Quantcast Summer Internship 2024 Report (Yash Vekaria)

Purpose: of this report is to summarize my approach to solve the assigned coding problem.

Approach: My approach to extract the most active cookie corresponding to the queried date has been divided into the following steps:

1. Performing validation of the user input and raise appropriate error (if input is invalidated)
   I perform tests under three broad categories to test the user input:

   - *Testing cookie log file name input by the user – I perform the following checks*:
     - Length of the filename should be non-zero
     - Extension/Type of the file should be ".csv"
     - Existence of the file in the current directory

   - *Performing Tests to validate the date input by the user*:
     - Length of the date string should be non-zero
     - Length of the date string should be exactly 10 as per the instructions
     - Input should be a valid date in YYYY-MM-DD format
     - The date should not be older than the year 1994. This is because cookies were introduced on the web for the first time in 1994. So, technically, there should be no logs dated before this year.
     - The queried date should not be any future date as cookies cannot be available in the log file for any future date. (I do not check if the date in the logfile is a future date or not as that case is already eliminated by imposing a test on the input, however it can be easily done).

2. Reading the cookie log file

3. Process the cookie log file:
   This step involves the following sub-steps:

   - Preprocessing each line of the logfile to remove any whitespaces in the beginning or end of the cookie string or datetime string.
   - Ignoring a specific log file entry under the following scenarios:
     - If cookie string or datetime string has whitespace inside the string
     - If either the cookie string or datetime string is empty
     - If the cookie string is not valid cookie string as per RFC 6265

4. Create cookie map:
   I create a dictionary of dictionary type mapping of date present in the log file to all the cookies associated with each date. Each cookie key has a corresponding value tracking the number of occurrences of the cookie value on that particular date.

5. Sorting the sub-dictionary for the queried date:
   Finally, the sub-dictionary mapping for the queried date (that contains all cookies observed on that date as keys and their frequencies as values) is extracted and sorted based on the values in descending order.

6. Printing the most active cookies:
   Finally all the cookies that occur with the same frequency as the highest frequency are printed on the console output.

7. Code Testing:
   Rigorous testing of different corner cases and valid as well as invalid scenarios is performed using Python's unittest framework for each function used in the coding assignment.

Coding Style: I have followed the modular approach of writing the production grade code following appropriate naming conventions, usage of commenting for different parts of the code, and making the code reusable (if needed) by separating out different functionalities. All the test cases are written and tested using test.py file. Appropriate exception handling is used as necessitated. Using pandas could have made the solution even more simpler. However, it has not been used as it was disallowed in the instructions. Naming conventions used are as follows:

- Class Names:
  - InputValidator()
  - CookieLogProcessor()
  - CustomError()
- Function Names and variables:
  - These are written in lower cases with interpretable and intuitive names and multiple words are separated by underscores.
  - Internal functions of a class starts with "_" (underscore)
- Test Suite:
  - All functions start with test_ (followed by the function name being tested) as supported by Python's unittest framework and class names start with "Test" followed by the name of the class being tested.

Assumptions made:

Certain assumptions are made based on the instructions provided:

- If whitespaces are present before or after the cookie string or timestamp string, then I remove such whitespaces using strip() and then see if the remaining entry is valid or not. If it is, then it's considered, else ignored.
- It is assumed that the CSV file has the first row which is always the header and remaining lines have two values for cookie string and timestamp respectively that are

separated by a comma. If multiple commas are present, only the first two entries will be parsed. Since libraries like Pandas were discouraged to use, I read the csv file using the general file input/reading method. CSV parsing libraries automatically escape such characters to avoid this problem.

- Cookie string is considered to be present as per the guidelines of RFC 6265 and contain the following characters: a-zA-Z0-9!#$%&'*+-.^_`|~. Empty cookie strings are considered as invalid.
- Timestamp in the log file is expected to be in “%Y-%m-%dT%H:%M:%S%z” format while queried date is expected to be in “%Y-%m-%d” as interpreted through the instruction examples.