syllabus for

computing for poets

comp 131

Instructor: **Mark LeBlanc** SC-103 508.286.3970
mleblanc@wheatoncollege.edu Hours: MW 2-3, T 2-3
http://cs.wheatoncollege.edu/mleblanc

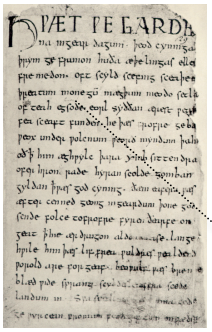
Meeting Times: Mon-Wed-Fri 9:30-10:20am, Room A102, Science Center

The use of computers to manage the storage and retrieval of written texts creates new opportunities for scholars of ancient and other written works. Recent advances in computer software, hypertext, and database methodologies have made it possible to ask novel questions about a poem, a story, a trilogy, or an entire corpus. This course exposes you to leading markup languages (XHTML, CSS, XML) and teaches computer programming as a vehicle to explore and "data mine" digitized texts. Programming facilitates top-down thinking and practice with computational thinking skills such as problem decomposition, algorithmic thinking, and experimental design. Programming on and with texts introduces students to rich new areas of scholarship including stylometry and authorship attribution. Prerequisites: A love of the written (and digital) word; no previous computer programming experience is required.

<poem>*Three Rings for the Elven-kings under the sky,*
Seven for the Dwarf-lords in their halls of stone,
Nine for Mortal Men doomed to die,
One for the Dark Lord on his dark throne
In the Land of Mordor where the Shadows lie.
One Ring to rule them all, One Ring to find them,
One Ring to bring them all
and in the darkness bind them
*In the Land of Mordor where the Shadows lie***.</poem>**
The Fellowship of the Ring, JRRT

<?xml?>

Using computers to analyze poems and stories is an exciting new area of research. Although many tools exist for working *with* texts, what do you do when the tools you have at your disposable cannot answer *your* questions? In this course, you will learn to write computer programs (also called "software" or "scripts") that can answer your original questions.



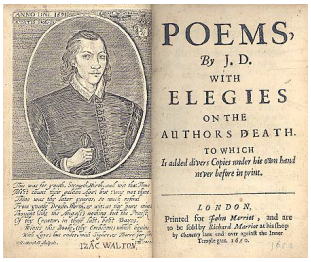
The programming language we will learn is named Python – a modern, accessible, yet powerful language when computing with texts.


```
array = line.split()
for word in array:
   # CASE 1: working on poetry
    if word in dictionary:
       counts[word] = counts[word]+1
```

Some of the assignments and labs that you will work on this semester will analyze texts to:

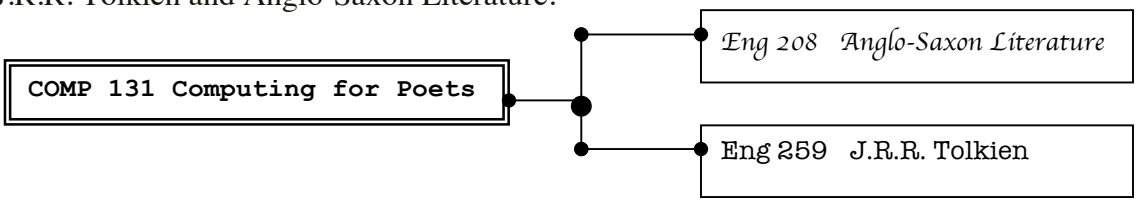
- set up a website hosted from a Linux Apache server
- learn XHTML, CSS, JavaScript, and XML
- study the beginning steps in authorship attribution as you keep statistics for the relative frequencies of the most commonly used words in a poem, story, or corpus
- design and implement a text mining experiment on a corpus of your choice

- write Python scripts to perform text mining experiments that produce results in Excel-ready or XHTML files; your programs (scripts) will perform the following:
  - ✓ compute the percentage of vowels in a text;
  - ✓ search for patterns of letters or words using the powerful pattern matching language of regular expressions ("regex"); for example, how many six letter palindromes can you find?
  - ✓ search Tolkien's Lord of the Rings trilogy to test the conjecture that the author tends to use words like "tall" near elf names; (*do you think he does?)*
  - ✓ search your own papers that you have written in the past for questionable writing style;
  - ✓ determine the top-10 most frequently used words in the Anglo-Saxon corpus;
  - ✓ while text mining the entire Anglo-Saxon corpus, find the words that appear in poetry that never appear in the prose; *(do you think there are any?)*
  - ✓ find *hapax legomena* (words that appear only once in an entire collection of works), for example, in fifty John Donne poems;
  - ✓ learn to read, markup, and parse XML documents, including those using the Text Encoding Initiative (TEI) schema;
  - ✓ text mine TEI documents using ElementTree (a subset of XPath), for example, to search for all `<person>`'s mentioned in Eliza Wheaton's nineteenth century pocket diaries


POEMS,
By J. D.
WITH
ELEGIES
ON THE
AUTHORS DEATH.
TO WHICH
Is added divers Copies under his own hand never before in print.
LONDON,
Printed for John Marriot, and are to be sold by Richard Marriot at his Shop by Chancery Lane end over against the Inner Temple gate. 1650.

CONNECTION

This course is "connected" with two courses from Professor Mike Drout in English: J.R.R. Tolkien and Anglo-Saxon Literature.



In Anglo-Saxon studies, the relationship between Old English poems has been a vexed question for nearly 150 years. Most of the poetry is anonymous and exists only as tenth-century copies in manuscripts (some of it is assumed to be much earlier). We have only three named authors of poetry in the Anglo-Saxon period (Cædmon, Cynewulf and King Alfred), and there are various problems with linking these names (much less their biographies) with more than a very few

specific poems. Although a few prose texts are by known authors (particularly the homilists Ælfric and Wulfstan), even the majority of the prose is anonymous. Thus for years scholars of Old English have struggled to divine relationships between texts based on vocabulary, meter, and style. These results have been at best contentious and at worst completely unsuccessful. As we proceed to learn more and more scripting in this course, we will set up and run experiments using the entire Anglo-Saxon corpus. Some of the questions you could ask may never have been asked before. For J.R.R. Tolkien, we will design markup schemas that help define structural elements across texts, e.g., the Lord of the Rings trilogy.

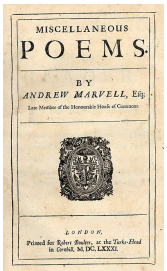
We will learn to combine Python programs, comma-separated output from those programs, and Excel spreadsheets in a manner very similar to the way your instructors do when in their Lexomics Research Group (see `http://lexomics.wheatoncollege.edu`). A major "take home story" for this course is to learn ways to view and analyze texts in an entirely new way. By leaning on the computer, new methods of analysis (that you could *not* do by hand) will now be at your disposal.

In addition to programming in Python and working in Excel, XHTML and XML, we will also study how computers store individual characters, including the traditional (English-only) ASCII character code and the international standard called UniCode.

At this point, you may be asking: "Why Python (as a choice of programming language)?" Good question (and it is one your instructor has considered at length). Python is an excellent choice for your first programming language. The interpreter environment encourages you to experiment. Note however, that Python is an industrial-strength programming environment. It is used increasingly in industry in a broad range of areas including but not limited to web sites, data and text mining, and scientific computing.

NOTE: This course is but an introduction to using computing to study written texts. Computers allow us to study texts in exciting new ways that we could not otherwise do; however, as we'll discuss at length, **we are wise if we keep in mind what computers cannot do**. The following quotes can help us (1) stay humble and (2) stay focused.

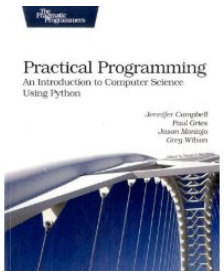
"As students of a powerful new form of scholarship, we have much to offer.
We do ourselves no justice when we forget that the quantifiable features
we deal in are but the shadow of a shadow."
John Burrows, Computers and the Humanities, v37, 2003, p30.


MISCELLANEOUS
POEMS
BY
ANDREW MARVELL, Esq;
LONDON,

"The onus of competency, clarity, and completeness is on the practitioner.
The researcher must document and make clear every step of the way.
No smoke and mirrors, no hocus-pocus, no 'trust me on this.' "
Joseph Rudman, Computers and the Humanities, v31, 1998, p353.

In computer science, if you are almost correct you are a liability.
Fred Kollett (1941-1997), MathCS, Wheaton College, Norton, MA

Text:



Practical Programming:
An Introduction to Computer Science Using Python
by Campbell, Gries, Montojo, and Wilson
Pragmatic Bookshelf (2009)

We'll be using lots of online websites as reference, especially those from Scott Kleinman, Associate Professor of English at California State University, Northridge. Scott is working with the Lexomics Research Group on some research.
He works in Old and Middle English literature *(http://www.csun.edu/~sk36711/WWW/)*.

I also strongly recommend that you buy a 3-ring binder. I'll pass out lots of handouts.

Your Grade:

Things to do	Grading Percents	Due Dates
Labs (lecture and lab are "blurred")	**5% overall**	in class as needed
6 Assignments	**55% overall**	
a1: Set up your website	5%	Fri., Feb. 5
a2: JRRT Madlibs	10%	Wed., Feb 17
a3: Regex Play	10%	Wed., Feb 24
a4: "tall" … "elf" in Tolkien	10%	Mon., Mar. 01
a5: Anglo-Saxon poetry	10%	Fri., Apr 02
a6: Marking-up JRRT	10%	Wed., Apr 14
Quizzes	**20% overall**	
Quiz I	5%	Fri., Feb. 19
Quiz II	5%	Fri., Mar. 12
Quiz III	5%	Wed., Apr. 07
Quiz IV	5%	Wed., Apr. 28
No Final Exam		
Final Project: Text Mining Experiment	**20% overall**	
Presentation	5%	last week of class
Paper: Methods, Results, Discussion	15%	Fri., May 07

Late Submissions:
Due is due. Always turn in whatever you have on time. Something turned in on time is much better than not having it accepted because it is late. Late is not an option. (Good, glad we can all agree with this). agree with this.)
Note: **Python Programs** are due on various dates (see detailed syllabus in moodle (onCourse)); however, since I know from experience that many students like to use the last night for testing, I will allow you to submit your programs until 5am the following day. For example, assignment a2 is due Wed., Feb. 17th (see above), but you can submit it electronically until 5am Thursday, Feb. 18th – but be careful! The course website (onCourse) makes it appear as if the program is due on Thursday, but remember, that means THUR at 5am!

Honor Code Revisited:

It goes without saying that all submitted work will be the student's own, in keeping with the Wheaton Honor Code, unless the assignment has assigned groups. For labs, you may get "help" from fellow classmates, but remember that all completed work must be your own. Use discretion; don't ask your colleague for "the" answer or for lines of code. However, I do encourage you to discuss the problem in general, such as the type of statements or functions one might use. For homework, your answers and software must be your own from beginning to end. Here is an analogy. Almost no one would every "use/steal" a line or two from another person's poem. Consider it the same with your programs. Don't "borrow/use" lines or sections of your program from another classmate. Your program is (like) your poem; everyone's program should be unique. Be wise. If a colleague is asking you for too much help, be honest and remind them your program is just that, *your* program.

Tips for working on your own ....

(0) It is expected that you spend at least 2+ hours on reading, study and preparation for every 50 minutes of lecture and discussion.

(1) It is expected that you spend at least 6-10 hours per week on your current programming assignment. WARNING: Programmers typically underestimate the time it takes to complete a software project; 6-10 hours per week on your programming assignment may be one of those "underestimations."

In classroom "LABS"

(0) The computer work in class (labs) are a critical part of the course. Appropriately so, it may be hard to distinguish if you are in lecture or "lab". In a way, hands-on classtime is your time to "hack", solve unique problems, and show that you can work hard on the problem at hand. Typically, your lab time will prepare you to work on your next programming assignment. You must be in class to get credit for the session. If you happen to miss a class, you are strongly encouraged to complete the in-class work on your own time, but please do not ask for credit.

(1) In order to best grasp the material presented in lab, I strongly suggest that you completely redo any labs that you find difficult. (Read that last sentence again, unless of course you've already reread it once).

Quizzes

Your four quizzes will test your comprehension of Python scripts as well as your ability to write your own Python, regular expressions, and XHTML. During lecture, I will give "hints" of what a typical test question might be; take good notes! There will be no make-ups, nor will the lowest quiz be dropped. If you are an athlete and/or you have a conflict with a quiz date, please see me within the first week of classes. (Note: Quiz #2 is on the Friday before Spring Break; please don't make travel plans until after our quiz).

HELP

I have listed my office hours on the syllabus at the top. But I'm usually near a keyboard so we can schedule alternate times to meet. Study, study, study and talk about it with me and others as often as you can. *Please don't wait too long before you see me; a quick chat in my office can often clear things up. I'm here a lot...*

Week	**Topic (more details and links via moodle (onCourse))** Note: At the start of each lecture, I will write on the board the exact readings (pages and/or URLs) that are associated with the material covered in this class meeting.
1	**Jan 27/29** Intro to the web, (X)HTML, and our Linux Apache server (Dreamweaver, WinSCP, putty, and W3C validation)
2	**Feb 01/03/05** more X(HTML), CSS, and JavaScript working in hexidecimal and Unicode
3	**Feb 08/10/12** Prof. Drout (English) guest lecture Intro to Python: working with the string API Python output as XHTML (Python IDE: Wing)
4	**Feb 15/17/19** Reading files; handling a corpus of files with glob Iterating over lists Introduction to Regular Expressions ("regex") (searching for patterns in MS Word; practicing regex with the RegexPlay .cgi http://cs.wheatoncollege.edu/regexplay)
5	**Feb 22/24/26** Working with conditional expressions if: else:
6	**Mar 01/03/05** Working with repetitional control: "loops"; More with lists
7	**Mar 08/10/12** Prof. Drout guest lecture Review of results from the "tall" … "elf" experiment
8	**Spring Break** – Monday, March 15 – Friday, March 19 – **Spring Break**
9	**Mar 22/24/26** Introduction to Python "dictionaries" (hash tables) Intro to XML (XML Copy Editor)
10	**Mar 29/31** and **Apr 02** Intro to the Text Encoding Initiative (TEI) Guest lecture by Patrick Rashleigh, Faculty Technology Liaison ("web geek")
11	**Apr 05/07/09** Marking-up JRRT Intro to XPath (via ElementTree) in Python
12	**Apr 12/14/16** Text mining Anglo-Saxon poems (including Scott Kleinman's lemmatized versions) Wheaton founder Eliza Baylies Chapin Wheaton and her (now TEI) pocket diaries Guest lecture by Prof. Kathryn Tomasek (History)
13	**Apr 19/21/23** Introduction to stylistics and authorship attribution, e.g., John Burrows' DELTA method Visit to College Archives to view physical pocket diaries: College Archivist Zeph L. Stickney
14	**Apr 26/28/30** Final Project Experimental Design: methods
15	**May 03/05/07** Final Project Oral Presentations Wrap-up - Evaluations