Python for Informatics

Assignment 6

"URL Reader"

As you complete this assignment, please do not use single character variable names. Instead, use meaningful names (names that tell the reader what is represented and why). For example, "chunk" is meaningful, "c" is not.

Description:

- Rename the ***socket1.py*** program from our textbook to ***URL_reader.py***.
- Modify the ***URL_reader.py*** program to use ***urllib*** instead of a ***socket***. This code should still process the incoming data in ***chunks***, and the size of each chunk should be 512 characters. Note that our textbook refers to a data chunk as a "block". *Chunk, block, and buffer* are all commonly used terms in the industry to refer to an aggregation of data that is treated as a single unit to be processed, often as part of a set of such units. Be sure that you are accepting an URL from the user (i.e., keyboard). The section entitled **Reading binary files using urllib** (please read this section carefully!) shows you how to open a file and read fixed sized binary (character) values from a file, one chunk at a time. The first and second examples in the **Reading binary files using urllib** section separate the action of opening the file from the action of reading the file. The motivation of this assignment is **to allow for the processing of very large files**, which may be too large to fit into working memory. You cannot use a chunk (buffer) that grows beyond the 512 character limit! Instead, the idea is that you will read a fixed size chunk, and print it; then read the next fixed size chunk, and print it; then, read the next fixed size chunk, and print it; then, etc.. Do not use the value 512 as a magic number. Rather, it should be established as a named constant. ***Do not use magic numbers! A named constant is designated in ALL CAPS, with underscores separating multiple words*** (this is mentioned in our textbook). By the way, the term "magic" indicates that the effect of the number is in some way wondrous or inscrutable. Through the use of named constants, we remove all ambiguity, wonder, and inscrutability from our numbers.

  Many students make the mistake of doing something like this:

  ```
  entire_file += chunk
  ```

  Reading chunks in a loop and concatenating those chunks into one very large "entire_file" buffer defeats the main intent of this assignment. You cannot do this!

- Be careful with your names. If the value you are assigning to it is a *chunk size*, don't use the variable name *chunk*. Instead, establish it as the named constant ***CHUNK_SIZE***. Make your

names meaningful! ***MAX***, ***PRINT,*** and ***DISPLAY*** are not as meaningful as ***PRINT_LIMIT***. Since you are processing nothing but chunks of characters, there are no ***words*** in this assignment. So, please don't include ***word*** as part of any variable or constant name. Your names should clarify, not mislead.

- Add code that prompts the user for the URL so it can read any *web server text file resource*. The prompt should tell the user to type in an URL of a web server text file resource.

  The URL should identify a text file that is provided by a web server such as http://data.pr4e.org (Links to an external site.).

  Here's another web server that offers text files for download by URL:

  http://textfiles.com/adventure (Links to an external site.).

- Add error checking using ***try*** and ***except*** to handle the condition where the user enters an improperly formatted or non-existent URL.
- The decode method can be used to convert the binary data to a string format that is pleasing to the eye. For example,...

  print(str_utf8.decode(chunk))

  Count the number of characters received (read), and stop displaying any text after it has shown ***exactly*** 3000 characters. Space characters, tab characters, and newline characters are characters, and should therefore be included in your count. All of your processing is in terms of chunks. So, you are not allowed to count or print character by character. Instead, you must increment your count by the size of the chunk of characters that you read. That size will always be equal to your default chunk size, except for the very last chunk that you read from the file. The last chunk that you read from the file will likely be less than your default chunk size. Since your chunk size will not divide evenly into 3000, you will need to add some logic to ensure that exactly 3000 characters (no more, and no less) are displayed. When you print your chunks, it is okay for them to be separated by a newline character. Many students make the mistake of doing something like this:

  entire_file += chunk

- Reading chunks in a loop and concatenating those chunks into one very large "entire_file" buffer defeats the main intent of this assignment. You cannot do this! Any other values that you might use should be derived from these two named constants. The idea is for to allow for the values 3000 and 512 to be adjusted (to 5000 and 256, for example), and still ensure that your code operates correctly.

  Very important! You are not allowed to calculate manually or programmatically the number of iterations needed to reach your 3000 character print limit. In other words, instead of counting iterations (or divisions), you need to count the number of characters that you have read so far.

- Continue to retrieve the entire document, count the total number of characters, and display (i.e., *print*) the total number of characters (i.e., how many characters are in the entire document). The reading of the remainder of the document should be done within the same read loop that you use to read the beginning of the file. In other words, you must use one and only one read loop!
- Be sure that *all constant numeric values are established as named constants, and that they adhere to the Python convention of naming constants*.

Note: Because you are printing characters one chunk at a time, and you need to stop printing when you reach 3000 characters, there's a point where ***you will need to print only the portion of the chunk*** that enables you to reach the 3000 character print limit. **This printing of calculated portion of the last printable chunk is arguably the most challenging aspect of this assignment.**

Deliverable:

Please do not include multiple versions of your code. You should only submit the final version of your program. Two files as attachments at our course shell assignment page. The first file should be a Python ***URL_reader.py*** file with the specified functionality. The second file should be a screenshot image file (.png or .jpg) demonstrating the correct execution of your program with an URL of your choice. Your image does not need to show the entire output. Rather, it should just show the final result. Please ensure that your full name is specified as a Python comment at the top of the ***URL_reader.py*** file.

Submission Deadline:

Please see the course schedule in our syllabus for all assignment submission deadlines.

Peerwise Reminder:

No more reminders.