Lab 6 Part 1: a menu-driven program for text analytics

You are familiar with the classic programming I-P-O cycle, that is, input-process-output. In an IPO example, the program is invoked, the data is read from a file (or provided interactively), then the results are generated (saved or shown) and the program terminates.

An alternative is the presentation of a menu of options, something like what you might see at an ATM bancomat machine.

In this example, the user submits the card, the system authenticates the user, the user is presented with a set of options, one of which is to exit the program; the program runs until the user chooses to quit.

At a high level, the general structure for all menu-driven programs is more or less the same:

```
( Pre-processing steps )
Set Finished to False
While Not Finished
      Present Menu of Options
      Get User Choice
      Case Choice of
           1 :  execute option 1
           2 :  execute option 2
             …
           9 :  set Finished to True
```

One of the main advantages of this approach for the programmer is that it is easy to develop and integrate program functionality one unit at a time, and then test them in any desired order.

In the example of the ATM, the pre-processing would be to authenticate the user, while in one of our recent examples, it might be to get the file name and conduct any data corrections, such as converting units of measurement, or adjusting for missing entries.

Your task is to implement a menu-driven program for text analytics in Python. Instead of using the notebook format, you need to write a .py file in an IDE. The program is launched from the command line, asks for a text file containing a dataset, performs a preprocessing step and then presents the user with a menu of possible functionality:

Menu of Options	Explanations
Probability estimate using	Given a sentence, it returns the corresponding score, based on the chosen method.
1. Zeroth order MC	
2. First order MC	
3. Second order MC	
Autocomplete	
4. First order MC	Given a word and a number k, it returns the most probable words to follow in descending order of their likelihood.
5. Second order MC	Given 2 consecutive words and a number k, it returns the most probable words to follow in descending order of their likelihood.
Text generation	Given a number k, it returns the k sentences generated by the selected method.
6. Zeroth order MC	
7. First order MC	
8. Second order MC	
9. Quit	

For this program, what is required by way of pre-processing, before the menu is presented? What data structures do you need for that step? What should they contain?

You can write your program in any IDE of your choice. We would recommend using the community version of Pycharm.

Hint: use local functions to encapsulate your code into smaller subroutines when you implement the execution for each of the options. This will help you to improve readability of your code and, thus, to elevate debugging your code.