Getting started

Review class handouts, executing all examples shown in them in Eclipse, as well as creating and typing examples for functions that are listed, but not illustrated with an example. This is essential for your understanding of the Pandas package functionality required to complete this assignment. Complete the reading and practice assignments posted on the course schedule.

Programming Project: Recommend

Rating-based movie recommendation.

Data and program overview

In this assignment you will be working with data on movies and people's ratings of these movies. The task will be to create movie recommendations for a person, based on the match between personal ratings and critics' ratings of the movies.

I provide two data sets for this assignment, in zip files called *data.zip* and *data-tiny.zip*. Download and unzip the files in your project folder. Unzipping should result in two folders added to your project folder: *data* and *data-tiny*.

The following data is provided in csv files:

- A table with movie information (*IMDB.csv*); we will call this the ***movies data***.
- A table with ratings of all movies listed in the movies data, by 100 critics (*ratings.csv*); let's call this the ***critics data***. The column names in the critics data correspond to the name of each critic. All ratings are integer numbers in the 1..10 range.
- A table with one person's ratings of a subset of the movies in the movies data set (*pX.csv*), the ***person data***, where X is a number. The name of the person is provided as a column title in the file.

Review the data files to familiarize yourself with their content and structure.

The program that you write must work as follows.

1. Ask the user to specify the subfolder in the current working directory, where the files are stored, along with the names of the *movies, critics*, and *person data* files.
2. Determine and output the names of three critics, whose ratings of the movies are closest to the person's ratings based on the Euclidean distance metric (as described later within definition of function `findClosestCritics()`).
3. Use the ratings by the critics identified in item 2 to determine which movies to recommend:
    - The movie recommendations must be based on the average ratings of movies by the three critics identified in step 2 above, and consist of the highest rated movies in each movie genre[1]. (see also the definition of function `recommendMovies()`).

[1] Movie genre is determined by the `Genre1` column of the movies data.

4. Display information about recommended movies as described below and illustrated by the sample interactions below.
    - Recommendations must be listed in alphabetical order by genre.
    - Missing data (e.g. running time) should not be included.

The sample interactions below demonstrate the running of the program.

Sample interactions

First, let's use the tiny data set. The interaction below uses personal data file **`tinyp.csv`** that contains movie ratings by a person named `Kimberwick`. Critics `Aldbridge, Moon, Benris` had the closest recommendations.

```
Please enter the name of the folder with files, the name of movies file,
the name of critics file,  the name of personal ratings file, separated by spaces:
data-tiny  tinyIMDB.csv tinyratings.csv tinyp.csv

The following critics had reviews closest to the person's:
Aldbridge, Moon, Benris

Recommendations for Kimberwick
"127 Hours"  (Adventure), rating: 8.0, 2010, runs 94 min
"50/50"      (Comedy), rating: 7.0, 2011, runs 100 min
"About Time" (Comedy), rating: 7.0, 2013, runs 123 min
```

The next interaction shows the output given the larger data set

```
Please enter the name of the folder with files, the name of movies file,
the name of critics file, the name of personal ratings file, separated by spaces:
data  IMDB.csv ratings.csv   p8.csv

The following critics had reviews closest to the person's:
Quartermaine, Arvon, Merrison

Recommendations for Catulpa:
"Star Wars: The Force Awakens"    (Action), rating: 9.67, 2015, runs 136 min
"The Grand Budapest Hotel"        (Adventure), rating: 9.0, 2014, runs 99 min
"The Martian"                     (Adventure), rating: 9.0, 2015, runs 144 min
"How to Train Your Dragon"        (Animation), rating: 9.67, 2010
"Kubo and the Two Strings"        (Animation), rating: 9.67, 2016
"Hacksaw Ridge"                   (Biography), rating: 9.33, 2016, runs 139 min
"What We Do in the Shadows"       (Comedy), rating: 9.0, 2014
"Prisoners"                       (Crime), rating: 8.33, 2013, runs 153 min
"Spotlight"                       (Crime), rating: 8.33, 2015, runs 128 min
"The Perks of Being a Wallflower" (Drama), rating: 9.67, 2012, runs 102 min
"Shutter Island"                  (Mystery), rating: 8.33, 2010, runs 138 mi
```

Note that in the above interaction there are sometimes more than one movie listed per genre. As, for instance, is the case with the two Adventure movies, both of them had the highest average rating, hence both are included in the list.

Important Notes and Requirements

In addition to the requirements stated so far, your code must satisfy the following to gain full credit:

- Your program should not use any global variables and should have no code outside of function definitions, except for a single call to `main`.
- All file related operations should use device-independent handling of paths (use `os.getcwd()` and `os.path.join()` functions to create paths, instead of hardcoding them).
- You must define and use functions specified below in the **Required Functions** section. You may and should define other methods as appropriate.
- You should use the `pandas` data structures effectively to efficiently achieve the goals of your functions and programs.
- The formatting of the recommendation printout should use the length of the longest movie in the list (which should be computed in the program) in formatting the output in a way that aligns categories.

Required Functions

You **must** define and use the following functions, plus define and use others as you see fit:

a. Function `findClosestCritics()` which will be used to identify three critics, whose recommendations are closest to the person's recommendations. The function should take two parameters of type DataFrame, the first one providing data about critics ratings, and the second – about personal ratings. The function must return a list of three critics, whose ratings of movies are *most similar* to those provided in the personal ratings data, based on *Euclidean distance*.

Euclidean distance of two vectors $p = (p_1, p_2, \dots, p_n)$ and $c = (c_1, c_2, \dots, c_n)$ is computed as $\sqrt{(p_1 - c_1)^2 + (p_2 - c_2)^2 + \cdots + (p_n - c_n)^2}$. To compute how similar ratings of a critic are to the ratings of the person, we compute the distance between a vector, in which the coordinates $(c_1, c_2, \dots, c_n)$ are the *critic's* ratings of each movie, and the vector composed of the *person's* ratings $(p_1, p_2, \dots, p_n)$. The lower the distance, the closer, thus more similar, the critic's ratings are to the person's.

For example, if the personal data included three ratings $(4, 7, 6)$, where the critic rated **the same movies** as $(4, 5, 6)$, the Euclidean distance would equal $\sqrt{(4-4)^2 + (5-7)^2 + (6-6)^2} = 2.0$.

Hint: for this function, create a DataFrame with critics names as its columns, movie titles as its index, and data in each column set to be the difference between the critic's and the person's score of each movie, squared. Then, calculate the sum of all column values and find the smallest three values using sorting. Return a list of the associated critics' names.

b. Function `recommendMovies ()` which will be used to generate movie recommendations based on ratings by the chosen critics. The function must accept four parameters: the critics and personal ratings data frames, the list of three critics most similar to the person, and the movie data frame. The function should determine out of the set of movies that are *not rated* in the personal data, but are *rated* by the critics, which movies have the highest average of the rating by the most similar critics in each movie genre (specified by the `Genre1` column of movie data). In other words, you need to compute

the top-rated unwatched movies in each genre category, based on the average of the three critics' ratings.

You may assume that the critics data will always be complete, i.e. will include ratings of all movies.

The function must then (a) put together information about these top-rated movies sufficient to produce the printout, showing the details of each of the recommended movies as illustrated by the interactions, and (b) return it using some data structure of your choice.

Hint: An easy way to generate a list of unwatched movies with all critics' ratings is to merge the person data with the critics data and select the portion of it, which has missing values in the person's column. After that, to find the highest rated movies per genre, first join the resulting file with the movies data to have full movie information. In this joint DataFrame, compute the average rating by the three critics (storing the average in a new column) and then select the highest rating in each genre (using `groupby`). After that, select those movies in each genre, that have the rating not lower than the highest in the genre.

c. Function `printRecommendations ()` with two parameters: the first containing information about the recommended movies, and the second – the name of the person, for whom the recommendation is made (the name is specified in the header of the personal ratings data file). The function must produce a printout of all recommendations passed in via the first parameter, in alphabetical order by the genre, as shown in the sample interactions. Make sure to examine the sample interactions and implement the details of the printout. The function should return no value.

d. Function main(), which will be called to start the program and works according to your design.

More Hints

- For some csv data, you may not need all of the columns. You can specify which columns to import into a data frame, or you could drop unnecessary ones to improve performance and simplify testing and debugging.
- Keeping your data frames indexed by the title should help in making joins easy. Note that the title can be both an index and one of the columns, if necessary.
- Make sure to inspect intermediate results via printing them or saving to files. To see the data frames completely when they are printed, you can include the following function calls in your program to set display parameters for pandas

```
pd.set_option('display.max_columns', 1000)
pd.set_option('display.max_rows', 1000)
pd.set_option('display.width', 1000)
```

- When you open and save the csv files in Excel, Excel may change the encoding used for file characters. To return the encoding to the UTF-8 standard that Python likes, open the file in Notepad, and when saving it, specify the encoding as UTF-8.
- Although I have provided a sample small data sets, for testing purposes, I encourage you to create your own one, for which you should know the result in advance.

Grading

The grading schema for this project is roughly as follows:

Five points will be awarded for the correct implementation of each of the four functions above (which may call other functions that you define), which uses data structures, methods and functions of the `pandas/numpy` package appropriately and effectively.

Three points will be awarded for making the code sufficiently general to handle different input files, i.e. not tied to the specific content of the files that you are given (though it might be somewhat dependent on the *structure* of those files, i.e. what is provided by the columns, rows, etc.)

Two points will be awarded for style, as defined by the guidelines in Handout 1.