CSC 242: Introduction to Computer Science II

Assignment 8

Due Wednesday May 27th, 11:59 pm

Reading

Review Chapter 10 and read Chapter 11 in **Introduction to Computing using Python: An Application Development Focus, Second Edition** by Ljubomir Perković.

Logistics

In this class programming assignments may be completed in consultation with up to two other classmates. You must identify the classmates with whom you collaborate in a comment at the top of the assignment, and the number of collaborators on any assignment **may not exceed two other people**. You must also submit a comment in your submission for each assignment that describes in detail how each collaborator contributed to the assignment. If you did not collaborate with anyone on the assignment, you must include a comment that says that. You may not under any circumstances discuss the assignments with classmates other than your identified collaborators. Working so closely with anyone other than your identified collaborators, Mr. Zoko, or the lab assistant, so as to produce identical or near identical code is a violation of the Academic Integrity policy. This policy will be strictly enforced.

Please include the following with your assignment submission:

1. A comment at the top of your Python file identifying any classmates with whom you discussed or in any other way collaborated on the assignment. You may work (directly or indirectly) with **no more than two** other people.
2. Add a comment at the top of your Python file that describes for each person what they contributed to the assignment. This must be at least 2-3 sentences and be **very specific and detailed**.

A submission that does not include a list of collaborators and a comment indicating how you collaborated with classmates will earn a 0. If you worked alone you must put a comment at the top of your file that indicates that or you will also receive a 0. There will be no exceptions to this rule.

Again, you are subject to all the rules specified in the Academic Integrity policy. Please read it carefully before beginning this assignment.

Assignment

Please remember that you are not allowed to consult online resources when completing homework assignments. If you have questions about this assignment, please contact me.

Implement the functions below in a file called **csc242hw8.py** a template for which has been provided on the D2L site. **You must include appropriate doc strings** (e.g. strings that appear on the line following the function header) to the class that clearly and concisely describe what the functions are doing. A submission without doc strings will not earn full credit.

The functions written for this assignment must be **recursive** and **must not** use global variables. Do not modify the function names or parameters in the template file. The recursive functions **may not use loops except to list the contents of a directory**. In some cases certain built-in Python functions are disallowed. None of the functions should alter the lists passed as parameters nor should the functions make copies of the lists in order to alter them. Only non-destructive solutions should be submitted. Please read the problem description carefully. Function that are described as returning values should not do any printing. Solutions that do not follow these guidelines will not earn full credit, even if they produce the correct results in all cases.

Please note that local variables are absolutely fine in any of the functions.

1. (15 points) Write a **recursive** function **recNestedListSumOfNumbers**() that takes an arbitrarily nested list as a parameter and **returns** the sum of the numbers in the list. You are not allowed to use any list methods in your function other than indexing (e.g. lst[i] for some integer i), slicing (e.g. lst[i:j] for some integers i and j), or len(). The function may not use a loop. Solutions that use built-in functions rather than recursion to solve the problem will earn very little credit. You may assume that the initial value given to the function is a list, and your function does not have to behave in any reasonable way when given something other than a list as a parameter. The list may contain values of any type and you must only take the sum of the ones that are integers or floating point values. The type() function is helpful in determining what values can be found in the list. Numbers found within other collections (tuples, dictionaries, etc.) should not be included in the sum. The following shows the behavior of the function on some sample parameters. Please remember that these examples are just that. Your function must work correctly on all valid parameters, and you may not make any assumptions about the number of nested sublists or the depth of the nesting in the list provided as a parameter to the function:

```
>>> print(recNestedListSumOfNumbers([[1,2,3,4,5,6,7,8,9,10]]))
55
>>> print(recNestedListSumOfNumbers([]))
0
>>> print(recNestedListSumOfNumbers([[[[[[[[[5]]]]]]]]]))
5
>>> print(recNestedListSumOfNumbers([[1,2,3],[4,[[[5,[[[6]]]]]]]]))
21
>>> print(recNestedListSumOfNumbers([[[[[[[[[10,20]]]]]]]]]))
30
```

2. (15 points) Write a **recursive** function **dirPrint**() that takes as parameters a pathname of a folder as a string and an integer indent and **prints** to the screen the pathname of every subfolder contained in the folder, directly or indirectly. The subfolder path names should be output with an indentation that is proportional to their depth with respect to the topmost folder and should use the parameter indent to achieve this. The example below illustrates the execution of the function on several sample folders. Two sets of folders (can be found in the zip file containing the template and posted to D2L:

   Reminder : See the antivirusredux.py example.

```
>>> dirPrint('count',5)
      count
        count\A
          count\A\A
            count\A\A\A
        count\B
          count\B\A
            count\B\A\A
          count\B\B
            count\B\B\A
              count\B\B\A\A
        count\C
          count\C\A
>>>
>>> dirPrint('Test',2)
   Test
     Test\Test1
       Test\Test1\Test1-A
         Test\Test1\Test1-A\Test1-A-A
           Test\Test1\Test1-A\Test1-A-A\A
       Test\Test1\Test1-B
         Test\Test1\Test1-B\Test1-B-A
           Test\Test1\Test1-B\Test1-B-A\B3
             Test\Test1\Test1-B\Test1-B-A\B3\B4
         Test\Test1\Test1-B\Test1-B-B
       Test\Test1\Test1-C
         Test\Test1\Test1-C\C1
         Test\Test1\Test1-C\C2
           Test\Test1\Test1-C\C2\C31
             Test\Test1\Test1-C\C2\C31\A
               Test\Test1\Test1-C\C2\C31\A\A
                 Test\Test1\Test1-C\C2\C31\A\A\A
                   Test\Test1\Test1-C\C2\C31\A\A\A\A
           Test\Test1\Test1-C\C2\C32
         Test\Test1\Test1-C\C3
     Test\Test2
       Test\Test2\A
         Test\Test2\A\B-1
         Test\Test2\A\B-2
       Test\Test2\B
         Test\Test2\B\B1-1
           Test\Test2\B\B1-1\B2-2
             Test\Test2\B\B1-1\B2-2\B
               Test\Test2\B\B1-1\B2-2\B\B
       Test\Test2\C
     Test\Test3
>>>
```

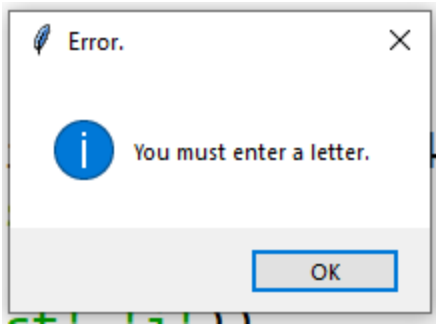
Problem 3:

Part 1 (10 Points): Write a **recursive** function **fileCountWithLetter**() that takes as parameters a pathname of a folder as a string and a string that represents the letter you are using to see if it is contained in a filename you are searching for under the root folder. You must return a count of how many files you find with that letter in the file name in the directory structure. The filename and letter are case sensitive.

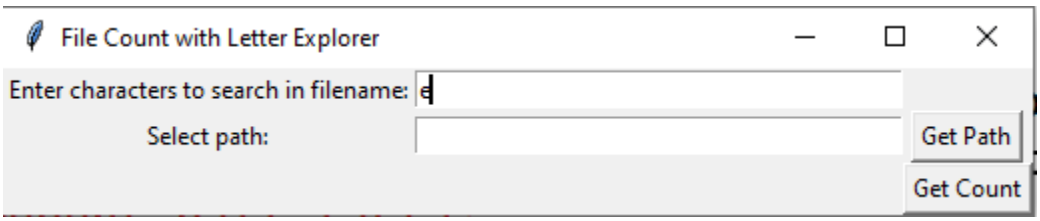
```
>>> print('FILE COUNT WITH LETTER: ', fileCountWithLetter('Test','a'))
FILE COUNT WITH LETTER:  3
>>> print('FILE COUNT WITH LETTER: ', fileCountWithLetter('Test','i'))
FILE COUNT WITH LETTER:  6
>>> print('FILE COUNT WITH LETTER: ', fileCountWithLetter('Test','e'))
FILE COUNT WITH LETTER:  6
```

Part 2 (10 points) Write a GUI named **FileCountWithLetterExplorer** that creates an interface that allows the user to pick a folder using the filedialog. It also contains a field to enter characters to search in the filename. You will use the fileCountWithLetter code from part 1 and show the count of how many files have that name in a popup. In this problem, you will use the filedialog widget to allow the user to pick a folder. Since the book doesn't cover this dialog, I have included an example named FileDialogExample which is found in the homework file to show you how to use the filedialog to pick a directory. The return value of the dialog is a string that represents the directory path you will be recursively searching. You can copy and paste the code from the previous problem. Calling a method recursively is the same as calling a regular function recursively but the method has to have a parameter representing self. You can use either grid or pack to align the GUI elements. She the following screenshots for an example usage:

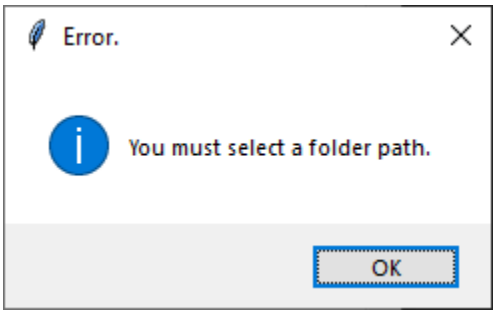
Click "Get Count":



Enter “e” in the first field:

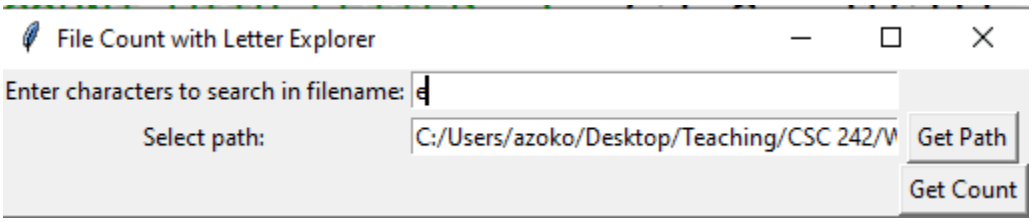


Click “Get Count”:

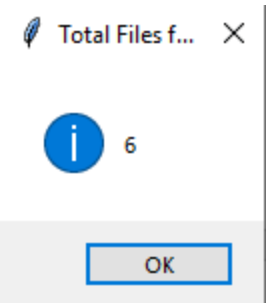


Click “Get Path”. You will see a system specific dialog box appear to select the folder:

Once you select the folder the UI looks like this (files with e are being searched for):



Now, click “Get Count”:



Results should be the same for file and folder combos found in part 3.

Problem 4 (20 points):

Implement a **recursive** function **fileContentsCount**() that takes as parameters the name of a folder and the string we want to find in the file. The function returns the count of files that string appears in. Please note that your function must work as described on any directory structure, not just the one provided as an example. You must use recursion to go through the subfolders. You may not use python crawlers or apis (walk) other than the ones used in the antivirus-redux example to find files in sub directories. Doing so will be an automatic 0 for this problem. You may use a for loop to loop through the contents of a folder. The following illustrates several searches using a sample set of folders and directories located in the zip file containing the exam template. You ***don't*** need to worry about punctuation, case or file error handling for this problem.

```
>>> fileContentsCount('Test-2','April')
2
>>> fileContentsCount('Test-2','like')
4
>>> fileContentsCount('Test-2','e')
11
>>>
```

Problem 5 (30 points):

Its been a few weeks since we've done GUIs. I don't want you to forget how to do them as they will be on the final.

I'm going to change things up a bit. I want you to write a GUI from scratch with the following criteria.

- Design a patient intake form for a hospital.
- Collect three mandatory values via an Entry box. These values are mandatory and the user should be warned if the value isn't entered with a warning presented via a dialog box. You should only show one error at a time.
- Add labels to clearly explain the form.
- Collect at least one value with a slider.
- Based on what the user enters, you must calculate and display a severity of the case. You must display the severity to the screen in whatever way you want when the user submits the form.
- Write the values, including the severity calculated, to a file. The values must be written to the file in such a way that it can easily be read in by another program. Each patient in take should be written to its own line.
- All errors must be caught.

Submitting the assignment

You must submit the assignment using the assignment 8 dropbox on the D2L site. Submit a Python file (csc242hw8.py) with your implementation in it and comments describing your collaboration status. Submissions after the deadline listed above will be automatically rejected by the system. See the syllabus for the grading policy.

Grading

The assignment is worth 100 points. Any student who does not submit comments in the Python file describing the contributions of each team member or indicating that he/she worked alone will earn a 0 on the assignment.