

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

Week 6

—

Jupyter and Basic Python 3

Notes on Assignment 4

- Using *flags* (like `remove_blank`):
 - “flags” are arguments that give options rather than data
 - Try to have core functionality only be written once; helpful if you ever need to change anything
- `letter_counts` - no need to tokenize, loop over words etc:
 - Can simply do `for character in s`
 - Remember strings are sequences

Notes on Assignment 4

- You can use `random.random()` in a conditional directly rather than saving it in a variable that you only use once

```
if random.random() > 0.5:
```

- Avoid **hardcoding**: e.g., in the dice sums problem:

```
sum_counts = {0: 0, 1: 0, 2: 0, 3: 0, 4: 0...
```

Notes on Assignment 4

- `string.split()` splits in a greedy way,
e.g. maximum amount of whitespace

- What's the difference?

`s.split()` vs. `s.split(" ")`

Notes on Assignment 4

- Variable naming:
try to have names reflect the contents/purpose

- Which is better?

```
for word in line.split()
```

or

```
for words in line.split()
```

Decomposition

Breaking down
an abstract problem
into smaller parts
we can handle

How to draw an Owl.

"A fun and creative guide for beginners"

variables
loops
conditionals
functions
methods
modules



*Who rhymes
more often,
Beyonce or
Taylor Swift?*

Fig 1: Draw two circles

Fig 2: Draw the rest of the damn Owl

Question-Answer pair worked example

Notes on Assignment 4

- Style point: make objects what we will use them for

- e.g., `proportion_of_oneoff_types`

Accumulate counts on an integer

vs.

Accumulate a list of oneoff types and get its length

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

```
file = open('output.txt', 'w') # creates/overwrites
for word in counts:
    line = "{}, {}".format(word, counts[word])
    file.write(line + '\n') # 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`

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

Later, or in another script:

```
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:

```
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`

```
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!

```
rfj5679@quser21:~
{
  "cells": [
    {
      "cell_type": "markdown",
      "metadata": {},
      "source": [
        "## Welcome to Assignment 5!\n",
        "\n",
        "And welcome yet again to a new world in which
        mming. We started with the command line, moved on
        diting `.py` files with our code using command-lin
        s, and now we're here in Jupyter-land. This file i
        ed a `Jupyter notebook`, a user-friendly and hig
        document that allows us to not only write code but
        y see the outputs in this web browser editor appli
        cool!\n",
        "\n",
        "\n",
        "So this is another transition but hopefully o
        your life easier rather than harder. We're very l
        Quest infrastructure has an easy setup for Jupyter
        ready, so we can essentially just go to a URL in
        , log in, and directly access our code and files.
        resources for this week on the course website for
        tion about Jupyter. \n",
        "\n",
        "\n"
      ]
    }
  ]
}
```