Lab Exercise #15

Assignment Overview

This lab exercise provides practice with Pandas data analysis library.

Data Files

We provide three comma-separated-value file, `scores.csv` , `college_scorecard.csv`, and `mpg.csv`. The first file is list of a few students and their exam grades. The second file includes data from 1996 through 2016 for all undergraduate degree-granting institutions of higher education. The data about the institution will help the students to make decision about the institution for their higher education such as student completion, debt and repayment, earnings, and more. The third file includes information about the fuel economy of a list of vehicles.
In the Part A of the lab, we want to compare the execution time required to read the file, and to calculate the mean for a specified column. It's better to practice with `scores.csv` that has only a few lines and then switch to the `college_scorecard.csv` to calculate the execution times.
In Part B, we will use pandas exclusively using the `mpg.csv.`

Pandas

In this lab, we are using a few of the tools provided by the Pandas library. To read a CSV file, Pandas has a method called `read_csv`. It has many parameters, but in this lab we use only the filename. The method reads the CSV file into a DataFrame object. A DataFrame is a two dimensional tabular data structure. Here is an example using the `scores.csv` file:

```
data_frame = pandas.read_csv("scores.csv")

print(data_frame)
```

```
     First       Last  Exam1  Exam2  Exam3  Exam4
0    Grace     Hopper    100     98     87     97
1   Donald      Knuth     82     87     92     81
2    Adele   Goldberg     94     96     90     91
3    Brian  Kernighan     89     74     89     77
4  Barbara     Liskov     87     97     81     85
```

You can get the column titles by `data_frame.columns`:

```
Index(['First', 'Last', 'Exam1', 'Exam2', 'Exam3', 'Exam4'],
dtype='object')
```

You can also find the size of the table by `data_frame.shape`:

```
(5, 6)
```

Subsetting is the process of retrieving parts of a data frame. A data frame can be subset in a variety of ways, the most common of which involve selecting a range of rows and columns, or selecting columns by column label. To select a column or columns of a data frame, the command `data_frame["column"]` where `data_frame` is the `DataFrame object` and `column` is the

name of the selected column. The column name is enclosed within single or double quotes. Alternatively, the command `data_file.column_title` can be used as long as the column name is not the same as an existing method. The data type of the output is a series, which can be thought of as a list with labels. For example `data_file.Exam1` returns only exam 1 scores:

```
0    100
1     82
2     94
3     89
4     87
Name: Exam1, dtype: int64
```

If you have the column index, you can use it to extract the column title, then use it to index the `data_frame` to access only one column:

```
data_frame[data_frame.columns[0]]
```

```
0      Grace
1     Donald
2      Adele
3      Brian
4    Barbara
Name: First, dtype: object
```

When you create a DataFrame, you have the option to add input to the 'index' argument, to make sure that you have the index that you desire. You can also reshape the dataframe after creation such that it has an index that you desire using the method `set_index()`. For example,

```
data_frame = pandas.read_csv("scores.csv")
data_frame.set_index('Last', inplace=True)
print(data_frame)
```

```
             First  Exam1  Exam2  Exam3  Exam4
Last
Hopper       Grace    100     98     87     97
Knuth       Donald     82     87     92     81
Goldberg     Adele     94     96     90     91
Kernighan    Brian     89     74     89     77
Liskov     Barbara     87     97     81     85
```

`inplace` means do not create a copy of it. It will instead modify it. Note the difference between the printed data frame above and in the previous page. You can find the commonly used methods below. These methods could be applied on a column by calling these methods for a specific column:

DataFrame.method	Description of output
describe()	Summary statistics for numerical columns
head(), tail()	First/last 5 rows in the DataFrame
min(), max()	Minimum/maximum of values in a numerical column
mean(), median()	Mean/median of values in a numerical column
std()	Standard deviation of values in a numerical column
set_index()	Set the DataFrame index using existing columns.

For example,

```
data_frame = pandas.read_csv("scores.csv")
print("Median: ", data_frame['Exam1'].median())
print("Mean: ", data_frame['Exam1'].mean())
Median:  89.0
Mean:  90.0
```

Import Pandas and try the code above to experiment with reading and processing a CSV file.

Part A – READING A CSV FILE AND PERFORMING SIMPLE OPERATIONS

To find out how long your Python program is taking to execute some tasks, you can use the Time module. `Time.time()` returns the number of seconds since January 1970. If you call it before and after a function call, and subtract the values, you can find out how long it took the system to execute your function.

Description

Download the starter code `'lab15a.py'`. Write a program that reads the file `'college_scorecard.csv'`. In this lab, you should implement two functions for reading and loading the CSV into a data structure, and two functions for finding the median of column index 1(`" OPEID" column)`. The goal is to compare the execution times together. You should implement the following functions and report how long they take to execute:

1. `read_csv_1(filename)`:
   This function uses `csv.reader` to read the file and saves the file data in a list of lists, then returns it.

2. `read_csv_2(filename)`:
   This function uses `pandas.read_csv` to read the CSV file and returns a DataFrame.

3. `find_median_1(data, index)`:
   This function receives a list of lists as input and calculates and returns the median of a column `index` rounded to 2 decimals. To find the median, we first need to reorganize our data set in ascending order. Then the median is the value that coincides with the middle of the data set. If there are an even amount of items, then we take the average of the two values that would "surround" the middle.

4. `find_median_2(data_frame, column_name)`:
   This function receives a DataFrame as input and calculates and returns the median of the column `column_name` using Pandas library rounded to 2 decimals. Use the `.median()` method.

★ **Demonstrate your completed program to your TA. On-line students should submit the completed files (named "lab15a.py") for grading via Mimir. They also should submit a text file describing their observation about how long the functions takes.**

Part B – EXTRACTING SUBSET OF DATA

Download the starter code `'lab15b.py'`. Write a program that reads the file `'mpg.csv'` using the pandas `read_csv()` method and read the contents of the file into a DataFrame.
Using DataFrame methods, output the:

- The subset of the first 5 rows of columns titled mpg and horsepower
- The subset of the last 5 rows of the columns titled `mpg, horsepower, model_year, and name`
- The median of the `"acceleration"` column
- The US cars with the highest mpg (best fuel economy). The origin of the vehicle models is defined in the column `"origin".` The model of the cars are defined in the `"name"` column.

In this lab, you should define the following functions. You should also create a function that finds the best fuel economy based on its origin and prints the name of the vehicle that has that fuel economy.

1. `read_csv_2(filename)`:
   This function uses `pandas.read_csv` to read the CSV file and returns a DataFrame.

2. `find_median_2(data_frame, column_name)` → median
   This function receives a DataFrame as input and calculates and returns a `float` which is the median of the column `column_name` using Pandas library rounded to 2 decimals.

3. `find_highest_mpg(data_frame, country) -- > (car_name,high_mpg)`:
   This function receives a `DataFrame` and `country` as input and returns the model of the `country` cars (`str`) with the highest mpg and the highest mpg (`float`) using Pandas library.

To select more than one column or make the output a data frame rather than a list, double brackets should be used. For example,
`data_frame[["column1", "column2", ...]]` returns a data frame with `column1, column2, ...` included. For example,

```
data_frame = pandas.read_csv("scores.csv")
print(data_frame.loc[['Exam1', 'Exam2']])
```

```
   Exam1  Exam2
0    100     98
1     82     87
2     94     96
3     89     74
4     87     97
```

Similar to lists, to select a row by position, the command `data_frame[a:b]` where `a` and `b - 1` are the initial and final rows included in the output.

The `loc[]` is used to select a range of rows and/or a subset of columns. For example, the following lines returns a dataframe containing the first 4 rows and the columns `'Exam1', 'Exam2'`of the data frame data set as shown below. Note the 0:3 are the labels in the column index.

```
data_frame = pandas.read_csv("scores.csv")
print(data_frame.loc[0:3,['Exam1','Exam2']])
```

```
   Exam1  Exam2
0    100     98
1     82     87
2     94     96
3     89     74
```

To extract a subset of a DataFrame based on the values in another column, you can define your row indices based on Boolean condition. For example,

```
print(data_frame.loc[data_frame["mpg"] == 18,["mpg","name"]])
      mpg                        name
0    18.0   chevrolet chevelle malibu
2    18.0           plymouth satellite
16   18.0                   amc hornet
37   18.0                 amc matador
45   18.0  amc hornet sportabout (sw)
48   18.0                ford mustang
76   18.0             volvo 145e (sw)
97   18.0            plymouth valiant
99   18.0                   amc hornet
100  18.0               ford maverick
107  18.0                 amc gremlin
111  18.0                  maxda rx3
135  18.0  plymouth satellite sebring
153  18.0              chevrolet nova
163  18.0               plymouth fury
174  18.0                  ford pinto
```

★ **Demonstrate your completed program to your TA. On-line students should submit the completed files (named "lab15b.py") for grading via Mimir.**