15-110 Fall 2019
Midterm Exam 03

Out: Tuesday 26th November, 2019 at 15:00 AST
Due: Tuesday 26th November, 2019 at 16:30 AST

Introduction

This midterm exam includes all the topics studied so far in the course.
The total number of points available from the questions is 115, where 15 points are *bonus* points (i.e., you only need 100 points to get the maximum grade).

Warning: if the code of a function doesn't execute because of syntax errors of *any type* (i.e., the function doesn't even reach the end) then you'll get 0 points, the function won't be evaluated at all during the grading process! This means that you should / must try out your code, function by function, before submitting it!

In the handout, the file `exam03.py` is provided. It contains the functions already defined but with an empty body (or a partially filled body). You have to complete the body of each function with the code required to answer to the questions.
You need to submit to Autolab the `exam03.py` file with your code.

Only the provided *reference cards* (possibly with your annotations) are admitted as a support during the exam.

The code must be written and tested using Spyder on the computers in the classroom.

1 Read and process experimental data

The biology research group has performed a number of experiments adopting two different techniques, labeled `Invasive` and `NotInvasive`. Each experiment has consisted of a number of trials, to account for randomness and other sources of uncertainty. The output of each trial is a floating point number.

The outcomes of the experimental campaign have been saved on a comma-separated CSV data file. The file has a header record with the names of the fields, and each data record takes the following general form:

`experiment_type, output_trial_1, output_trial_2, ..., output_trial_n`

For each experiment, a different number of trials might have been executed, meaning that different records might feature different numbers of fields.

`experiment_type` is either the string `Invasive` or `NotInvasive` (the string is never quoted in the file), while the `output_trial_` values are all floats. There might be white spaces between the fields (hint: make use of the string method `strip()` to get rid of white spaces in strings, as well as of the built-in function `float()` to cast from a string to a float). The file might include comments, that are line records starting with the character `'#'`. These comment lines should be skipped.

An example of the CSV file with the experimental data is shown below:

```
experiment , trials
# Each experiment features multiple trials
Invasive , 0.7 , 1.5
NotInvasive, 0.3 , 1.52 , 12.1
Invasive, 1.5 ,2.0 , 1.0
Invasive, 1.0 , 3.0 , 0.9 , 1.1
Invasive , 1.0 , 3.0 , 2.5
```

Problem 1.1: (35 points)

Implement the function `read_and_store(data_file, experiment_label)`. The method takes as input two strings, the name of a CSV file, `data_file`, with experiment data, and an experiment label, `experiment_label`. The file is organized as described above. The function aims to read the file and store the relevant data into a dictionary data structure, which is returned by the function.

- The file should be opened for reading only. If the file doesn't exist or isn't readable, the method should print out the message `'File␣not␣accessible'` and return `False`. Otherwise, if everything goes okay, the function just returns `True`.

- For each experiment, an *average outcome* value has to be computed out of the results of the trials. However, only the records whose label correspond to `experiment_label` should be considered.

  For instance, if the name label passed as input is `Invasive`, only the records starting with `Invasive` needs to be considered for data processing. In this case, if the file contains an experiment record such as: `Invasive, 1.5, 2.5, 3.0`

  the average outcome value for the specific experiment will be equal to $\frac{1.5+2.5+3.0}{3} = 2.33$.

- For each experiment, the average outcome value and all trial values need to be stored in the dictionary `data_dict`. This must be a dictionary data structure that uses the computed averages as keys. This means that, if the CSV file contains n data records, the dictionary will contain (at most) n entries. The "value" of each dictionary entry is a lists of floats, corresponding to the numeric values of the trials of a record (remember that the CSV read method reads a record into is a list, of strings).

For instance, for the record example above, the corresponding dictionary entry will be `{2.33: [1.5, 2.5, 3.0]}` .

- The function returns the dictionary `data_dict`.
- Once reading is completed, the input file has to be closed.

For the provided test file, the resulting dictionary is the following: `{1.1: [0.7, 1.5], 1.5: [1.5, 2.0, 1.0], 2.166: [1.0, 3.0, 2.5]}`.

Note that in this case only three entries are in the dictionary. In fact, one key is repeated, such that only the first record with average 1.5 enters the dictionary.

Problem 1.2: (40 points)

Implement the function `write_data(data_dict, experiment_label)` that saves the experiment data stored in `data_dict` into a new CSV file named with the label of the experiment type followed by `.csv`. For instance, if the experiment label `experiment_label` is `NotInvasive`, the file should be named `NotInvasive.csv`.

- The header of the CSV file must have the following text: `average, trials`.
- For each experiment, each written record must include the results from the trials and, as a first field, their average value. For instance, from the previous example, if the experiment type is `Invasive`, the record

  ```
  1.500,1.500,2.000,1.000
  ```

  will be in the written file. All numeric fields are float and must be written with *three decimal digits.*
- Moreover, the records must be written sorted in decreasing order according to their average value. Experiments with the same average value can be sorted arbitrarily relative to each other.

  For instance, given the aforementioned input data file and selected `Invasive` as experiment type, the output file should contain:

  ```
  average,trials
  2.167,1.000,3.000,2.500
  1.500,1.500,2.000,1.000
  1.500,1.000,3.000,0.900,1.100
  1.100,0.700,1.500
  ```
- The file must be created in the sub-folder `Experiment`. If the sub-folder doesn't exist, it must be created first. If the file already exists in the `Experiment` sub-folder, then the new experiment data must be appended to the existing data, otherwise a new file must be created.

  Note: this part of the question amount to 10 points. If you don't know how to manage folders, you can just create the file in the current folder. In this case 8 points will be detracted from your total scores.

- Don't forget to close the file after writing.

Problem 1.3: (35 points)

Implement the function `plot_data(data_dict)` that takes as input a dictionary with experiment data as created by the `read_and_store()` function, and displays the data in a graphical format. The function creates two graphical representations of the data, as explained below.

- The first representation consists in the plot of the average values associated to each experiment (the dictionary keys!). Each value should be displayed using a diamond red marker, no interconnecting lines between points. The plot should have an aspect ratio of 1:1 and have appropriate titles and x,y labels.
- The second representation consists in a histogram of ALL the experiment data. The histogram should be in green color and have 5 bins. Again, title, and x and y labels should be there.