

1336 Comprehensive Final Examination

General Statement

You must write this assignment starting with **Template for Functions (chapter 5+).py**. Refer to the **Style Requirements.pdf** document posted in Blackboard for more requirements. Data files will be supplied with these instructions. Use **Template for Libraries (chapter 5+).py** for writing modules that will be imported.

The Problem

Write a file named **FinalExam.py** with a main program that imports the library module **FinalExamFunctions.py** and calls the sentence processing function (described below) as long as the user says they want to continue.

Build a library module named **FinalExamFunctions.py** with the 3 functions described below.

Global constants: build a tuple named ALPHABET, including both lowercase and uppercase letters, i.e.,
`ALPHABET = tuple(['a', 'b', ..., 'z', 'A', 'B', ... 'Z'])`.

Make sure the lowercase letter are first in the list. Also use a global constant for the default data file name, which is "characters.txt".

Write a **sentence processing function** that passes the default file name to a file reading function and receives a buffer list in return. It then passes that list to the list processing function and receives a sentence list in return. Finally, the sentence processing function prints the sentence list one item at a time with a space between the list items.

Write a **file reading function** that tries to open the file named in the parameter and read file of numbers into a list named buffer. Use a general purpose exception to handle any error: if there is ANY error, simply leave the buffer list empty. Finally, return the list.

Write a **list processing function** which receives a list as a parameter and loops through the list doing the following: convert each list element to integer and use the value as an index of ALPHABET to get a character, which it concatenates to a string named thisWord.

If the conversion or index step generates ANY error, append the thisWord to a list named sentence (but only if thisWord is not an empty string at that point) and then empty thisWord before continuing to loop through the buffer list.

When the loop reaches the end of the list, be sure the last word is also concatenated to the sentence list. After the sentence list is built, return the sentence list to the sentence processing function.

Extra credit (10 points)

Build another global constant tuple of punctuation symbols that you will accept (',', ',', ';', '?', and so on -- you decide). If a list item causes an error in the list processing function, catch it with the exception as described above, but test to see if it is in the punctuation list. If it is, append the punctuation to the word before proceeding to append the word to the sentence list.

Extra credit (70 points)

Add a function to the library that allows you to type a string which will use the alphabet tuple to write the index of your typed characters, one at a time, to build a data file named "characters.txt" that can be read by the sentence processing function. The input is typed as a sentence, not as single words, or characters, at a time. Modify the main program to offer a menu choice: call the typing function, call the sentence processing function, or quit. Continue offering this choice until the user chooses to quit.

1336 Comprehensive Final Examination

Special Notes:

1. You earn up to 60% by correct operation and up to 40% with good style, readability, and documentation.
2. You are on the Honor System not to consult others on this problem (the examination is open book and open notes, so you may use any resources, including the Internet, that are not other people's code). I will not become the Cheat Police, and I will not use online surveillance. But I will not accept identical code submissions with only the names or other obvious and trivial things changed. It is not as difficult to spot as might be supposed.
3. Code that is *essentially identical* to another person's code (ignoring changes in spacing or other trivial adjustments) is considered evidence of cheating and **all identical code submissions receive a score of 0 on the examination**. If your code is identified as cheating in my Blackboard feedback you may contact me to make your case.
4. If you use code from the Internet, and another student does as well, it could result in parts of your submissions being identical. It's not my job to understand how that happened. Please use Internet examples to *inform* your code, but do not copy it exactly, or you risk having to explain duplications with others as described above.

Submit (in **Blackboard**) the files **FinalExam.py** and **FinalExamFunctions.py**.

Name the files **exactly** as given. Do not add anything to the program name(s). Not your name, not your ID number, not words like "exam" or "revised", or any additional characters as part of the file name.

The **Python** program is scored with a maximum value of **100 points**.