Computer Programming
Spring 2020

CSI2100-01 Lab 10

Bernd Burgstaller
Yonsei University

Overview

- Programming Problems

- Deliverables, Due Date and Submission

- Notes:

  1) Please consider using the OnlinePythonTutor when debugging your code.

  2) Some lab problems can be tested with Hyeongjae Python.

     - Please refer to the last page of this assignment for details.

  3) Because of our coronavirus precautions, you are provided with an excerpt of the relevant sections of the textbook together with this lab assignment.

Hint

- Please do **not** submit a main program with problems 1–3.

Programming Problems

Problem 1: Implement the following function:

```
def countLetters(line):
  """ Count all letter characters in string ``line'' and write the
      result to file ``answer.txt''.


      The number of letter characters must be written to the file:
       countLetters('abA1 23') -> writes 3
       countLetters('!') -> writes 0
  """
```

Hint: to write an integer to a file, you need to convert it to a string first. Be sure to add a new-line at the end of the string that you write to the file.

Example: `myfile.write(str(22) + '\n')`

Problem 2: Write a function that receives 3 filenames as **strings**. The content of the first and the second file should be copied onto the third file (the contents of the first file followed by the contents of the second file).

You should assume that both the first and the second file are textfiles and that all lines end with a newline '\n' character. Do not add additional data during the copy operation. (For example, do not add an additional '\n' between the first and the second file.)

```
def copyFiles(f1, f2, f3):
  """ Copies f1 and f2 onto f3.


      The function assumes that files f1 and f2 can be opened, and that
      no error occurs in writing file f3.
      Therefore, the function will always return 0.
      (Error handling with file I/O will be part of next week's lecture.)
  """
```

Your function will be called as follows:

```
copyFiles('in1.txt', 'in2.txt', 'out.txt')
```

Problem 3: Based on textbook p. 333, Exercise P4. implement the following function:

```
def countAllLetters(line):
  """ Counts letters in 'line' and returns result list. If the line
      does not contain any letter, returns an empty list.

      Note 1: the list of letters must be sorted alphabetically.
              (This is a requirement in addition to the textbook problem.)
      Note 2: your letters in the result-list must be stored in lower-case.

  """
```

Your function will be called as follows:

```
countAllLetters('This is the shortest LINE...,   ')
```

For this example, your function must return the list

```
[('e', 3), ('h', 3), ('i', 3), ('l', 1), ('n', 1), ('o', 1), ('r', 1),
 ('s', 4), ('t', 4)]
```

Hint 1: we do not distinguish between lower- and upper-case letters. Thus you are advised to convert your function's input string to lower-case letters (in Lecture 8, we discussed a string method to do this conveniently.)

Hint 2: to count all letters from the alphabet, you could use a list of 26 integer counters, which you initialize to all zero: [0, ..., 0]. Python's multiplication operator is convenient here:

```
>>> [0] * 3
[0, 0, 0]
```

Hint 3: you will have to convert back and forth between characters from the user, and the index values in your list of counters. The **`ord()`** and **`chr()`** functions from Lecture 2 are helpful here.

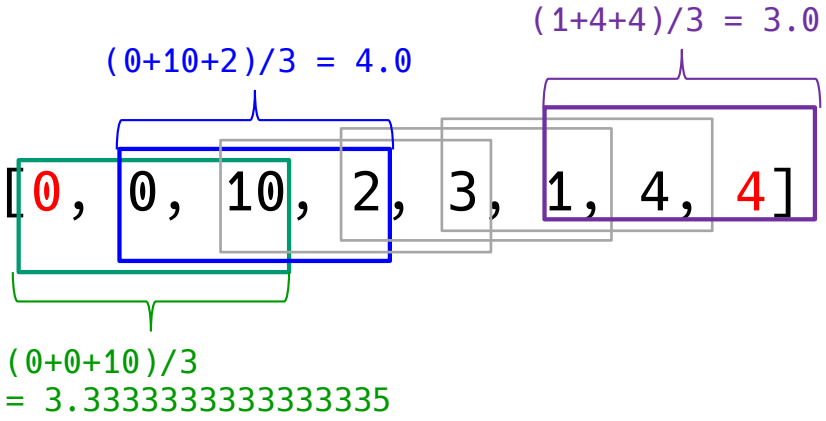
Hint 4: after counting all letters in the user input, you should create the result list and **return** it.

Problem 4: Data measured from sensors may contain random noise. One way to smooth such a list of values is to replace each value with the average of its neighbors, including the value itself. For the elements at the edge, which don't have enough neighbors, we substitute the original value for the missing neighbor:

Original list: `[0, 10, 2, 3, 1, 4]`

List with substituted values at the edge: `[0, 0, 10, 2, 3, 1, 4, 4]`

Computing the average values is called a sliding-window computation, as can be seen in the following example. (Each rectangle represents a ``window''.)



The averaged result list for the example on the previous slide is:

```
[3.3333333333333335, 4.0, 5.0, 2.0, 2.6666666666666665, 3.0]
```

Your program must read its input from the provided file **data.txt**. After smoothing, your program must **print** the smoothed list **on the screen** (like with the above example). Note that you should print each float value as is, without formatting to a certain number of floating-point digits.

Hint 1: to compute the list where values have been substituted at the edges, you can make use of Python's list concatenation operator:

```
xl = [ mylist[0] ] + mylist + ...
```

Hint 2: you should test your program with various input data sets. Edit the file data.txt for this purpose. You can edit the file data.txt in PyCharm. You can assume that file data.txt will contain one integer per line. The file contains at least one integer. The file does not contain any other data than integers.

Hint 3: please have a look at the provided file data.txt to see how input looks like. Basically, there's one sensor data-value per line. Each line ends in a new-line.

Problem 5: Download program SparseText.py from the Lab 10 folder on YSCEC. Read the source code and the explanations in the textbook (Section 8.3.4, pp. 300) to understand what the program is doing. Modify the program such that it removes all vowels in the text (a, e, i, o, u, A, E, I, O, U). You also find text file alice_tea_party.txt to test your program in the Lab 10 folder on YSCEC.

- Please change the program output such that the number of removed vowels are reported. The line “x occurrences of the letter ’e’ removed” should be replaced by "y vowels removed”. (y are the number of removed vowels).
- Note that in the code provided on YSCEC the program greeting has been removed. (Please keep this as it is.)
- Your output-file must start with 'new_' instead of 'e_'.
- **5% extra score**: Line 16 of the provided source-code relies on the fact that each line of the input-file ends with a new-line character. This assumption is invalid in the general case. As an example, you may consider file text_no_eol.txt provided on YSCEC, which does not contain a new-line character at the end of the last line. Indeed, with this file the statistics printed by the program are wrong (“3 out of 8 characters removed”). You may verify the content of file text_no_eol.txt with the **od** tool mentioned on page 9 of the Lecture 8 slide stack. To get the extra score, you should fix the program such that it works correctly even with text files which lack a new-line.

Marking Criteria

- Marking Criteria
  - Score is only given to programs that compile and produce the correct output with Python version 3.
  - Points are deducted for programs that are named wrongly. See the list of deliverables for the required file names.
  - Points are deducted for programs that produce warnings.
  - Points deductions on programming style: please provide comments in your code.
    - Please provide docstrings with **all your** functions.
    - It is not necessary to provide docstrings for functions **in code provided to you**.
  - Please pay particular attention to the **requested output format** of your programs. Deviating from the requested output format results in points deductions
  - Use of modules:
    - You are allowed to use the `sys` module with all programing problems.
      - Please always use an exit code of 0 with sys.exit(): `sys.exit(0)`.
    - You are allowed/asked to use modules stated with individual programming problems and contained in provided code (if any).
    - No other modules are allowed.

Plagiarism

- Plagiarism (Cheating)
  - This is an individual assignment. All submissions are checked for plagiarism.
  - Once detected, measures will be taken for **all** students involved in the plagiarism incident (including the ``source'' of the plagiarized code).

Deliverables, Due Date and Submission

- Please prepare the files for the programming problems. The names of the files, their due dates and the archive file-name is given in the below table.
  - Please upload your archive file by the stated due date on YSCEC.
- Lab problems marked as '✓' can be tested on our Hyeongjae Python site http://hyeongjaepython.elc.cs.yonsei.ac.kr/

Problem	File name	Due	Archive name	Hyeongjae Python
1	lab10_p1.py	Wednesday June 3, 2020 23:00	lab10_<student id>.zip	✓
2	lab10_p2.py			—
3	lab10_p3.py			✓
4	lab10_p4.py			—
5	lab10_p5.py			✓ (N1)

(N1): no tests of the functionality for the extra score are provided. Please come up with your own test cases and test in PyCharm on your local computer.