L09 Tasks

L09-T1: Exception handling with file read & write

Write a program that:

1. Reads a text file into a list in one subroutine and
2. Writes the list to a file in another subroutine.

The main program should ask for file names for read & write, and contain separate subroutine calls for reading and writing. Both subroutines must have exception handling related to file handling. Everything should be done inside the exception handler, so that possible error situations can be caught.

The structure of the file to be read and written is the same: single integers, always one number per line.

If the file processing is not successful, just inform the user about the situation with the error message below and terminate the program execution with the `sys.exit(0)` command.

The name of the file that caused the problem must be inside the apostrophes:

- `Error processing file 'x.txt', stopping.`

Note that the directory structures look different on Linux and Windows. The Mac directory structures are the same as in Linux.

The `T1_file_in1.txt` and `T1_file_in2.txt` can be found in Moodle. They are already in CodeGrade, please do not upload them.

When your program writes to disk, the result will be compared to a model text in CodeGrade. You recognize this part of the text by the text: `### Print file ###` in outputs.

Example run 1 (successful):

```
Enter the name of the file to be read:
T1_file_in1.txt
File 'T1_file_in1.txt' read successfully, 7 lines.
Enter the name of the file to be written:
T1_file_out.txt
File 'T1_file_out.txt' was successfully written.
```

Example run 2 (problem with reading a file):

```
Enter the name of the file to be read:
T1_file_in2.txt
Error while processing file 'T1_file_in2.txt', stopping.
```

Example run 3 (problem with writing a file):

```
Enter the name of the file to be read:
T1_file_in1.txt
File 'T1_file_in1.txt' read successfully, 7 lines.
Enter the name of the file to be written:
/var/cannot_write.txt
Error processing the file '/var/cannot_write.txt', stopping.
```

L09-T2: Handling exceptions with user input

Write a program to test for various exceptions on user input, especially `ValueError`, `IndexError`, `ZeroDivisionError`, and `TypeError`. Implement your program as a menu-based program. You should make your own subroutine to test every error type.

In the error-testing functions, you should use a structure where you first try to perform the desired operation with typically 2 lines of code, and if it is not successful, you go to the exception handler. The exception handler should only consider the exception to which this subroutine is devoted to.

`ValueError` is an exception that occurs when a function receives an argument of the correct data type but an inappropriate value. For example, a negative integer is passed to `math.sqrt()`.

`IndexError` typically occurs when referring to a list with an index that does not exist. Therefore, in the subroutine, test with a list of elements 11, 22, 33, 44 and 55. In division calculation, show the results in a format that is 5 characters long and has 2 decimals. Type error occurs, for example, when trying to multiply two strings together, i.e., forgetting to change the user's input from a string to a number.

Your program should not crash if something else than an integer is given as a menu entry, see example run below.

Example run:

```
What do you want to do:
1) Test for ValueError
2) Test for IndexError
3) Test for ZeroDivisionError
4) Test for TypeError
0) Stop
Your choice:
1
Give a non-negative integer:
-4
ValueError happened. Non-negative number expected for square root.
What do you want to do:
1) Test for ValueError
2) Test for IndexError
3) Test for ZeroDivisionError
4) Test for TypeError
0) Stop
Your choice:
2
Input index 0-4:
7
Got an IndexError, index 7.
What do you want to do:
1) Test for ValueError
2) Test for IndexError
3) Test for ZeroDivisionError
4) Test for TypeError
0) Stop
Your choice:
3
Enter divider:
```

```
0
ZeroDivisionError occurred, divider 0.
What do you want to do:
1) Test for ValueError
2) Test for IndexError
3) Test for ZeroDivisionError
4) Test for TypeError
0) Stop
Your choice:
4
Enter number:
word
Got TypeError, word*word with strings failed.
What do you want to do:
1) Test for ValueError
2) Test for IndexError
3) Test for ZeroDivisionError
4) Test for TypeError
0) Stop
Your choice:
zero
ValueError happened. Enter the selection as an integer.
Your choice:
0
```

L09-T3: Algorithm for classifying lines of strings into groups

Write a program that

1. reads the contents of a text file into a list
2. finds out the different car brands in the file
3. prints the number of different brands and all the different brands on the screen, and
4. saves the different car brands in the file

In the main program, define a list for the data to be read and for different car brands. Also store the names for the file to be read and written.

Read the file in a subroutine that receives the file name and list as parameters. After that, analyze the data in one subprogram and print and save the car tags in the third subprogram. If there was no data in the read file, tell the user about it and don't call analysis and writing subroutines unnecessarily.

The error messages displayed to the user are as follows:

- `"Error processing file 'x', stopping."` [x is replaced by the name of the file that caused the error]
- "The file was empty, no car brand was recognized.", if the read file was empty.

The car brands in the file are arranged in such a way that there is always one car brand in one row and all cars of the same brand are in consecutive rows. Thus, you can find out the different brands by going through the list and whenever the car brand changes, you add the previous brand to the list. *Be careful that the first and last car make and row are processed correctly.*

The `T3_file_in1.txt` and `T3_file_in2.txt` can be found in Moodle. The file `T3_file_in3.txt` is empty, it is not in Moodle. These files are in CodeGrade, please do not upload them.

CodeGrade also compares the output files using linux command `diff`.

Example run:

```
Give the name of the file to read:
T3_file_in1.txt
Give the name of the file to write:
Out1.txt
There were 8 different car brands in the file.
Kia
Mazda
Mercedes-Benz
Opel
Renault
Seat
Toyota
Volkswagen

#####

Give the name of the file to read:
T3_file_in2.txt
Give the name of the file to write:
Out2.txt
There were 7 different car brands in the file.
Kia
Mazda
Mercedes-Benz
Renault
Seat
Tesla
Volkswagen

#####

Give the name of the file to read:
T3_test3.txt
Give the name of the file to write:
Out3.txt
The file was empty, no car brand was recognized.
```

L09-T4: Analyzing Titanic Passengers, Error handling & data cleaning

Your task is to prepare data of Titanic passengers for analyzing. We have an existing .csv file containing the details of all passengers aboard the infamous Titanic cruise on 15 April 1912. You can read more about Titanic from Wikipedia: https://en.wikipedia.org/wiki/Titanic.

The data contains details of roughly 1000 passengers. However, some of the data is missing, for example all passengers do not have their age showing or they are not travelling in any cabin (not that all people could afford a cabin in 1912). We want to be able to analyze the data, but as there are missing data included, we need to clean it before processing. This task requires you to do some basic data cleaning, which you will quite likely run in your life later.

The data is included in `titanic.csv`. It has been fetched from public Github repository (https://github.com/datasciencedojo/datasets/blob/master/titanic.csv)

The file `titanic.csv` is available in CodeGrade, you should not upload it.

For this task, you need to do the following tasks:

1. Ask user input for the filename to read. If the user file is faulty or any error happens while reading, catch the Error and quit the program
2. Ask the user with menu for what to do. If the user inputs non-integer value, catch the error and quit the program
3. Read the csv data and allow the user to filter out missing data. Remember, first row is just the headers, so we want to drop them out from our processing.
4. Filter out all passengers that do not have age. Tell the user who were the people with missing age and how many of the them were in the data in total.
5. Filter out all passengers without a cabin. Tell the user who were the people with missing cabin and how many of them were in the data in total.
6. Calculate the average age of all the passengers, who had a proper age. First, filter out the people with missing ages and then use the passengers with the age in the calculation. NOTE: Do not print the passenger names when calculating the averages, only the amount of passengers with missing details & amount of passengers with proper details (see example run 1)

Example run 1:

```
Welcome to Titanic passenger analyzer
Enter filename for titanic data:
titanic.csv
The file contains 891 passengers

You can do the following
1) Calculate passengers with missing ages
2) Calculate passengers without a cabin
3) Calculate average age of passengers (skip missing ages)
0) Exit
Your choice: 1
# Skipping some of the data, because there is just too many to show
...
...
Razi, Mr. Raihed was missing age
Sage, Miss. Dorothy Edith "Dolly" was missing age
van Melkebeke, Mr. Philemon was missing age
Laleff, Mr. Kristo was missing age
Johnston, Miss. Catherine Helen "Carrie" was missing age
We had 177 passengers with missing age
We have 714 passengers with proper age

You can do the following
1) Calculate passengers with missing ages
2) Calculate passengers without a cabin
3) Calculate average age of passengers (skip missing ages)
0) Exit
Your choice: 2
# Skipping some of the data for simplicity’s sake
#
Rice, Mrs. William (Margaret Norton) was missing cabin
Montvila, Rev. Juozas was missing cabin
Johnston, Miss. Catherine Helen "Carrie" was missing cabin
Dooley, Mr. Patrick was missing cabin
```

```
We had 687 passengers with missing cabin
We have 204 passengers with proper cabin

You can do the following
1) Calculate passengers with missing ages
2) Calculate passengers without a cabin
3) Calculate average age of passengers (skip missing ages)
0) Exit
Your choice: 3
We had 177 passengers with missing age
We have 714 passengers with proper age
Average age of all 714 passengers was 30 years

You can do the following
1) Calculate passengers with missing ages
2) Calculate passengers without a cabin
3) Calculate average age of passengers (skip missing ages)
0) Exit
Your choice:
0
```

Example run 2: Faulty filename

```
Welcome to Titanic passenger analyzer
Enter filename for titanic data:
faulty.txt
Error processing file 'faulty.txt', stopping.
```

Example run 3: Faulty menu input

```
Welcome to Titanic passenger analyzer
Enter filename for titanic data:
titanic.csv
The file contains 891 passengers

You can do the following
1) Calculate passengers with missing ages
2) Calculate passengers without a cabin
3) Calculate average age of passengers (skip missing ages)
0) Exit
Your choice:
faulty text
Faulty user input. Exiting.
```