# W8 Assignment parts 1 & 2

The final two assignments are a software development project.

I recommend that they are both about the same project idea as that will be less work for you and will almost certainly allow you more opportunities to explore your own interests.

In the first part of the assignment you will define the project, explain the topics/questions you intend to explore, and give outline details about how you intend to use Python to explore your topic computationally.

In the second part of the assignment you will write the programs you need, collect the data resources you will explore, and write a project report summarising your work.

**IMPORTANT**: Your assignment will be judged and marked as a software development project. It will not be judged on the linguistic concepts you propose to investigate

-[ ] text

### Project Part 1

A project development proposal describing the ideas and objectives you intend to explore and an outline of the software you will build. As well as descriptive text, the document you submit for this assignment can include diagrams, lists, sketches, and so on. It should include the following sections:

1. The project title
2. A description of the project and the question it will explore
3. An outline of how you intend to approach the project:
   1. The data and text resources you will need
   2. Where you will get free access to sufficient quantities of data?
   3. The functionality of the application you will create, perhaps with a tentative list of the functions you will build
   4. How you will you test your software to confirm it works correctly?
   5. The software tools you will use e.g. IDLE & MS Word or Jupyter & MarkDown or something else
4. An outline project plan

About 500 words or two sides of A4.

### Project Part 2

Your project report should include:

1. The application you built (regardless of whether it works or not!).
2. The code you wrote
3. The data you used
4. The results you obtained
5. A project development diary describing your work on the creation of this application. This will often (but not always) include your work on:
   * analysis = breaking the problem into parts
   * designs = descriptions of functions, what they will do, how they will work with each other
   * false starts
   * stuff that did not work
   * stuff that did work
   * testing
   * changes of direction
   * results
   * reflections on work done, ideas for improvements
   * …

**IMPORTANT**: Project diaries are NOT written at the end of the project. Project diaries are verbatim records of the application development process. They are a contemporaneous record of thoughts, ideas and events as they actually happened. They are not a dressed-up ‘project report’ version of reality.

About 2000 words.

## Example Project Ideas

Just some thoughts to get you started

### ‘Brexit’ trends

Frequency analysis of the word ‘Brexit’ over time in various publications.

### The vocabulary of the Sherlock Holmes novels of ACD

Word frequency analysis:

* In all novels
* In each individual novel
* How it changed over time

### Vocabulary Comparisons over time e.g. 1850 vs 1900

Choose a few novels by a few authors that were popular in those time periods and compare vocabularies.

### Most frequent words occurring near a character

For instance: In the Sherlock Holmes novels which words occur ‘near’ Holmes and which occur ‘near’ Watson. What does comparing those ‘nearby’ words tell us about the relative roles of those characters in the novel( s).

Take care with sentence, paragraph and chapter boundaries. If the ‘nearby’ word metric is based on a simple ‘number of intervening words’ distance, a ‘nearby’ word might be in a different chapter.

Perhaps do the same with some romance novels, of different time periods, of different genres?

### Most likely next word

For a chosen author analyse a collection of their texts. Build a next\_word(a\_word) function that returns the the word most likely to follow a\_word in that authors works.

Generalise this so that it works for any collection of texts.

### Function = pluralize(a\_word)

Build a function that returns the plural of a\_word. Try to deal with as many irregular words as possible. What if a\_word is already a plural? What does the function do then?

### Function = stemmer(a\_word)

Build a function that returns the root form of a\_word (you decide what that means exactly). So for instance:

| word | root |
| --- | --- |
| coloured | colour |
| industrialised | industrial |
| industrial | industry |

This is a very useful feature to have when searching texts and text databases. In that context the correctness of the function is judged on how effectively it produces the best word, or words, for searching.

Much work has been done on this topic (for example: Porter, about 1980). Create your own version and then compare it to others and then improve.

### Function = tokenize(text)

Build a function that returns a list of the individual words in text, minus any spaces, punctuation, and other ‘special’ symbols. Hyphenation needs to be treated carefully.

Many places and other objects have names that are more than one word. Also ‘Her Majesty’, ‘Armed Forces’, ’ Internal Revenue’, ‘carbon tetrachloride’

### Function = sentiment(text)

If ‘text’ is a product review is it a positive product review or a negative product review? Is the given number of stars in agreement with the language used? What to do with reviews that say nothing, perhaps even literally nothing. How many stars should they get?

Do the same with tweets. Do positive tweets outnumber negative tweets?

### MarkDown extension - numbered headings

There is no feature in MarkDown to have headings numbered. Create a function that adds numbered headings.

| MarkDown | Numbered |
| --- | --- |
| # Head ## Head | # 1 Head ## 1.1 Head |
| # Head\_a # Head\_b ## Head\_c ## Head\_d | # 1 Head\_a # 2 Head\_b ## 2.1 Head\_c ## 2.2 Head\_d |

Consider options to put the heading number after tha heading rather than before, or to turn the hierarchical numbering style on and off (so that all sub-headings under superior head start from 1).

### Concordance Builder

Build a concordance for a book.

### News Media Comparisons

Choose a topic and compare the language used to report that topic by different news media.

For instance Times vs Telegraph vs Guardian vs Daily Mail vs Sun vs Mirror (websites or printed editions?)

### Function = sound\_alike(a\_word)

Alternative, perverse, spellings of a\_word. For example, alternative spellings of ‘fish’ are ‘ghoti’ (gh = /f/ as in enough, o = /i/ as in women, ti= /sh/ as in nation) and ‘fici’ (the sound «fish» is always spelled «fici» in words whose base has more than one syllable)

Dearest creature in Creation,  
Studying English pronunciation,  
I will teach you in my verse  
Sounds like corpse, corps, horse and worse.

http://spellingsociety.org/#

### Story Generator - mark I

The program asks for inputs such as the name of a place, an action, an adjective, a preposition, a proper noun, etc. Once all the inputs are in place, they are inserted into some ready-made strings and assembled into a pre-made story template.

Perhaps have multiple templates and ask the user to select the ‘type’ of story required.

### Story Generator - mark II

The program starts with:

* ready-made collections of different word types: verbs, nouns, adjectives, and so on.
* rules for combining word types
* template strings and storylines

The program prompts the user to enter a series of inputs. These can be particular adjectives prepositions, verbs, nouns, etc.

Once all the inputs are in place the inputs, rules, strings, and templates are combined with some randomness to generate a story.

### 35. Plagiarism Checker

Content writing is one of the most prolific online businesses. The market lacks a free tool that can be used to check for plagiarism in documents. You can use a natural language processing library along with the Google search API to create a program that searches the first few pages of Google and checks for plagiarism.

### email spam filter

Lots of online material to get you started. Many spam filters rely on Bayesian statistics which is not too difficult but is mathematical and would be yet another thing to learn.

### More ideas

Breakfast Experiments <https://languagelog.ldc.upenn.edu/nll/index.php?s=%22breakfast%20experiment%22>

=== end ===