Homework 7 - File I/O & CSV

CS 1301 - Intro to Computing - Spring 2020

Important

- Due Date: **Tuesday, March 10th, 11:59 PM.**
- This is an individual assignment. High-level collaboration is encouraged, **but your submission must be uniquely yours.**
- Resources:
  - TA Helpdesk
  - Email TA's or use class Piazza
  - How to Think Like a Computer Scientist
  - CS 1301 YouTube Channel
- Comment out or delete all function calls. Only import statements, global variables, and comments are okay to be outside of your functions.
- **Read the entire document before starting this assignment.**

Responsible Computing: Some functions in this assignment will have a 'Responsible Computing' badge next to them. These functions are written to help you think about how you can use your skills as a programmer to analyze problems related to *sustainability*, such as ethics, health, security, and the environment.

Hidden Test Cases: In an effort to encourage debugging and writing robust code, we will be including hidden test cases on Gradescope for some functions. You will not be able to see the input or output to these cases. Below is an example output from a failed hidden test case:

```
Test failed: False is not true
```

Written by Peter Han (phan38@gatech.edu) & Caitlin Yang (caitlinyang@gatech.edu)

PART 1: File I/O

For the Part 1 File I/O functions, the .txt file being read from will be formatted as seen below. Be sure to download the provided file to the same directory as your `HW07.py` file. To open the .txt file, use notepad for Windows and text edit for Mac.

You are working on a file that contains data on the 2020 Academy Awards. The file contains all the categories that films and actors can be nominated for, and it contains their *Standing* meaning if they were *Nominated* or the *Winner* of that category. It is important to note that when the category is for a person, such as Best Director, they will have that person's name and the movie that they were associated with in parenthesis as seen below.

```
Category Name
Movie
Standing

Category Name
Person (Movie)
Standing

...
```

count_nominations

Function Name: count_nominations
Parameters: filename (`str`), category(`str`)
Returns: A `tuple` with the first element being the winning movie name (`str`) and the second element being the number of total nominations (`int`)
Description: After watching the Oscars, you were curious as to how many nominations each winning film received in total. Given a category, first find the winning film of that category (you'll only be given a film category and not a person category). Then, go through all the nominations and count how many times that film was nominated. Note: If the film was a winner, it should still be considered as being nominated for the award.

A film should be counted even if that category was a person category. For instance, if the winning film was 'Interstellar' and in your file, the category was Best Lead Actor:

```
Best Lead Actor
Matthew McConaughey (Interstellar)
Nominated
```

it would still add one to the total count.

If an empty string was passed in as the category, return an empty tuple.

```
>>> cat1 = 'Best Sound Editing'
>>> print(count_nominations('academyawards.txt', cat1))
('Ford v Ferrari', 4)

>>> cat2 = 'Best Live Action Short Film'
>>> print(count_nominations('academyawards.txt', cat2))
('The Neighbors' Window', 1)
```

---

categories

Function Name: categories
Parameters: filename (`str`), categoryList(`list`)
Returns: a `dictionary` whose keys are the categories and values are `lists` containing `tuples`
Description: Given a list of Oscar categories, your job is to create a dictionary whose keys are the category names and whose value is a list of tuples containing the nominees. If the category is a person category (ex: Best Director), the tuple must have two elements with the first element being the person's name and the second element being the movie they played a role in. Otherwise, the tuple should be a single item tuple containing the film's name. If the category list is empty, return an empty dictionary.

```
>>> categoryList = ['Best Live Action Short Film','Best Supporting Actress']
>>> print(categories('academyawards.txt', categoryList))
{'Best Supporting Actress': [('Kathy Bates', 'Richard Jewell'),
                             ('Laura Dern', 'Marriage Story'),
                             ('Scarlett Johansson', 'Jojo Rabbit'),
                             ('Florence Pugh', 'Little Women'),
                             ('Margot Robbie', 'Bombshell')],
 'Best Live Action Short Film': [('Brotherhood',),
                                 ('Nefta Football Club',),
                                 ('The Neighbors' Window',),
                                 ('Saria',), ('A Sister',)]}

>>> categoryList2 = ["Best Original Screenplay"]
>>> print(categories('academyawards.txt', categoryList2))
{'Best Original Screenplay': [('Knives Out',),
                              ('Marriage Story',),
                              ('1917',),
                              ('Once Upon a Time in Hollywood',),
                              ('Parasite',)]}
```

---

winner_format

Function Name: winner_format
Parameters: readfile (`str`), writefile (`str`), category(`str`)

Returns: `None`
Description: Given a specific category, you want to write a file that contains the category name and a numbered list of the nominees for that category. The numbered list should be tabbed over by one tab `'\t'`. The first movie (#1) should be the winning movie of that category and it should have the string `*Winner*` in front of it. The rest of the movies should come in the order that they appear in the file. An empty category name will never be passed in.

```
>>> writefile = 'category.txt'
>>> category = 'Best Picture'
>>> winner_format('academyawards.txt', writefile, category)
```

Your file `category.txt` should look like this and should not have an extra `'\n'` at the end:

```
Best Picture
  1. *Winner* Parasite
  2. Ford v Ferrari
  3. The Irishman
  4. Jojo Rabbit
  5. Joker
  6. Little Women
  7. Marriage Story
  8. 1917
  9. Once Upon a Time in Hollywood
```

PART 2: CSV

For the Part 2 CSV functions, the .csv file format will have a header row at the top of the .csv file. Be sure to download the provided csv file to the same directory as your `HW07.py` file. To open the .csv file, use notepad for Windows and text edit for Mac instead of using Excel or Google Sheets.

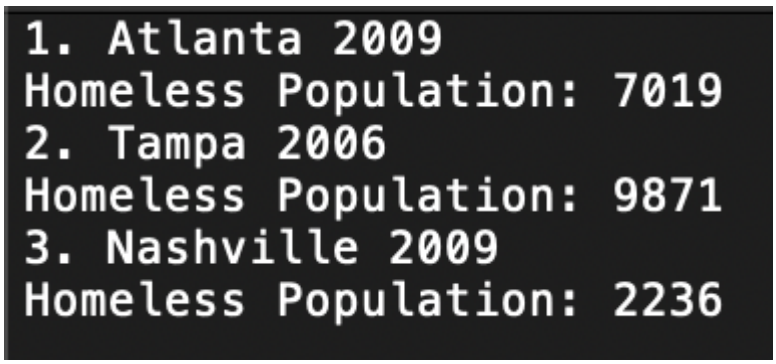
data_reformatter Responsible Computing

Function Name: data_reformatter
Parameters: cities (`list`), years (`tuple`)
Returns: None
Description: The U.S. Department of Housing and Urban Development has kept track of the homeless population in several southeastern metropolitan centers from 2005 - 2015. The data was put into a .csv file (`homeless_2005_2015.csv`) that can be analyzed. Given a list of cities and a range of years (inclusive), create a new .txt file called `homeless_population.txt` that is a list of the highest population year and their corresponding population within the given range for each of the cities. The cities should be ordered as they are in the argument.
.

If the date range is invalid (first date is greater than second date) return "Invalid Input". Years outside of 2005-2015 should be ignored as they are not present in the data set. Cities not present in the data set should be ignored, as shown in test case below.

```
>>> cities1 = ['Atlanta', 'Tampa', 'Nashville']
>>> years1 = (2004, 2009)
>>> data_reformatter(cities1, years1)
```

The output file should be named `homeless_population.txt` and have this exact format without an extra '\n' character at the end. 2009, 2006, and 2009 are the years of the highest homeless population for Atlanta, Tampa, and Nashville from 2004-2009.




```
>>> cities2 = ['Orlando', 'New York', 'Miami']
>>> years2 = (2006, 2011)
>>> data_reformatter(cities2, years2)
```

The output file should be named `homeless_population.txt` and have this exact format wihtout an extra '\n' character at the end and it should replace the contents from the previous function call.

```
1. Miami 2006
Homeless Population: 4709
```

---

homeless_rate Responsible Computing

Function Name: homeless_change
Parameters: city1 (`str`), city2 (`str`), dates (`tuple`)
Returns: A `tuple` where first element is the city with the greatest reduction of the homeless population (`str`) and the second is the value of the change (`int`)
Description: The U.S. Department of Housing and Urban Development has kept track of the homeless population in several southeastern metropolitan centers from 2005 - 2015. The data was put into a .csv file (`homeless_2005_2015.csv`) that can be analyzed. You would like to compare which cities have decreased their homeless population the most between different years. Given two cities, find the difference between the two years and return the city with the most reduction in homeless.

If the date ranges are invalid (out of range of the dataset, first date is greater than second date, etc.) or if a city doesn't exist, return 'Invalid Input'.

```
>>> city1 = 'Tampa'
>>> city2 = 'Atlanta'
>>> dates1 = (2005, 2007)
>>> print(homeless_change(city1, city2, dates1))
('Tampa', -3388)

>>> city3 = 'Orlando'
>>> city4 = 'Atlanta'
>>> dates2 = (2005, 2007)
>>> print(homeless_change(city3, city4, dates2))
'Invalid Input'
```

Grading Rubric

Function	Points
count_nominations()	15
categories()	15
winner_format()	20
data_reformatter()	25
homeless_rate()	25
Total	**100**

Provided

The `HW07.py` skeleton file has been provided to you. This is the file you will edit and implement. All instructions for what the functions should do are in this skeleton and this document.

Submission Process

For this homework, we will be using Gradescope for submissions and automatic grading. When you submit your `HW07.py` file to the appropriate assignment on Gradescope, the autograder will run automatically. The grade you see on Gradescope will be the grade you get, unless your grading TA sees signs of you trying to defeat the system in your code. You can resubmit this assignment an unlimited number of times until the deadline; just click the "Resubmit" button at the lower right-hand corner of Gradescope. You do not need to submit your `HW07.py` on Canvas.