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

Week 5

Basic Python (More Assignment 3 Notes)
Later this Week

- Survey:
  - Midterm self-evaluations
  - Midterm course feedback
  - Final project ideas?

- Final project note:
  - Now there will be a default assignment
  - But it will be much more self-directed than usual
Notes from Assignment 3

All Assignment 3s graded on Quest,
[netid]/week3/assignment_graded.py

In-line comments as usual:

### [RV] Blah blah blah
Notes from Assignment 3

- `Department[1532:]`?
- `and` (boolean ‘and’) vs. `&` (bitwise ‘and’) - my bad!
  - Use `and` for boolean comparisons (short circuiting ✅)
  - `&` also has higher precedence (can be confusing)
- `for line in open(f)` Does not strip whitespace!
  - If you got 5-letter palindromes using `min_length`, this is because each line has ‘\n’ on the end!
Notes from Assignment 3

- There’s a near-infinite variety of ways to do most things.
- **Example:** reverse_string
  - `s[::-1]`
  - `l = list(s), while len(l) > 0, l.pop()`
  - `l = list(s), l.reverse(), ', '.join(l)`
  - `i = len(s) - 1, while i > 0, i -= 1`
  - `new_s = '', for c in s, new_s = c + new_s`
Notes from Assignment 3

- Efficiency: not a huge deal for now, but be aware! e.g. consider how many times we loop over what

  Which is better?

  for word in s.split(): vs. for word in stopwords:
  if word in stopwords: if word in s.split():

- Anti-corollary: “Don’t optimize prematurely”
  Doing it whichever way is fine, until it gets too slow to work
**Style Notes from Assignment 3**

- Standards? Somewhat, e.g. style guide: [https://www.python.org/dev/peps/pep-0008/](https://www.python.org/dev/peps/pep-0008/)
- Opinions? Many!

- Key consideration is **readability**.
  - Other people may have to read your code
  - You may have to read your own code in five years
Style Notes from Assignment 3

- Readability Basics:
  - # comments are good practice to explain the purpose and functionality of more complicated bits
  - The best code is also somewhat “self-documenting”
  - Variable names are a form of comment
  - Logical decomposition helps readability
Style Notes from Assignment 3

- Consider:

  \[
  a = \text{sum}(\text{vals}) \\
  b = \text{len}(\text{vals}) \quad \text{vs.} \quad \text{return } \frac{\text{sum}(\text{vals})}{\text{len}(\text{vals})} \\
  \text{return } a/b
  \]

  \[
  \text{length1} = \text{len}(s1) \\
  \text{length2} = \text{len}(s2) \quad \text{vs.} \quad \text{if } \text{len}(s1) > \text{len}(s2): \\
  \text{if } \text{length1} > \text{length2}: \quad \ldots \\
  \ldots
  \]
Style Notes from Assignment 3 (cont.)

- Variable naming: try not to overload (one name does one thing)

```
# file object
document = open(f)

# string
document = document.read()

document = letters_only(document)  # string

document = document.split()  # list
```

vs.

```
# file object
document = open(f)

text = letters_only(document.read())  # string

words = document.split()  # list
```
Style Notes from Assignment 3 (cont.)

- Variable naming: try not to overload (one name does one thing)
  - Special case of this: `\texttt{.join()}`
    - `\texttt{output = ' '}`
    - `\texttt{output = output.join(words)}`
  - Both `\texttt{output's are strings, but they're different - first is the delimiter, second is the actual output. Just do: \texttt{output = ' '.join(words)'}}`
Advanced Syntactic Sugar

- **List Comprehension**
  
  ```python
  output = ' '.join([c for c in s if c.isalpha()])
  ```

- **Ternary Conditional Assignment**
  
  ```python
  x = 0 if random.random() > 0.3 else 1
  ```

- **Step slicing:**
  
  ```python
  my_string[start:end:step]
  ```