LING 331
Text Processing for Linguists

Week 7

Jupyter and Basic Python 3
Today: Code Review! A3 and A4

- Choose a round-robin cycle
  - A checks B’s, B checks C’s, C checks A’s
- Start by going to your partner’s directory and doing:
  
  ```bash
  cp assignment4.py assignment4_peer.py
  ```

- Edit `assignment4_peer.py`, make comments like I do:
  
  ```
  ### [AE] This is great!
  ```

- ~30 mins review, ~15 mins discussion
Today: Code Review!

- Questions to ask yourself:
  - Does this work?
  - Do I understand why it works?
  - What could be better? (style, efficiency, readability)
  - Compare to our grading and/or solutions
Jupyter! - Live Assignment 5 Demo

Basic steps:

● `wget` assignment link into a Quest `assignment5` directory

● Do `unzip` `assignment.zip`

● Go to https://jupyter.questanalytics.northwestern.edu
  (must be on NU VPN)

● Navigate to your `assignment5` dir
  and open ‘Assignment 5.ipynb’
Writing Files

- With a file path as a str \( f \), we’ve seen `open(f)`
- `open` takes a `mode` argument which explains how to open it
  - Actions:
    - `'r'` to read (default) like Unix `<`
    - `'w'` to write (to a new file) like Unix `>
    - `'a'` to append (add to existing file) like Unix `>>`
  - Formats:
    - `'t'` for text (default)
    - `'b'` for binary

  Action and format can both be included and are both optional
Writing Files

- **Write using the** `.write()` **method on a file object.**
- **Say given a** Counter **of word counts in some text**

```python
file = open('output.txt', 'w')  # creates/overwrites
for word in counts:
    line = "{}, {}".format(word, counts[word])
    file.write(line + '
')  # must be str
file.close()  # makes sure everything is written
```

- **Unlike print,** `.write()` **only takes one argument, a string**
JSON (Javascript String Object Notation) provides a way to save objects as text

- Say given our dictionary variable `cmudict`

```python
import json
json.dump(cmudict, open('cmudict.json','wt'))
```

Later, or in another script:

```python
cmudict = json.load(open('cmudict.json','rt'))
```
JSON (Javascript String Object Notation) provides a way to save objects as text

● Can also just convert them to strings:

```python
json.dumps(cmudict)
'
{"3-D": ["TH R IY1 D IY2"], "3D": ["TH R IY1 D IY2"], "A": ["AH0", "EY1"], "A\'S": ["EY1 Z"], "A.": ["EY1"], "A.\'S": ["EY1 Z"], "A.S": ["EY1 Z"], "A42128": ["EY1 F AO1 R T UW1 W AH1 N T UW1 EY1 T"], "AA": ["EY2 EY1"], "AAA": ["T R IH2 P AH0 L EY1"], "AABERG": ["AA1 B ER0 G"], "AACHEN": ["AA1 K AH0 N"], "AACHENER": ["AA1 K AH0 N ER0"], "AAH": ["AA1"], ...
```
Pickle provides a way to save objects in binary

- Say given our dictionary variable `cmudict`

  ```python
  import pickle
  pickle.dump(cmudict, open('cmudict.pkl','wb'))
  
  Later, or in another script:
  
  cmudict = pickle.load(open('cmudict.pkl','rb'))
  ```
JSON vs. Pickle

- Saved as plaintext (easy to open and look at)
- Can even be edited directly outside python (carefully)
- Compatible with many other programming langs
- Some objects are not JSON serializable, e.g. set

- Not human readable
- Python-only
- Slower (generally)
- But works on almost any object

*Takeaway*

Use JSON unless you can’t.
FYI, Jupyter notebooks are in JSON format!