Lab 7: File I/O & Args

Purpose

To learn how to open a file for reading and writing, and to use command line arguments.

Implementation

Create a file named lab7.py. In the file, write a function called main() that takes at least two - and optionally a third - command-line argument. The required argument must be the name of a file to open for reading and the name of a file to write the result in. The optional third argument must be a flag: `-s`. If the user provides the wrong number of arguments or if the optional argument is not `-s` then print:

```
Usage: [-s] infile_name outfile_name
```

Your program should attempt to open the file, infile_name, for reading. Opening the file could fail for several reasons (e.g. the given file doesn't exist or the given file doesn't have the needed permissions). Failure to open the file could result in an exception being raised, causing your program to crash.

We will be learning about exceptions this week. Please review the lecture on exceptions. In short, your code will try something. If that something works, then great, continue on. If that something does not work, then "catch" the exception and do something else. The code outlined below gives the structure of working with operations that may raise exceptions.

```
try:
      # what you want to attempt to do
except:
      # what to do if the previous code raises an exception
```

Should opening the file raise an exception, print the message (replacing `<filename>` with the name of the actual file) in the `except` block:

```
Unable to open <filename>
```

Your program should open a file and read each line. The file will consist of elements separated by a space. The elements are numbers and "other" strings that are not numbers. You can assume that the numbers are always whole numbers. You should keep a count of each kind of

value read from the file and **write** the results into the file, **outfile_name**, when you are finished reading the entire file **(infile_name**).

For example, given the following input file:

```
abc123 34
2 34 h
3333333
2
```

Your program should output the following lines to a file, **outfile_name**:

```
numbers: 5
other: 2
```

So what about that `-s` option? Should the user request that option, additionally keep track of the sum of all the numbers in the file and write the sum after writing the counts of each of the types.

Make sure to have the following lines at the bottom your file to call the main() function:

```
if __name__ == '__main__':
    main()
```

You are free to write helper functions that help your main() function.

Sample Runs

Use the following file contents (create test files by yourself) and command-line arguments for your test. In each run, the `$` symbol indicates the command prompt. You are not required to write tests using `assert`, nor to submit your tests in this lab.

File Contents

```
hi 34
56 77
aef56
5 6 7 8 23 blaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaah g
```

No Options

```
$ python lab7.py test0.txt test0.out
```

The `test0.out` shall contain the following lines:

```
numbers: 8
other: 4
```

Using -s Option Before File

```
$ python lab7.py -s test0.txt test0.out
```

The `test0.out` shall contain the following lines:

```
numbers: 8
other: 4
sum: 216
```

Using -s Option After File

```
$ python lab7.py test0.txt test0.out -s
```

The `test0.out` shall contain the following lines:

```
numbers: 8
other: 4
sum: 216
```

Incorrect Number of Arguments Failure

```
$ python lab7.py
Usage: [-s] infile_name outfile_name
```

Incorrect Option Failure

```
$ python lab7.py -r test0.txt
Usage: [-s] infile_name outfile_name
```

Invalid File Failure

```
$ python lab7.py junk.txt
Unable to open junk.txt
```

Submission

Submit your file to Gradzilla to have it graded. Then, submit `lab7.py file to Canvas by the due date.`