Regular Expressions Practice

Group Members and Roles

- Rashi (driver)
- David (reviewer)

Purpose

In this discussion activity, we'll the use of regular expressions to parse an HTML script of a play. Our ultimate aim will be to count the number of lines spoken by each character in the play.

While there are significant limitations to parsing HTML with regex, some discussed in this memorable StackOverflow post (https://stackoverflow.com/questions/1732348/regex-match-open-tags-except-xhtml-self-contained-tags/1732454#1732454), for our purposes we will do just fine.

Hamlet

David Tennant as Hamlet, a man of many feelings.

William Shakespeare's *Hamlet* is a famous English play in which the characters have lots of feelings about lots of things. The original script is here (http://shakespeare.mit.edu/hamlet/full.html).

(A). Read Data

We are going to directly read HTML from the site on which the script is hosted. To load it into your computer with variable name `script`, run the following code block. You will need to be connected to the internet in order for this to work.

In [1]:
```
import urllib

url = "http://shakespeare.mit.edu/hamlet/full.html"

# retrieve the data
filedata   = urllib.request.urlopen(url)
# read as bytes
bytes_text = filedata.read()
# decode bytes to string
script     = bytes_text.decode("utf-8")

# inspect the first 500 characters
script[0:1000]
```

Out[1]:
```
'<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"\n "http://www.w3.org/TR/REC-html40/loose.dtd">\n <html>\n
<head>\n <title>Hamlet: Entire Play\n </title>\n <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-
1">\n <LINK rel="stylesheet" type="text/css" media="screen"\n       href="/shake.css">\n </HEAD>\n <body bgcolor="#fffff
f" text="#000000">\n\n<table width="100%" bgcolor="#CCF6F6">\n<tr><td class="play" align="center">The Tragedy of Hamlet,
Prince of Denmark\n<tr><td class="nav" align="center">\n      <a href="/Shakespeare">Shakespeare homepage</A> \n    | <A
href="/hamlet/">Hamlet</A> \n    | Entire play\n</table>\n\n<H3>ACT I</h3>\n<h3>SCENE I. Elsinore. A platform before the
castle.</h3>\n<p><blockquote>\n<i>FRANCISCO at his post. Enter to him BERNARDO</i>\n</blockquote>\n\n<A NAME=speech1><b>
BERNARDO</b></a>\n<blockquote>\n<A NAME=1.1.1>Who\'s there?</A><br>\n</blockquote>\n\n<A NAME=speech2><b>FRANCISCO</b></
a>\n<blockquote>\n<A NAME=1.1.2>Nay, answer me: stand, and unfold yoursel'
```

(B). Extract Character Names

We would like to extract from this rather messy looking HTML a list of characters in the play. To do so, we are going to match all instances of the character name occurring above their dialogue. For example:

HAMLET

> Not so, my lord; I am too much i' the sun.

This passage indicates that Hamlet is the character who speaks the line. We would like to capture strings like **HAMLET** using regular expressions.

These names are enclosed in HTML `<a>` tags, with a special variable `NAME=speechNUM` (where `NUM` can be any positive integer). Within the `<a>` tags are also `<b>` tags. In some cases, the first `A` in the opening HTML tag may be capitalized. Here are four examples:

- `<A NAME=speech1><b>BERNARDO</b></a>`
- `<A NAME=speech2><b>FRANCISCO</b></a>`
- `<A NAME=speech34><b>KING CLAUDIUS</b></a>`
- `<A NAME=speech197><b>First Ambassador</b></a>`

Write a regular expression called `pattern` that (a) matches the complete string in each example above and (b) captures the name itself within a group. You should not capture any of the additional HTML tags. Feel free to take the list of examples above, and paste it in Pythex (https://pythex.org/) to experiment with your expression. Once you've settled on your solution, import the `re` module and use `re.findall` to obtain all character names. At this stage, you should have repeats: for example, there are 23 distinct speeches by the character `BERNARDO`. Save your results as a list called `speeches`.

```
We were having big issues with our final project, and we spent all of discussion talking about our final project with
Hinal. Thank you hinal <3
```

Check that you can replicate the following:

First check:

```
len(speeches)
```

```
1148
```

Second check:

```
speeches[0:5]
```

```
['BERNARDO', 'FRANCISCO', 'BERNARDO', 'FRANCISCO', 'BERNARDO']
```

In [ ]:
```
# first check
```

In [ ]:
```
# second check
```

(C). Getting Speeches with Lines

We would now like to compute the number of lines assigned to each character in the play.

Run the following code block, which will add a new expression to your `pattern` from before.

In [ ]:
```
pattern2 = pattern + r'\n<blockquote>\n((?:.|\n)+?)\n</blockquote>'
pattern2
```

Spend a few minutes trying to understand `pattern2`. How does it work? What does `|` do? What about `.`? Reviewer, feel free to do a bit of research here and help out your partners. Then, write a brief explanation below. It's fine to run the code below and then come back to the explanation if you're not sure.

Your Explanation

In [ ]:
```
# write in here
```

Now, add the the pattern above immediately after your pattern from Part (B). Again use `re.findall()` to construct a list of matches. Call it `speeches_with_lines`.

In [ ]:
```
# grab matches here
```

Check that `speeches_with_lines` has *nearly* the same length as `speeches`. It will be slightly shorter due to a few HTML irregularities in the source material. No worries!

In [ ]:
```
# check length here
```

(D). Counting Lines

Examine `speeches_with_lines` and note its structure. Write a function `count_lines` which, when applied to a single element of `speeches_with_lines`, gives the number of lines spoken by that character. For example, in the source material, the following speech has two lines.

HAMLET

> I am glad to see you well:
> Horatio,--or I do forget myself.

Hint: in HTML, distinct lines are separated by the tag `<br>` .
Hint: `"cat cat cat".count("cat")`

In [ ]:
```
# your function here
```

Next, create a list `L` of tuples of the form `(character, length_of_speech)` using your new function. There should be one tuple for each speech, so the same character will occur multiple times. This can be done in a single line via a list comprehension. The first 10 entries should be:

```
[('BERNARDO', 1),
 ('FRANCISCO', 1),
 ('BERNARDO', 1),
 ('FRANCISCO', 1),
 ('BERNARDO', 1),
 ('FRANCISCO', 1),
 ('BERNARDO', 1),
 ('FRANCISCO', 2),
 ('BERNARDO', 1),
 ('FRANCISCO', 1),
 ...
```

In [ ]:
```
# create your list of tuples
```

If you've made it this far, you've done well. If there are fewer than 10 minutes left in Discussion, feel free to wrap up and submit the assignment. Otherwise, please continue on to part (E).

(E). Aggregate

Create a dictionary called `lines_dict` whose keys are the characters and whose values give the total number of lines spoken by that character. Which characters speak the most lines?

If you still have time, find a way to display the results in descending order by number of lines.

In [ ]:
```
# construct and show lines_dict
```

In [ ]:
```
# show a sorted version (might not be a dict)
```