Ciências ULisboa

Fundamentos de Programação

Project – Assignment 3

2023/2024

The goal of the FP Project, divided by three assignments, is that students practice the subjects that they learn in classes.

In addition to topics already covered in the first two assignments, in this third assignment students will practice file input/output and exceptions.

What is your task in this 3rd assignment, after all?

In this third assignment, students are given the files `assignment3.py` and `gameFunctions.py`. The first contains part of a program, already encoded, that "calls" several of the functions in the module `gameFunctions`, allowing the user to play the bottle game as in the 2nd assignment.

We now want to add a new functionality to the game: the possibility to stop the game at any time, and store the relevant information, in order to be possible to restart playing any time in the future, from the exact point where it was interrupted.

For this purpose, students are asked to implement three additional functions described below. These functions should be inserted in the `gameFunctions` module, and called in the exact places of `assignment3.py` where indicated.

As you can see, the program is prepared both to start a new game and to restart an "old" game (a game partially played, and interrupted). All the information the program needs to (re)start the game must be fed into it by reading from a text file, which has different types of information in either case.

In the case of a new game, the exact contents of the bottles need not be fed into the program because there are functions in the `gameFunctions` module that allow to randomly fill the bottles. There are also other informations that need not be fed into the program in the case of a new game and that must be supplied in the case of an old game.

The first thing you have to decide is, for each kind of game (a new or an "old" one), the exact pieces of information the program needs to be supplied with. These decisions let you define the exact content structure of each kind of file. Two of the functions you are asked to implement must read from one of these kind of files. The third function must write a file representing an "old" game.

The functions students must implement are the following:

- `newGameInfo(fileName)` that:

- opens the file whose name is given by the string `fileName`, and reads its various contents;
- randomly generates a user expertise level, whose value must lie in a given interval defined by maximum and minimum expertise levels;
- calculates the number of bottles that should become full at the end of the game (the user's expertise subtracted from the total number of bottles to be used in this specific game);
- creates a dictionary representing the bottles information, whose keys are letters from the alphabet, and the corresponding values are randomly filled lists of symbols (from a given collection of symbols that should be used in this specific game); the function `buildGameBottles` should be used to obtain this dictionary;
- returns the values that are necessary to play this specific game.
- in case there is some problem with the file (for example, it does'nt exist, or the values it contains are not as expected), the function should raise an exception with a clarifying message.

- Function `oldGameInfo(fileName)` that:
  - opens the file whose name is given by the string `fileName`, and reads its various contents;
  - returns the values that are necessary to play this specific game;
  - in case there is some problem with the file (for example, it does'nt exist, or the values it contains are not as expected), the function should raise an exception with a clarifying message.

The above two functions must describe, as part of their documentation, the structure the file they read is expected to have.

- Function `writeGameInfo(fName, ???????)`, where students define what are the missing parameters `???????`. This function:
  - opens the file whose name is given by the string `fName`, and writes into it the values that are needed in order to, later on, restart the game, exactly from the state it is presently in.
  - Don't forget to close the file.

As you already know, you can build additional, auxiliary, functions to promote good programming.

What should students deliver?

There is no report to deliver because your *software* contains your documentation.

You should include the group number and the names and numbers of its members in the header of the `assignment3.py` and `gameFunctions.py` files. You should present your code well aligned and legible.

To deliver: A zip file containing:

- the `.py` files with your program;
- at least one text file containing the information of a new game
- at least one text file containing the information of an "old" game

The name of the *zip* file you deliver should have the format `FP3Axx.zip` (where `xx` is the number of your group). Only one of the members of the group should deliver the file.

How to deliver?

Using the FP Moodle page.

At 11:55 PM of December the 15th, the files uploaded to Moodle will be collected.

IMPORTANT NOTE: The files you deliver are assignable only and exclusively to the members of your group; Any sign of plagiarism will be investigated and it may lead to course failure of group elements and subsequent disciplinary process.