Methods in Computational Linguistics I

Course Number: LING78100
Mondays, 2pm - 4pm, Room 7395

Instructor Information:

Michael Mandel, PhD
Assistant Professor of Computer and Information Science at Brooklyn College, on the Computer Science and Linguistics PhD faculties at the Graduate Center
Email: mim@sci.brooklyn.cuny.edu
Web: http://mr-pc.org
Office: Graduate Center room 4410, Brooklyn College Ingersoll Hall 2232
Office hours: By appointment only

Practicum Sessions and Instructor Information:

This course has a co-requisite practicum session in which you should also enroll: Ling73600. The practicum will meet in room 7395 on Tuesdays from 4:15pm to 6:15pm. The instructor for the practicum is Hussein Ghaly, Ph.D. student in Linguistics, Graduate Center. During the practicum session, you will review concepts from class and see extended programming examples.

Description:

This is the first of a two-part course sequence to train students with a linguistics background in the core methodologies of computational linguistics. Successful completion of this two-course sequence will enable students to take graduate-level elective courses in computational linguistics; both courses offered by the Graduate Center's Linguistics program, as well as courses offered by the Computer Science program. As the first part of the two-part sequence, Methods in Computational Linguistics I will introduce computer programming at a level that will allow students to begin building computer applications that address various computational linguistic tasks. No previous programming experience is required. The programming language we will use is Python. We begin by learning the syntax of Python and how to program generally; we then focus specifically on linguistic application.

Who Should Take This Course:

This course is required for students pursuing the MA in Computational Linguistics or the PhD Certificate in Computational Linguistics at CUNY Graduate Center. Further, this course would be excellent for students who may be interested in research in computational linguistics or natural language processing (NLP). Other graduate students (including those outside of linguistics) who wish to gain basic programming skills in the Python language, which is useful for text processing and various linguistics and web applications, may also benefit from this course. Because this course introduces basic programming concepts, it would not be appropriate for graduate students in Computer Science.

Online:

Lecture slides, assignments, and handouts are available on the course website:

http://mr-pc.org/t/ling78100

Assignments should be submitted to the relevant dropbox on the course's page on Blackboard.

A Note on Pre-requisites:

This course is a prerequisite for Methods in Computational Linguistics II, usually offered in the spring.

Required Books:

We will be using the following textbook for the course:

- Python Programming: An Introduction to Computer Science, by John Zelle. Third Edition. Franklin Beedle & Associates publishers, 2017. ISBN: 978-1-59028-275-5. Retail Price: $40. Note: It is important that you get the third edition of the textbook because this matches the version of of the Python programming language that we will be learning in this course. The official website for the textbook has useful resources. http://mcsp.wartburg.edu/zelle/python/

Technical Requirements:

For assignments in this course, students will be expects to write computer programs in the Python programming language. Students may also be expected to take advantage of resources in the Natural Language Toolkit (NLTK) programming library. Students are expected to install Python 3.5 and the Natural Language Toolkit (NLTK) on their personal computers. Details about what specific versions of this software to install and how to configure it will be discussed during the first week of class. Students are advised to bring laptop computers to meetings of the practicum sessions.

Learning Goals:

By the end of the semester, students will:

1. Understand the basics of computer programming, including: names, functions, numerical and string data types, list-like container data types, control of flow, and other key concepts.
2. Be able to write short programs in the Python programming language
3. Be familiar with the use of software libraries for computational linguistic programming
4. Be able to accomplish some computer programming tasks that are useful for working in the field of computational linguistics, including: processing a text file or other basic string operations.
5. Be familiar with the basic topics that make up the field of computational linguistics.
6. Be prepared to learn more advanced programming and mathematical topics in the subsequent "Methods in Computational Linguistics II" course, which is the second half of this two-course sequence

Outline of Course Topics

- Computer Basics, Programming Code
- Introduction to Python and the IDLE editor vs Jupyter Notebooks
- Processing with Numbers: Data Types and Conversions
- String Processing with Python
- Opening, Reading, and Writing Data Files
- Booleans and Conditionals
- Basic Graphics and User-Interface Issues
- Designing Functions, Top-down Software Design
- Loops, Control of Flow, List Comprehensions
- Python Containers: Lists, Tuples, Dictionaries
- Defining and Using Classes, Object-Oriented Software Design
- Overview of Topics in Computational Linguistics
- Introduction to Machine Learning

Course Calendar:

Note that there is no class on the following days:
September 4 (Lecture)
September 19 (Practicum)
October 9 (Lecture)
November 21 (Practicum)

Assignments and Grading:

Details regarding due-dates for various homework assignments will be listed on the course website, as well as announced in class. There will be homework assignments every 2-3 weeks. All assignments will be scored out of 100 points.

All homeworks and projects should be turned in via blackboard at least 30 minutes prior to the beginning of the corresponding class period. Homeworks turned in late will be penalized 10% for each day they are late. An assignment that is turned in two days late and would have received a 90% will instead receive an 70%.

Attending class is mandatory and attendance will be taken at the beginning of every meeting. This rule does not apply to absences due to religious observances.

Grading Percentages for the Course:

- Regular homework assignments 60% (4 x 15%)
- Final homework assignment 25%
- Class participation & attendance 15%

Assignment Policy:

Do not cheat. You may discuss assignments with your classmates or with the practicum instructor, but the program that you hand in must be your own. Do not ask for or offer to share code or written assignments. If you discuss an assignment with a classmate or the practicum leader, you must indicate this in the documentation of your code, along with the name of the classmate or practicum leader. The first instance of cheating results in an automatic zero for the assignment (or final project). A second instance of cheating results in a zero (F) for the course. The Linguistics Department will be notified in writing of all instances of cheating. On a second instance a report will be submitted to the Office of Academic Integrity.

Incomplete Policy:

In extenuating circumstances, students may be given an Incomplete if material has not been completed by the end of the semester. When an incomplete is granted, the student and instructor will specify, in writing, a timeframe for all outstanding material to be submitted. If no other timeframe has been specified in writing, the deadline for all outstanding material to be submitted to resolve an incomplete will be one month following the last meeting of the class. This semester, that would make the deadline: January 18. An incomplete that is not resolved by the deadline will become an F.

Class Policies:

- **Come to class.** It will be difficult to do well in the class without regular attendance.
- Cell phones must be on silent, and are not to be checked or used during class - if you are expecting an urgent call, tell the instructor at the start of class.
- Laptops, tablets or lab computers are welcome in class so long as the sound is turned OFF.

Advice for all Students:

- **Begin assignments early!** When you are new to programming, it can be difficult to judge how much time you will need to write a program. Writing the first version of the program is only the start, it can take much longer to find the bugs in it and get it working correctly.
- Review the textbook chapter before class so that you can ask better questions during class.

How to Submit Your Assignments

For any assignment in which I ask you to write Python code, you should do the following steps:

1. Write the Python code and make sure that it runs without errors. If there are errors when you try to run it or if it does not work, then you should include a written explanation with your assignments that says that you could not get your code to run correctly. If asked, you should be able to email me any Python code you've written so that I can run it on the sample inputs.
2. You should include helpful comments in the code that explain what it is doing. It is most important to describe in the comments the inputs that a method expects and the output(s) that it produces along with a brief description of what it is doing. If you submit a method that does lots of calculations or assignments without explaining the underlying strategy of the method, then you won't get full credit. If it takes a lot of effort to understand your code because it is poorly commented, then this will reflect poorly on the final grade.
3. You must submit all of the Python code that you wrote via github. If you used some code that came with the textbook, that is fine (I will tell you when there are modules you can use), but I'd like you to **clearly indicate** the parts of the code were **not** written by you. (You should indicate this inside your code in comments.) If you just include dozens of pages of code and do not make it clear which portions you modified or wrote yourself, you will lose points.

Criteria for Grading Programming Assignments

Adapted from 'Methods in Computational Linguistics I' course, fall 2013 by Matt Huenerfauth and 2016 by Rachel Rakov

Programming assignments will be graded based on:

- Correctness: Does it do what was asked in the assignment? Did you prove it by running the code?
- Style: Did you follow good programming style? Good organization? Error handling?
- Understandability: Can someone else understand your code when reading it? Good comments? Good programming "style" usually follows many of the following principles:
- Include a comment at the top of the program stating: Program author, date, name/number of the assignment, and a brief description of what the program does
- Include concise comments that summarize major sections of your code
- Use meaningful variable and function names
- Comment each function with: description of what the function does, description of input parameters, and a description of the return values.
- Organize your code: group steps of a process logically; setup variables at the top of your code.
- Whitespace or comments improve legibility
- Avoid lines longer than 80 characters: Things shouldn't wrap-around because of how you print the code out. You can use \ to end lines early and wrap them.
- Avoid large blocks of copy-pasted code:
  - Never copy / paste from other people - that is cheating
  - If you are tempted to copy and paste your own code in order to repeat a process, you should very likely write a loop or function to do this repeated work instead