Version: 2023/09/05 12:44

CSC1010H Tutorial 7

File I/O and Exceptions

Introduction

Learning outcomes

Skills

- Solve programming problems involving reading from and/or writing to files.
- Construct robust programs that handle exceptions.

Knowledge

- Files: concepts of file, type, filename, file path; commands for opening, reading, writing and closing files.
- Exceptions: concepts of exception and exception handling; common types of exception (including TypeError, ValueError, IndexError, IOError); Form and function of try-catch[-else][-finally] statements.

Question 1 [30 marks] (Manually marked)

On the Vula assignment page, you will find a program called ‘`leader_board.py`’. This is a program that processes data from a reaction times study. It asks the user to input the name of a file and then prints a leaderboard of results.

Each line of the input file consists of the name of a study subject, followed by the number of tests conducted with that subject, followed by the subject’s average reaction time (the sum of the reaction time for each test, divided by the number of tests). For example,

```
Takunda 5 0.415626
Sheila 3 0.304821
Chao 3 0.386983
```

Given this data, the program outputs:

```
Subject         | Average Time
**************  | ************
Sheila          | 0.304821
Chao            | 0.386983
Takunda         | 0.415626
```

The program is not very robust; various problems with the inputs will cause an exception to be raised.

Modify the program so that it can handle all possible exceptions.

- If it cannot open the named results file then it should report the problem and ask for another filename.
- If there is a problem with the file content, i.e. an issue with one of the lines of the file, then it should do the following:
    - Report the line number and the nature of the problem.
    - Print the line.
    - Tell the user what the correct format should be.

Continued

NOTE: While there are many potential problems with file content, there are only two distinct types of exception that can arise.

Question 2 [35 marks] (Manually marked)

Write a program called ‘`reaction_test.py`’ that may be used to calculate the reaction time of test subjects.

Sample I/O

```
Enter the name of the output file:
results.txt
Enter subject name (or press 'enter' to quit):
Stephan
Enter the number of tests:
4
Get ready...
Press 'enter' NOW!
Reaction time: 0.390626
Get ready...
Press 'enter' NOW!
Reaction time: 0.448072
Get ready...
Press 'enter' NOW!
Reaction time: 0.359375
Get ready...
Press 'enter' NOW!
Reaction time: 0.390626
Tests complete.
Average reaction time: 0.397175

Enter subject name (or press 'enter' to quit):
```

(User input in **bold**.)

The program will ask for the name of a file to which results will be written. It will then repeatedly ask for the name of a test subject and for the number of reaction tests to conduct on that subject, and then perform those tests.

Each test takes the following form:

1. The computer will print ‘`Get ready...`’.
2. It will pause for a random amount of time from 1 to 5 seconds.
3. It will print “`Press 'enter' NOW!`”, record the current time, and wait for input.
4. The test subject (user) must press the ’enter’ key as quickly as possible.
5. Once input is received, the program will record the current time, and calculate and print the difference – the reaction time – rounded to 6 decimal places.

Once all tests have been completed, the program will print the average reaction time on the screen AND append the result to a file called ‘`results.txt`’ using the same format as for question one.

Continued

NOTE:

- Use the facilities provided by the Python `'time'` module to create pauses and to obtain the current time.
- Use the '`input()`' command to get the subject's response.
- For efficiency, Python buffers output to the screen. This means that the phrase 'Get ready!' might not appear until the program asks for input. The solution is to 'flush' output to the screen.
  For example,

  ```
  print('Get ready...', flush=True)
  ```

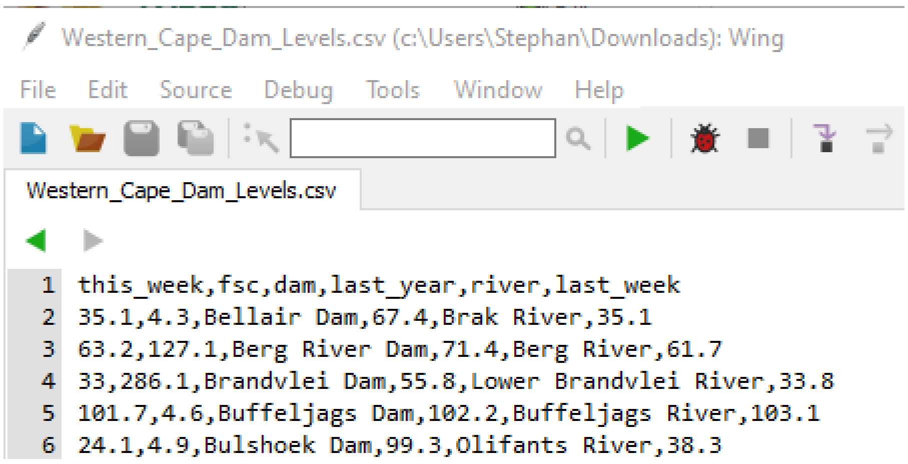
- Unfortunately, Wing IDE also buffers output to the screen. Your program should function adequately, but to see it function as it should, you will need to run your program from the windows command prompt.
- It's possible for an unscrupulous user to skew the results by pressing enter before prompted. The following method can be used to 'consume' inputs before calling '`input()`'.

  ```
  def flush_input():
      try:
          import msvcrt
          while msvcrt.kbhit():
              msvcrt.getch()
      except ImportError:
          import sys, termios
          termios.tcflush(sys.stdin, termios.TCIOFLUSH)
  ```
  (http://rosettacode.org/wiki/Keyboard_input/Flush_the_keyboard_buffer#Python)

Question 3 [35 marks]

This question concerns processing data on Western Cape dam levels. The data is stored in the form of a 'comma separated values' (CSV) text file. You will find a sample file on the Vula assignment page: *Western_Cape_Dam_Levels_280823.csv*.

Windows will treat the file as EXCEL, however, for our purposes it is preferable to simply view the text. This can be done by opening it in the Wing IDE e.g.


Western_Cape_Dam_Levels.csv (c:\Users\Stephan\Downloads): Wing
File Edit Source Debug Tools Window Help
Western_Cape_Dam_Levels.csv

```
this_week,fsc,dam,last_year,river,last_week
35.1,4.3,Bellair Dam,67.4,Brak River,35.1
63.2,127.1,Berg River Dam,71.4,Berg River,61.7
33,286.1,Brandvlei Dam,55.8,Lower Brandvlei River,33.8
101.7,4.6,Buffeljags Dam,102.2,Buffeljags River,103.1
24.1,4.9,Bulshoek Dam,99.3,Olifants River,38.3
```

(The screenshot is for an older data file.)

Continued

The first line describes the data items on subsequent lines. Each subsequent line contains data for a particular Western Cape Dam.

Data item name	Description
`dam`	The name of the dam.
`river`	Name of the river on which the dam sits.
`fsc`	Full Storage Capacity (FSC) in units of 1 million cubic meters.
`this_week`	Current level as a percentage of FSC.
`last_week`	Last week's level as a percentage of FSC.
`last_year`	The level this week last year as a percentage of FSC.

NOTE:

- You CANNOT assume that the order of data items is always the same from download to download. You MUST use the first line to figure out which item is which.

Write a program called "`analysis.py`" that asks the user for the name of a CSV data file and that, for each dam, calculates and outputs the following:

1. This week's level in units of 1 million cubic metres.
2. The level this week last year in units of 1 million cubic metres.
3. The FSC in units of 1 million cubic metres.
4. The name of the dam.

Finally, it should print the total FSC (in units of 1 million cubic metres) and the percentage of that at which supplies stand this week.

Sample Input/Output:

```
Enter the name of the CSV file:
testdata.csv
 THIS WEEK LAST YEAR       FSC DAM NAME
       2.4       3.9       4.9 bulshoek dam
       1.5       4.2       4.9 calitzdorp dam
       7.0      10.3      17.3 ceres dam
      31.2     118.7     122.5 clanwilliam dam
       9.4      14.9      33.9 steenbras dam-lower
      26.1      27.6      31.9 steenbras dam-upper

Total FSC is 215.4 million cubic metres.
Currently at 36.0%.
```

(User input in **bold**.)

Values should be printed to one decimal place.

(The short data file 'testdata.csv' that is used for the sample I/O may be found on the Vula assignment page.)

Continued

Marking and Submission

Submit the `leader_board.py`, `reaction_test.py`, and `analysis.py` programs contained within a single .ZIP folder to the automatic marker. The zipped folder should have the following naming convention:

yourstudentnumber.zip

NOTE: Questions 1 and 2 are manually marked, while question 3 is automatically marked.

Marking Guide

1. Leader board	30
2. Reaction tester	35
3. Dam data analysis	35
Total	**100**

End