LING 300 - Topics in Linguistics:
Introduction to Programming and Text Processing for Linguists

Week 1

Intro,
Unix, Shell,
Environment, Files
Who is this class for?

- Linguists, social scientists, humanists
- Little-to-no programming experience
- Applications to research

Goals

- Lots of hands-on practice
- Teach you how to teach yourself
Who is this class not for?

- Folks with lots of programming experience
- CS Majors (probably - email me if this is you)
- COMP_SCI 110 is similar in focus (and uses one of the same textbooks) - what’s different?
  - CS110 - broad, more CS-y (e.g. debugging and testing)
  - LING300 - narrow focus on applications to text, we will purposefully skip less-relevant stuff
What will we learn?

- Unix Command Line
  basic usage, remote access, and tools for text

- Basic Python
  programming concepts, syntax, useful libraries for text

- Applications (as much as we have time)
  web scraping, APIs, data munging, text analysis
When and where will we see each other?

Zoom at normal class times (optional but recommended)
short lecture (likely usually only for Monday class)
recorded if you can’t make it

Office hours - Monday 5-6pm, Tuesday noon-1pm
hangout room, with breakout for individual questions

Piazza discussion board for questions
help each other out!
Why are we doing this?

1. Get computationally “free” - GUIs only let you do things someone else decided on

2. Processing text data is useful for anyone’s research

3. This is the start of computational linguistics! web search, speech-to-text, conversational AI, “big data” language analysis, etc etc
How will we do it?

Syllabus on course website:  
http://faculty.wcas.northwestern.edu/robvoigt/ling300/

Assignments, peer review, final project

Videos/readings before class, working on assignments during

Graded on effortful completion, self-evaluation

(Universal pass/fail this quarter!)
The Struggle!

Learning programming is like learning a new language
You have to soak in it and use it daily
It will feel unnatural at first, push through
Don’t be afraid to play around and break stuff
The Struggle Illustrated
YOU CAN DO IT

ERRORS ARE YOUR NEW FRIENDS

No such thing as a dumb question here.
Our new home: the command line
Precision - the challenge of exactitude

One wrong letter, space, or punctuation mark can easily derail you.

These mistakes are at first very hard to see.

Double-check, triple-check your code and relevant documentation.

Take a break and come back to it.
Benefits of command line interfaces

Automatable

easy to do

Fast

GUI interfaces are computationally ‘heavy’

Consistent

same command always does the same thing

Transparent

you’ll learn what your files actually are
What is a file?

An abstraction!
... but ultimately, an array of bytes

e.g., for ASCII text:

<table>
<thead>
<tr>
<th>Character</th>
<th>L</th>
<th>I</th>
<th>N</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>100 1100</td>
<td>100 1001</td>
<td>100 1110</td>
<td>100 0111</td>
</tr>
</tbody>
</table>
Types of Files

Text
bytes representing characters
txt, code (like .py), html, logs

Executable
compiled code in binary format
to run as a program

Data
everything else: images, zip files,
doc/ppt/pdf, and so on
**Quest!**

Remote computing environment, cluster of computers running Linux

Common for “big data” and high-performance tasks

Can schedule complex stuff, not waste your own machine

Original plan was to use Quest exclusively

If it is slow because of where you are, you can do everything locally, then upload assignments

```
scp assignment.txt [netid]@quest.it.northwestern.edu:/projects/e31086/user/[netid]/week1/
```