# Assignment : Python

Your task for this practical assignment consists of two parts:

1. Create a Python script that performs a statistical analysis of a text document and provides functionaliy as stated in the Requirements section below.
2. Make the Python script that you have created accessible and usable via the URL

**http://. . . . . ./<your user name>/words.py**

**Requirements**

1. The script should display a web page that contains a form with two text fields and a `Submit' button. The first text field should allow a user to enter a single URL. You do not need to check whether the URL is syntactically well-formed.

If a user enters a URL into the first text field and presses the `Submit' button, then your script should retrieve the document that the URL points to. You should cater for the possibility that the URL is not valid, i.e., that there is nothing to retrieve at that URL, indicate an error to the user in such a case, and allow the user to start again.

The second text field allows the user to directly enter the text of a document and must be big enough for a reasonable amount of text (see the test cases for how big documents might be). If a user presses the `Submit' button, then your script should simply take the text as the document that needs to be processed.

A user may enter both a URL into the first text field and text into your second text field and then press the `Submit' button. In such a case your script should indicate an error and allow the user to start again.

A user may also just press the `Submit' button without entering anything into either of the two fields. Then the script should indicate an error and allow the user to start again.

2. The document may not only ASCII characters but also UTF-8 characters and your script should handle those characters correctly when performing the analysis.

3. Given the document, you should determine the total number of word occurrences in the document and also count the occurrences of each word in the document. The count should be stored in a Python dictionary.

In determining the words that occur the document, your script should comply with the rules below. These rules may look complicated, but are intended to make your life easier while establishing agreement on what a word is.

**i.** Only sequences of characters that (i) only consist of ASCII letters, ASCII digits, apostrophes, hyphens, and underscores, (ii) start with a letter or start with an apostrophe followed by a letter, and (iii) end with (a) a letter, (b) a digit, or (c) an apostrophe preceded by the letter s, should be considered to be words. Any character that cannot occur in a word should be considered to be a word delimiter.

For example, **10**, **123-reg**, **even-**, **evenin'** are not words, but **p0wned**, **l33t**, **hello\_world**, **reg-123**, **ClickMe**, **'twas**, **o'clock**, and **has'** are words.

The character sequence **$1.2bn** contains two word delimiters, **$** and **.** separating this sequence into **1** and **2bn**, neither of which is a word.

The character sequence **six%two** contains the word delimiter **%** separating this sequence into **six** and **two**, both of which are words.

**ii**. HTML comments should be removed and HTML tags and words within HTML tags should not be counted. For example, the HTML fragment

**<!-- This is a comment --> <a href="url">some words</a> <ul type="disc"><li>one</li> <li>two</li></ul>**

should for the purposes of the word count become the text

**some words one two**

Remember that HTML comments can span several lines, but cannot be nested.

iii. Words containing orthographic or grammatical hyphens should be considered and counted as one word, for example, **sign-writer**, **easy-to-use**, **non-disposable**, **honest-to-goodness**, and **catch-22** each are a single word. You can assume that the document does not contain any end-of-line hyphens, so

**has-**  
**been**

consists of the non-word **has-** and the word **been**, not the word **has-been**.

iV. The document may contain em-dashes represented by **--** (two consecutive hyphens/dashes) and/or **---** (three consecutive hyphens/dashes). Em-dashes should not be confused with orthographic or grammatical hyphens. For example, in the sentence

**It's that time of year again--time for New Years Resolutions.**

**again** and **time** are separate words.

V. Contractions count as one word and should be counted separately from the corresponding non-contracted words. For example, **doesn't**, **there's**, **I've** and **you'll** should each count as one word and they are counted separately from **does**, **not**, **there**, **is**, **I**, **have**, **you** and **will** (this is not particularly logical, but obviously much easier to program).

Vi.Possessive forms count as one word and are counted separately from the corresponding nouns. For example, **Peter's**, **Moses'**, **business's** each count as one word and they are counted separately from **Peter**, **Moses** and **business**, respectively. Similarly, **boys** and **boys'** are different words.

Vii.The document may contain quotations and/or quoted speech. Quotation marks may be represented by **"** (quotation mark), **``** (two grave accents), or **`** (one grave accent).

Viii.All words should be converted to lower-case before being counted. For example, both **Hotel** and **hotel** are counted as **hotel**.

4. Once you have completed the word count, your script should produce a HTML page that includes

* a statement of the URL that the user has entered (if one was entered);
* the document that was processed, visibly showing any HTML markup that it may have contained;
* a statement of the total number of word occurrences in the document;
* two HTML tables, the first showing the ten words occurring most often and the number of occurrences for each of these words (listed in order of number of occurrences with the word occurring most often listed first) and the second showing the ten words occurring least often and the number of occurrences for each of these words (listed in reverse order of number of occurrences with the word occurring least often listed first).

Each table should have two columns, one for the words, one for the number of occurrences and one row for each word. The columns should have appropriate headings and the tables should have appropriate headings.

You are permitted to use Python's built-in **sort** function to produce those tables.

This HTML page should be displayed to the user as response to the submission of the form.

5.Your code should follow the COMP519 Coding Standard. This includes pointing out which parts of your code have been developed with the help of on-line sources or textbooks and references for these sources.

## Test data

Test data, together with the expected results, can be found at test2017

## Submission

Submit your Python script via the departmental submission system. Do not forget to also make words.py accessible via the departmental web server.

## Assessment

This practical assignment will address the following learning outcomes of the module:

* be able to make informed and critical decisions, design and implement reasonably sophisticated server-side web applications using one or more suitable technologies.
* be able to demonstrate an understanding of the range of technologies and programming languages available to organisations and businesses and be able to choose an appropriate architecture for a web application

Marks will be awarded according to the following scheme:

* The Python script is accessible via the required URL, works without producing script errors, all required files were submitted, the files accessible via the web are identical to those that were submitted, and the access rights of the files in your filestore must be such that no other user can view their contents in the filestore: 10
* Input handling and retrival of document by URL: 16
* Word count operations: 47
* Creating HTML page with word count statistics: 15
* Formatting, commenting, and quality of code: 12