CS69001 Computing Laboratory – I

Practice Exercises: Set 3

Practice Problems on Python Programming

1. Write a program that accepts a sequence of lines (strings) as input from the user, and prints each line after changing the case of every alphabetic character. In other words, every alphabetic character in lowercase is converted to uppercase, and every alphabetic character in uppercase is converted to lowercase, and non-alphabetic characters are not modified. The program should stop executing when the user enters "stop" as the input line. (**Hint:** user input can be taken using `raw_input()`.)

2. In many online applications, you have to set your password. The password must follow certain rules, so that it is sufficiently 'strong', that is, it cannot be guessed easily. Write a Python program that helps a set of users to (i) choose usernames, and (ii) set passwords. The username should be distinct for each user. The program should first allow a user to choose a username, and check if the username is already in use. If yes, it should inform the user, and ask to select another username. Once a non-existing username is selected, the program should let the user select a password. A password must obey the following rules:

- The password must be between 6-12 characters in length.
- The only characters allowed are numbers between $[0-9]$, roman letters (lower or upper case), and the characters _, `$`, and `#`.
- Lowercase and uppercase letters are treated distinctly.
- The password should contain at least one number, at least one lowercase letter, and at least one uppercase letter.
- The password should NOT contain the corresponding username as a substring.

The usernames and passwords must be stored in a suitable data structure during the execution of the program, and every user must select a distinct username during a particular execution of the program.

3. You are given a plain-text file containing some text. The text contains several sentences separated by a stop (`.` followed by a whitespace, and each sentence contains several words (separated by whitespaces). Note that the text can be separated into paragraphs, where two paragraphs have one or more newlines in between. Write the following scripts in Python.

(a) Segment the given text into individual sentences, and store the sentences in a List. The 0-th element of the List should be the first sentence, the 1-th element should be the second sentence, and so on.

(b) Print out the longest sentence in the given text, where the length of a sentence is the number of characters in the sentence.

(c) Readability of a piece of text is measured using the Gunning Fog index:

`https://en.wikipedia.org/wiki/Gunning_fog_index`

Write a function to measure a simplified version of the Fog index for a string taken as argument:

- Measure the average sentence length in words
- Count the number of complex words, where we consider any word have more than 7 characters as complex.
- Compute Fog index as 0.4 times (average sentence length + percentage of complex words).

(d) Count the frequency (number of occurrence) of each distinct word in the text. Note that your script should make only one pass over the text. Lowercase and uppercase letters should be treated equivalently. (**Hint:** Create a hashtable/dict, where each distinct word is a key, and the value is the frequency of this word.)

4. In Assignment B2, you had used a specific textual format to input a Binary Tree (BT).

```
297 1 1
319 0 1
124 1 1
282 0 0
530 1 1
424 0 0
287 1 1
214 0 1
471 0 0
376 1 1
527 0 1
149 0 1
173 0 0
200 0 0
253 0 0
```

Consider that the textual input is now given in a file. Write a program to parse the file and create the BT. The name of the file will be stated as a command-line argument to your script.

To store the BT, use the Python library NetworkX

https://networkx.github.io

that provides a large number of functionalities for storing and processing graphs. Print out the inorder and preorder traversals of the BT created, to verify the correctness of your script.