LIN 127
Text Processing and Corpus Linguistics

Week 6

—

Assignment 4 Pre-review, Decomposition Exercises

Notes on Assignment 4 - Abstraction

- Here and continuing on -
  make use of things you've already done!
  - I try to note this in the problem description frequently

- E.g. `word_counts` - the docstring says "Tokenize the str `s` and accumulate counts for each word that appears in a dictionary."
  - Yet I see folks not using their tokenize function!

Notes on Assignment 4 - Scope

- Reminder about *scope*
  - Question: `run_tests` uses a `val` variable, can this be accessed elsewhere?
  - Answer: No, it's only "alive" within `run_tests` because it has local scope

Notes on Assignment 4 - Scope

- Concept of “namespace”
  - The set of “currently defined variables”
  - Different if we’re talking about built-in / global / local
  - Common question: “what’s in the namespace here?”

Notes on Assignment 4 - Scope

- Careful importing too much directly!
  The namespace can get crowded:
  - ```
    from random import random      # problem?
    randval = random()
    ```
- The cardinal sin!
  - ```
    from random import *
    ```
  - Now it's not even documented in the file anywhere what's in the namespace.

Notes on Assignment 4 - Randomness

- For random_word_generator, what's the problem?

```
if random.random() < 0.3:
    return V
elif random.random() < 0.4:
    return CVC
elif random.random() < 0.15:
    return CV
else:
    return VC
```

- The above actually become conditional probabilities!

Notes on Assignment 4 - Randomness

- Instead:

```
randval = random.random() # throw the dart once
if randval < 0.3:          # 30% probability
    return V
elif randval < 0.7:        # 40% probability
    return CVC
elif randval < 0.85:       # 15% probability
    return CV
else:                      # 15% probability
    return VC
```

Notes on Assignment 4 - Randomness

- However, if only using for one check (e.g. Q&A, monkeys):

```
if random.random() < 0.1:
    # do the thing
```

This is the most common use.

Notes on Assignment 4 - Splitting and Joining

- 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 - Splitting and Joining

- `monkeys_typing_shakespeare`, how to build the answer?

  String you keep adding to?

  vs.

  List of words you later join?

- I vote the latter, then adding punctuation is easy in the loop:

```
if random.random() < 0.1:
    words[-1] = words[-1] + random.choice(',.!?')
```

Notes on Assignment 4 - Efficiency

- Reading files:
  - use `for line in open(f):`
    unless you have a good reason not to
  - Doing so doesn't open the file all at once. 👍
    (what if your file is 100 GBs large?)

Notes on Assignment 4 - Efficiency

- `letter_counts` - no need to tokenize, loop over words etc (docstring has a typo on that actually...)
  - Can simply do `for character in s`
  - Remember strings are sequences

Notes on Assignment 4 - Efficiency

- Efficiency! Sometimes hard to spot. Where's the problem?

```
words = []
for line in open(f):
    tokens = tokenize(line)
    for token in tokens:
        if token in words:
            continue
        else:
            words.append(token)
return len(words)
```

Notes on Assignment 4 - Efficiency

- `if token in words:`
  - If `words` is a list, this has to do a sequential check through the entire list every time this is called.
    - Number of operations = size of list
  - If `words` is a set, this is an instantaneous operation, due to a nice thing called hashing
    - Number of operations = 1 (roughly)

Notes on Assignment 4 - Style

- 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

Notes on Assignment 4 - Style

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

- Which is better?

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

or

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

Notes on Assignment 4 - Style

- For `wc_lines` how best to do this?
  - Accumulate a count!
  - This is stylistically good because it matches the concept of the function - we are trying to ultimately obtain a count

Notes on Assignment 4 - Style

- Related 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?

Notes on Assignment 4 - Style

- 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 - Style

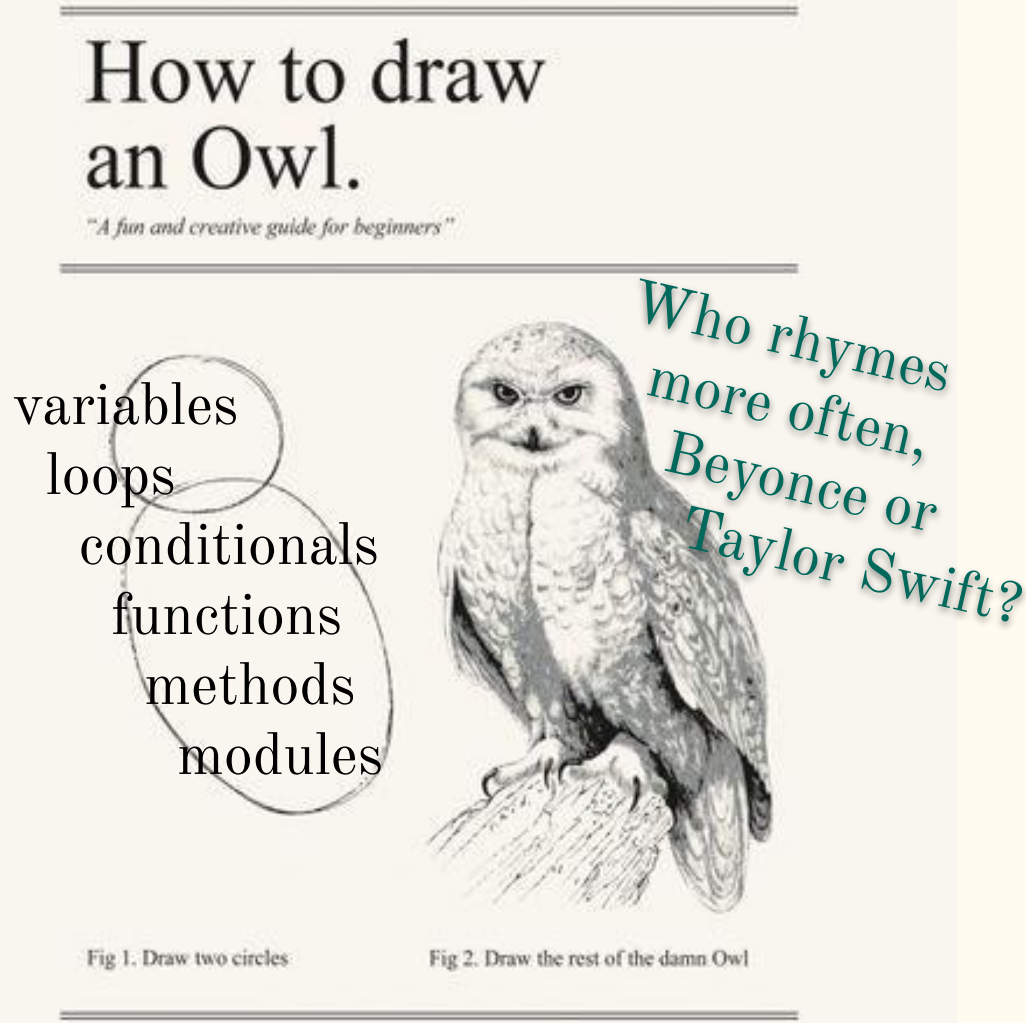
- Remember you can chain operations:
  - ```
    plain = s.strip()
    lower = plain.lower()
    list = lower.split()  # also list is not a
    for word in list:     # good var name
    ```

    vs.

  - ```
    for word in s.strip().lower().split():
    ```

Decomposition

Breaking down
an abstract problem
into smaller parts
we can handle



Question-Answer pair worked example

If time:

Anagram Finder worked example

Jupyter! - Live Assignment 5 Demo

Basic steps:

- `wget` assignment link into a new `a5` directory
- Do `unzip a5.zip`
- Install Jupyter Notebook (`pip install notebook`)
- Run `jupyter notebook` to access in your browser, open `a5.ipynb`