Practicum 6: Baseball Leaderboards

Due Date: Monday, October 18th at 11:59pm EST

Statistical analysis of the sport baseball—also known as ”sabermetrics”—took off in the 2000s when the Oakland Athletics, a low-budget baseball team, successfully and surprisingly outperformed other teams with much larger budgets to buy high-ranking and high potential players. Their success was popularized in the non-fiction book *Moneyball*. The quality of data in baseball is generally quite high; play-by-play data is available for virtually every game since the 1930s. The sport also lends itself to statistical analysis better than other sports like basketball and football because it is easier to divide into discrete events and quantify.

This week, we will use the Lahman baseball dataset. Each row is a player's batting data for a given year. **Download the data** and place it *in the same folder* as the `practicum06.py` file.

We'll use the baseball data to practice the following:

1. Write functions
2. Work with lists of lists
3. Filter data
4. Sort data

Note: I have tried to write this assignment in such a way that you should be able to complete it even if you are not familiar with anything about baseball. If I haven't made something clear about the sport and what I am asking you to do, please don't hesitate to ask for clarification.

There is a codebook describing the column values on the last page of this assignment.

1 Reading the Data with Lists of Lists

In previous weeks, our strategy has typically been to read a file, and while we go through each line, save and calculate data as needed. This has been sufficient for what we've needed to do, but often it's easier to read the file once and save all the data. This lets us avoid reading the file multiple times if we need to do multiple calculations; we can refer to the data we saved.

To do this, we are going to need to work with lists of lists. Just like how we can put integers, floats, or strings into a list, *we can also put other lists into a list*. For example, we can make a list that contains lists that each have two things in them, a name and an age:

```
list_of_lists = [['Ryan', 28], ['William', 26], ['Nancy', 30]]
```

We can access each individual list, and each element of each individual list, by using the same index operations that we have been using regularly for lists.

```
# Get the very first list
first_list = list_of_lists[0]
print(first_list)

# Get the first item (the name) from the second list
name2 = list_of_lists[1][0]
print(name2)
```

Write a function with the following signature.

```
Name: read_file
Description: Reads and returns data from a file

Parameters
----------
file_name: str
    The name of a CSV file to read

Returns
-------
data: list of lists
    Each sublist includes the data for a row from the file

Example
-------
list_of_lists_data = read_file('lahman_batting.csv')
```

Tips and Hints

- This should look very similar to the code we used last week to read a file. Use a for loop to go over each line, strip the line, split the line, and append the data. Remember to use enumerate and skip the header.
- Please convert the types of each data appropriately. **I have provided you with a function (`process_line`) in the provided template to make this easier.**

2 Filtering Numeric Data

The data consists of a number of numeric columns, including the year of the game and counts of various outcomes. We are often interested in subsets of data. For example, we may want to only study seasons since the 1990s or players with at least 200 hits.

Write a function with the following signature. It should be able to filter by any numeric column, not just one particular column. The return value is a list of lists, where the sublists are the rows that match the criteria.

```
Name: filter_by_numeric
Description: Filters numeric data by a particular column index according to
whether it is within the range of a given min and max

Parameters
----------
data: list of lists
    Each sublist contains row data
col_indx: int
    The index of the column to use
min_val: int
    The minimum value to filter by
max_val: int
    The maximum value to filter by

Returns
-------
filtered_data: list of lists
    Each list includes the data for a row, where the value specified by col_indx
    is between min_val and max_val

Example
-------
data2000s = filter_by_numeric(data, 2, 2000, 2021)
```

Evaluate the following to check that your function is correctly written. Put these checks in your `main()` function.

- Almost all of the data are some type of count. So if the minimum value is -1 and the maximum value is something unreasonably large (like 10,000), then the function should return the whole dataset. Provide a case like this, and print out the *length* of the resulting data (not the data itself). There should be 15,630 rows.
- Check that the record for hits (H, index 7) is by Ichiro Suzuki in 2004 with 262 hits.
  i.e. Check that there is only one record where hits >= 262.
- Check that the record for walks (BB, index 10) is by Barry Bonds in 2004 with 232 walks.
  i.e. Check that there is only one record where walks >= 232

3 Filtering Categorical Data

There are a few columns in the dataset that are strings and that we may also want to filter by.
Write function with the following signature:

```
Name: filter_by_categorical
Description: Filters categorical data (i.e. strings) by a particular column index
according to whether it is in a list of given values

Parameters
----------
data: list of lists
    Each sublist contains row data
col_indx: int
    The index of the column to use
category_vals: list of strs
    A list of strings to filter by

Returns
-------
filtered_data: list of lists
    Each list includes the data for a row, where the value specified by col_indx
    is in category_vals

Example
-------
# Data for the Boston Red Sox and Chicago White Sox
data_sox = filter_by_categorical(data, 4, ['BOS', 'CHA'])
```

Evaluate the following to check that your function is correctly written. Put these checks in your `main()` function.

- There are two baseball leagues: the Atlantic League (AL) and the National League (NL). Make sure that if you "filter" by these values, all of the records are returned. There should be 15,630 rows.
- The Montreal Expos (MON) were shut down as a team in 2004 and replaced by the Washington Nationals (WAS). Count the number of rows there are for these two teams together. There should be 544 rows.

4 Sorting and Leaderboards

Python has a built-in function `sorted()`. It takes a simple list and, unsurprisingly, sorts it.

```
nums = [5, 3, 10, 1]
sorted_nums = sorted(nums)
print(sorted_nums)
```

If we have a list of lists though, the behavior may be unexpected. It will sort by the first item in every list, and on any ties, use the second item, and so on if there are more ties. For example, if

we had the example list at the start of this assignment with names and ages, Python would first sort by name, and then by age if there were any people with the same name.

If we want to sort by only one value, then we need to use the `key` parameter and the special `lambda` function. For example, if we want to sort by the number of hits (H, index 7), we can do what is shown below. Note, I am about to use syntax you have not covered: `lambda`. In short, it is defining a very simple function called `lambda` (which is a special name) with a single parameter, `row`. **Take it as a given, and focus on where the index is used. We will discuss this in future weeks.**

```
sorted_hits = sorted(data, key=lambda row: row[7])
```

Finally, by default, the lists are sorted in ascending order (smallest to largest). If we want a leaderboard, we often want the largest values. We just need to use the `reverse` parameter (which is `False` by default).

```
leaderboard_hits = sorted(data, key=lambda row: row[7], reverse=True)
```

Now, let's bring this all together and write a function that returns the leaderboard for a given metric (hits, walks, etc) during a single season. If we use our numeric filter, we can make sure the leaderboard is for a particular season, not all time. Then we can sort it, and get the top records by slicing the list. Write a function with the following signature.

```
Name: get_season_leaderboard
Description: Gets the top players by some metric for a single season

Parameters
----------
data: list of lists
    Each sublist contains row data
col_indx: int
    The index of the column to use
top_n: int
    The top N players in the leaderboard
year: int
    The year of the season

Returns
-------
leaderboard: list of lists
    Each list includes the data for a row, where each is in the top N for a
    given season according to the metric specified by col_index

Example
```

```
-------
# Get the top 20 leaderboard for hits in 2018
leaderboard = get_season_leaderboard(data, 7, 20, 2018)
```

Evaluate the following to check that your function is correctly written. Check that the top 5 leaderboard for home runs in 2018 looks like the following:

```
Khris Davis: 48 home runs
J. D. Martinez: 43 home runs
Joey Gallo: 40 home runs
Jose Ramirez: 39 home runs
Mike Trout: 39 home runs
```

5 Submit Your Work

When you are done, submit the following to Canvas in a zip folder:

1. The Python file (.py) containing your code answers for this practicum.
2. The baseball data file (.csv)
3. A text file (.txt, .doc, .pdf, etc.) with any comments about anything you were not able to get working and what you tried to solve it.

Please name your zip folder `LastName_Practicum06.zip`.

This assignment was originally created by Stefan McCabe and Carolina Mattsson.
It has been updated here by Ryan Gallagher.

Codebook

Index	Header	Detail
0	first_name	

1	last_name	
2	year	Year of the season
3	stint	The *n*th team that someone played for this season
4	team	Team name (short ID)
5	league	The league that the team plays in (AL or NL)
6	AB	Times at bat
7	H	Number of hits
8	HR	Number of home runs
9	SB	Number of stolen bases
10	BB	Number of walks
11	SO	Number of strikeouts
12	HBP	Number of times hit by a pitch
13	PA	Number of times appearing at the plate