Assignment – 1

Data Structures and Algorithms (DSA) – Python Word Count

Note: 1. Solve all questions yourself, practice will help you in getting better understanding.

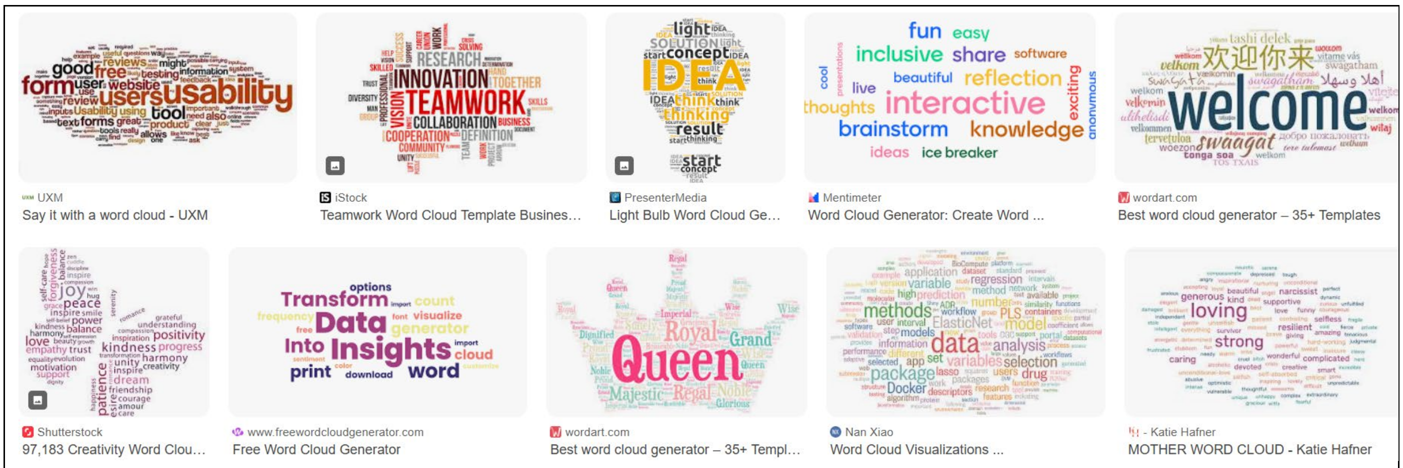
2. Do not use AI tools (ChatGPT etc); trying and debugging will build thorough concepts.

3. Submit a single ***yourname.py*** file only.

4. Submit your assignment file via email to sarmadidrees@mcs.edu.pk with subj: yourname.

5. Deadline for submission is **13 Oct**.

6. Your assignments will be tested with Turnitin –– ***No marks on plagiarism / AI content***.


UXM
Say it with a word cloud - UXM

iStock
Teamwork Word Cloud Template Busines...

PresenterMedia
Light Bulb Word Cloud Ge...

Mentimeter
Word Cloud Generator: Create Word ...

wordart.com
Best word cloud generator – 35+ Templates

Shutterstock
97,183 Creativity Word Clou...

www.freewordcloudgenerator.com
Free Word Cloud Generator

wordart.com
Best word cloud generator – 35+ Templ...

Nan Xiao
Word Cloud Visualizations ...

- Katie Hafner
MOTHER WORD CLOUD - Katie Hafner

Word Cloud

You've likely seen word-clouds before, if not, there are samples above. In order to create word clouds, the *software finds the most frequently occurring words in a text file*. This mini-programming assignment will ask you to do just that. We'll use the text of the famous novel by Charles Dickens, A Tale of Two Cities, in our assignment.

Part 1: Writing and running Python code

1. Make sure you have the environment for the course already setup (see section 1.4 of Text book Python for Data Analysis and follow the procedure.
2. Feel free to create a Python program using a text editor (nano) or you can do all the work in Python shell. Or, if you already have some experience in Jupyter, feel free to do your work there instead. We'll be working in Jupyter from this week on, so this assignment is just to get a little practice in Python programming.

Part 2: Grab source files

Included in the source files are:

1. **word_cloud.py** <-- Starter file if you wish to use it
2. **98-0.txt** <-- Tale of Two Cities, by Charles Dickens. Credit to Project Gutenberg.
3. **stopwords** <-- common words to exclude. Credit to Andreas Mueller.

You can also use **nltk stopwords** instead of those provided. Feel free to do so if you wish.

Part 3: Word Count

To complete this assignment, you will want to read and clean the input, then count the frequencies of each word. Remember that the data science process involves some pre-processing, then consists of some analysis itself. Optionally, you can also filter out common words ("the", "this", "and", etc.) by excluding words which appear in the stopwords file.

Overall, your approach will be:

- Create a data structure to store the words and the number of occurrences of the word.
- Read in each word from the file, making it lower case and removing punctuation. (Optionally, skip common words).
- For each remaining word, add the word to the data structure or update your count for the word
- Extract the top ten most frequently occurring words from your data structure and print them, along with their frequencies.

Checking your solution:

You will get different counts on words depending on what punctuation you remove, what stop words you use, etc. So don't worry too much about getting the exact count we have. But if you want to see what we found, here are two examples:

Without using stop words and removing the punctuation (. , " "), the top 10 most common words should be:

the : 8177

and : 4984

of : 4122

to : 3536

a : 2976

in : 2612

his : 1998

it : 1879

i : 1872

that : 1861

Using the stop words and removing the punctuation (. , " "), the top 10 most common words should be:

said : 642

mr : 616

one : 420

lorry : 313

will : 290

upon : 289

little : 264

```
man : 259
defarge : 259
time : 236
```

*Note: at least "said" and "mr" seem to be common words. Feel free to add more to your stopwords file if you wish to get to less common words.

Hints

1. **Which Data Structure?** If you aren't sure which data structure to use, remember that we discussed a data structure this week that gives us a key and a value at that key (dictionaries). This could be really useful here.
2. **Stripping off Punctuation.** The command "replace" on a string will replace one letter with another. For example:

```
word = word.replace(".","")
```

will remove any periods from the word.

3. **Sorting the data structure.** If you used an unordered data structure like a dictionary, you might need get the values out of it (into a list) to sort it. You could also use "collections.Counter" to help with this step.