problem: A Survey, Contrastive Experiment, Recommendations, and Toolkit." *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*: 378-82.

7	**Functions and modules:** Students learn how to modularize code (by writing functions) and to build Python modules. Reading (Python): *Think Python* Chapters 3 (pages 25-32) and 6 (Pages 61-65) Reading (Theory): Influential Recent Work Jockers, Matthew L. 2013. "Style." In *Macroanalysis: Digital Methods and Literary History*. Champaign, IL: University of Illinois Press: 63-104.
8	**Files and directories:** Students learn to use Python to programmatically read the contents of a file directory and process all of the appropriate files. Students will also begin scraping data from the web. Reading (Python): *Think Python* Chapter 14 (pages 161-162, 169) Reading (Theory): Influential Recent Work Jockers, Matthew L. 2013. "Nationality." In *Macroanalysis: Digital Methods and Literary History*. Champaign, IL: University of Illinois Press: 105-17.
9	**Python dictionaries** Students will be introduced to the Python dictionary data structure. Reading (Python): *Think Python* Chapter 11 Reading (Theory): A Respected Practitioner Reflects Burrows, John F. 2010. "Never Say Always Again: Reflections on the Numbers Game." In Willard McCarty, ed. *Text and Genre in Reconstruction: Effects of Digitalization on Ideas, Behaviours, Products and Institutions.* Open Book Publishers.
10	**Introduction to graphics using Python** Reading (Python): *Python Mode for Processing (Tutorial)*: http://py.processing.org/tutorials/ Reading (Theory): An Influential and Controversial Approach with Interesting Visualization Issues Moretti, Franco. 2003. "Graphs, Maps, Trees: Abstract Models for Literary History–1." *New Left Review* 24: 67-93.

11,12,13	**Textual Analysis Research Projects** Following please find four sample projects: 1. A fully functioning, well-developed Python tokenizer with a clear and user-friendly interface and robust error handling. This project could take several forms. The tokenizer might be developed for a particular corpus of texts, with an emphasis on insuring that it correctly handles any problematic characteristics of that corpus. Alternatively the tokenizer might be a generalized one that is then tested on several varied corpora (in this case the project should discuss areas where modifications would be needed to give good results). Another version of this project would compare the results of the tokenizer with a series of existing tokenizers on one or more corpora. In all cases the project would include both the tokenizer itself and a discussion of it. 2. A fully functioning, well-developed Python Style-Impersonator with a clear and user-friendly interface and robust error handling. This project would use a Markov model to produce substantial passages of text that mimic as closely as possible the style of a text of the user's choice. The program should produce text that sounds as much as possible like the original. It should also allow for the selection of parameters (such as the order of the model and the size of the output), so that the user can examine how those parameters affect the output. 3. A study of speeches, using original Python programs. These could be political speeches (primary or general election speeches, nominating or acceptance speeches, presidential inaugural addresses, State of the Union addresses, speeches from congressional sessions), public addresses of various kinds, sermons, or any kind of speeches in which the student is interested. The programs could either do a complete analysis, based on textual features of the student's choosing, or could do the preliminary work of collecting and counting features and preparing output appropriate for various other statistical programs or existing DH tools. (In this kind of project, there might be a little less focus on the nature of the programs themselves and more on producing results of interest to the student's discipline.) 4. A study of Henry James's letters that examines the question of how/whether James's adoption of dictation for many of his letters beginning in 1897 changed the style of those letters. This project would use original Python programs to analyze any textual features of the student's choosing, and could, as in #3 either produce completed results or output that would be processed by other DH tools.
14	**Final Exam Review and Student Presentations**
	Final Exam (to be held during the University examinations period)