Quiz 1: Web Mining with Beautiful Soup

In the early days of the web, most parsers could only parse well-formed XML and HTML. The poorly-formed stuff you saw on the Web was referred to as "tag soup", and only a web browser could parse it.

Beautiful Soup started out as an HTML parser that would take tag soup and make it beautiful, or at least workable. Beautiful Soup takes its name from the poem *Beautiful Soup* from Alice's Adventures in Wonderland and is a reference to the term "tag soup" meaning poorly-structured HTML code. The latest documentation of Beautiful Soup is here.

Goal

To use Beautiful Soup, the Python Library, to extract some information from the internet about Beautiful Soup, the Lewis Carroll poem.

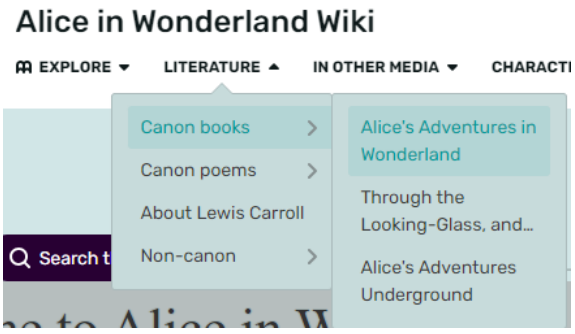
From the Welcome to Alice in Wonderland wiki page, extract

- The list of Canon book names,
- The list of Canon poems, their titles, and texts.

Understanding the data

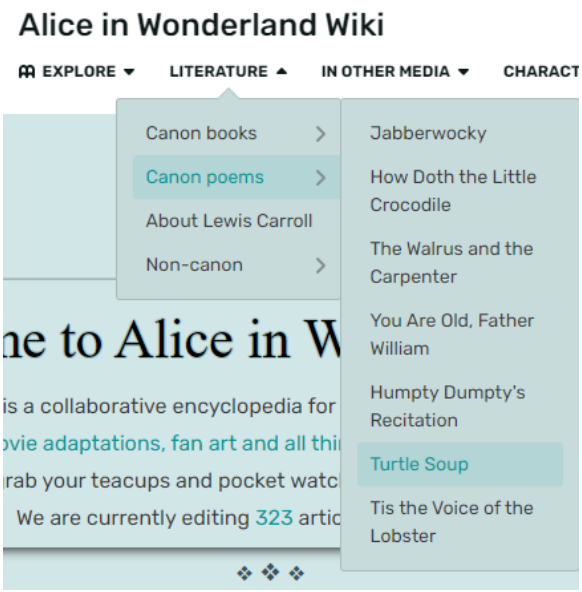
Zero points for this, the objective is to get a feel for the data.

1. Visit the wiki page.
2. The canon books are shown under **Literature**, like this[1]:



[1] Showing that the canon books are Alice's Adventures in Wonderland, Through the Looking Glass, and Alice's Adventures Underground.

3. The canon poems are also shown under **Literature** and can be seen like this:



Programmatic Extraction

Reminder: Use "View Page Source" of the web page to view HTML of the page. Use "Inspect" to understand the HTML under a particular tag. If possible, please use Google Colab (or another Jupyter environment) as your programming environment as it allows for mixing code and text in the same file. Use a relatively recent version of Beautiful Soup with either "html.parser" or "lxml" parser.

1. Write a Python program **`books()`** to return the names of the canon books[2].
2. Write a Python program **`poems()`** to return the names of the canon poems *and* their URLs[3].

[2] We already know the correct answer. It is:

```
["Alice's Adventures in Wonderland",
 'Through the Looking-Glass, and What Alice Found There',
 "Alice's Adventures Underground"]
```

[3] We already know the correct answer. It is:

```
[('Jabberwocky', 'https://aliceinwonderland.fandom.com/wiki/Jabberwocky'),
 ('How Doth the Little Crocodile',  'https://aliceinwonderland.fandom.com/wiki/How_Doth_the_Little_Crocodile'),
 ('The Walrus and the Carpenter',  'https://aliceinwonderland.fandom.com/wiki/The_Walrus_and_the_Carpenter_(poem)'),
 ('You Are Old, Father William',  'https://aliceinwonderland.fandom.com/wiki/You_Are_Old,_Father_William'),
 ("Humpty Dumpty's Recitation",   'https://aliceinwonderland.fandom.com/wiki/Humpty_Dumpty%27s_Recitation'),
 ('Turtle Soup', 'https://aliceinwonderland.fandom.com/wiki/Turtle_Soup'),
 ('Tis the Voice of the Lobster',   'https://aliceinwonderland.fandom.com/wiki/Tis_the_Voice_of_the_Lobster')]
```

3. Write a Python program **`poem_title_text(n)`** that calls **`poems()`** and, using `n` as the index and returns the poem title and its text[4].

What to submit[5]:

The files `quiz_1.ipynb` and `quiz_1.pdf`

[4] Your program is only required to work for n = 5 (Turtle Soup) and n = 2 (The Walrus and the Carpenter). The other poems don't follow the same structure.

[5] A bug in Google Colab may get in the way of outputting PDF files. Following the suggestions here, please place this fragment as the first executable cell in the notebook

```
from google.colab import drive
drive.mount('/content/drive', force_remount=True)
!apt-get install texlive texlive-xetex texlive-latex-extra pandoc > /content/drive/MyDrive/installs.txt
!pip install pypandoc > /content/drive/MyDrive/pandoc_installs.txt
```

And this fragment as the last executable:

```
!cp "./drive/My Drive/<location of your .ipynb file>" ./quiz_1.ipynb
!jupyter nbconvert --to PDF "quiz_1.ipynb"
```