CS 100 2019F Section 015
Homework 12

Due: At the beginning of class on Thursday December 5, 2019.

Do all of the items below and **submit** a text file created with the IDLE editor (or other editor) with the extension *.py* via Moodle. If you run into a problem, post to Moodle describing where you ran into trouble or email your instructor or classroom assistant, or ask your question during recitation hours. If you know the answer to someone's question on Moodle, post a response. You get course credit for asking and answering questions in Moodle.

- Read Section 14.5 (Catching exceptions) in the textbook.
- Read the Python tutorial sections 8.1 (Syntax Errors), 8.2 (Exceptions), and 8.3 (Handling Exceptions). The Python tutorial can be accessed through the documentation installed with IDLE:
  Help → Python Docs → Tutorial → 8.
  Errors and Exceptions If you are using an alternate IDE, visit:
  https://docs.python.org/3/tutorial/errors.html#errors-and-exceptions
  to browse the tutorial online.
- In the Python editor IDLE, create and save a Python file that is named, if your name is Harry Houdini, for example, *HW12_HarryHoudini.py* and begins with a comment containing your name, class and section, the posting date and number of the homework assignment.

Problem 1

Recall that when the built-in function *open()* is called to open a file for reading, but it doesn't exist, an exception is raised. However, if the file exists, a reference to the opened file object is returned.

Write a function *safeOpen()* that takes one parameter, *filename* — a string giving the pathname of the file to be opened for reading. When *safeOpen()* is used to open a file, a reference to the opened file object should be returned if no exception is raised, just like for the *open()* function. If an exception is raised while trying to open the file, *safeOpen()* should return the value `None`.

For example, assuming the file *ghost.txt* doesn't exist, the following is correct output:

```
>>> # open()
>>> print(open('ghost.txt'))
Traceback (most recent call last):
  File "<pyshell#8>", line 1, in <module>
    print(open('ghost.txt'))
FileNotFoundError: [Errno 2] No such file or directory: 'ghost.txt'
>>>
>>> # safeOpen()
>>> inputFile = safeOpen('ghost.txt')
>>> print(inputFile)
None
>>>
```

Problem 2

Recall that when the built-in function *float()* is called it returns a floating point number constructed from a number or string. However, if the string doesn't represent a valid floating point value, an exception is raised.

Write a function *safeFloat()* that takes one parameter, *x* — a number or string that needs to be converted to floating point number. When *safeFloat()* is used to convert a number or string, an equivalent floating point number is returned if no exception is raised, just like for the *float()* function. If an exception is raised while trying to convert a number or string, *safeFloat()* should return `0.0` floating point value.

For example, the following is correct output:

```
>>> # float()
>>> float('abc')
Traceback (most recent call last):
  File "<pyshell#9>", line 1, in <module>
    float('abc')
ValueError: could not convert string to float: 'abc'
>>>
>>> # safeFloat()
>>> f = safeFloat('abc')
>>> print(f)
0.0
>>>
```

Problem 3

A radar speed gun is a device used in law-enforcement to measure the speed of moving vehicles in miles per hour. The measured speeds are supposed to be stored in a file, one number per line, as follows:

```
65.6
70.2
54.9
```

Unfortunately, due to an intermittent fault, occasionally multiple numbers are written on a single line as follows:

```
73.2 65.6 69.8
```

Furthermore, occasionally the radar gun outputs a single stray character such as: `67.9z`, `6$4.9`, or `a3.9`, to illustrate just a few.

Given a file that has radar speed gun readings, write a function *`averageSpeed()`* to calculate the average of the numbers in the file. Your code must adhere to the following specifications:

a. Prompt the user for the name of the input file to process. When the user enters a nonexistent file name, give the user a second chance. After two wrong entries in a row, quit the program with an appropriate message.
b. Ignore numbers containing stray characters.
c. Ignore any reading for slow vehicles moving at 2 miles per hour or less.
d. Print the final average to the console.
e. Make use of the functions *`safeOpen()`* and *`safeFloat()`*.

For example, the following is correct input/output:

```
>>> inputFile = open('radar.txt')
>>> content = inputFile.read()
>>> print(content)
35.2
1.8
65.6
67.9z
70.2
73.2 a3.9 65.6 69.8
6$4.9
54.9
>>> inputFile.close()
>>>
>>> averageSpeed()
Enter file name: ghost.txt
File not found. Please try again.
Enter file name: phantom.txt
File not found. Yet another human error. Goodbye.
>>>
>>> averageSpeed()
>>> Enter file name: radar.txt
>>> Average speed is 62.07 miles per hour.
>>>
```