Purpose

This homework will give you practice using two very important Python modules for data manipulation: Numpy and Pandas. Numpy is a powerful library for performing mathematical and logical operations on Arrays. It provides an abundance of useful features for operations on n-arrays and matrices in Python. Pandas builds upon Numpy adding even more features and the ability to handle tabular data from nonhomogeneous types. Both of these modules are commonly used by data scientists and analysts.

Skills

After learning Numpy and Pandas, you'll be able to input, clean, and aggregate large quantities of data, and then use that data with other Python modules such as SciPy that is used for statistical analysis or Matplotlib that is used for visualizing the data.

Knowledge

Knowledge of these two very powerful modules will add to your overall Python and data manipulation skills. When given a new data set in your future career, you will have some powerful options on how to find answers to questions about the data.

Important

1. Due Date: **04/01/2021 at 11:59 pm**
2. This homework is graded out of **100** points.
3. This is an individual assignment. You may collaborate with other students in this class. Collaboration means talking through problems, assisting with debugging, explaining a concept, etc. Students may only collaborate with fellow students currently taking CS 2316, the TA's and the lecturer. You should not exchange code or write code for others. For individual assignments, each student must turn in a unique program. Your submission must not be substantially similar to another student' s submission. Collaboration at a reasonable level will not result in substantially similar code.
4. For Help:
    - TA Helpdesk (Schedule posted on class website)
    - Email TA's or use Ed Discussion Notes
    - [Numpy Reference Guide PDF, Numpy User Guide PDF]
    - [Pandas Documentation]
    - Handouts
5. Comment out or delete all your function calls. Only global variables, and comments are okay to be outside the scope of a function. When your code is run, all it should do is run without any errors.
6. HAVING FUNCTION CALLS OR EXTRANEOUS CODE OUTSIDE THE SCOPE OF FUNCTIONS WILL RESULT IN AN AUTOMATIC 0.

7. IF YOUR CODE CANNOT RUN BECAUSE OF AN ERROR, IT IS A 0%

Introduction

Please read through the entire document before starting this homework. The goal of this homework is to demonstrate your understanding of Numpy and Pandas by using Numpy arrays and Pandas dataframes. You will be defining a few functions that create and manipulate arrays and dataframes. You should test them out first on your own computer, then when you have one or more of them working, upload the entire file (which must be named HW03.py) to GradeScope to see how well your code performs on the test cases we have created. The score shown on Gradescope may not reflect your final grade on this assignment. You can submit the homework file as many times as you'd like before the deadline.

Allowed imports: `numpy, pandas`

One-line Formatting

Correct:

```
def function_a(param1):
    return [i for i in param1]
```

or

```
def function_a(param1):
    return type(param1)
```

Incorrect:

```
def function_a(param1):
    val = [i for i in param1]
    return val
```

or

```
def function_a(param1):
    return helper(param1)

def helper(param1):
```

```
    return [i for i in param1]
```

Functions

Note: The grading rubric at the end of the document for point distribution.

Function name: `spring_weather`
Parameter(s): `temperatures` (np.array), `minTemp`(int), `maxTemp`(int)
Return Type: int
Description: It's the start of spring, and you're curious what the average weather is like this month. Write a function that takes in `temperatures` (a np.array representing the temperature each week) and finds the average temperature for all values that are greater than or equal to `minTemp` and less than `maxTemp`.

There is a one–line maximum requirement.

Python Test Cases

```
>>> temperatures = np.array([[70, 85, 58, 61, 68, 80, 77],
                             [68, 78, 64, 70, 43, 77, 59]])
>>> spring_weather(temperatures, 60, 80)
    70.33333333333333

>>> temperatures = np.array([[72, 81, 55, 56, 86, 53, 75],
                         [71, 79, 79, 82, 87, 70, 69],
                         [74, 49, 77, 88, 91, 76, 88]])
>>> spring_weather(temperatures, 70, 85))
    76.0
```

Function name: `random_sale`
Parameter(s): `items` (np.array), `cost` (np.array), `tax` (np.array), `discount` (np.array) `budget` (int)
Return Type: np.array

Description:
Write a function that takes in `items` (a np.array of items), `cost` (a np.array of the cost of each items in `items`), `tax` (a np.array of the tax rate corresponding to each item in `items`), `discount` (a np.array of the discount corresponding to each item in `items`) and the `budget` (an int). It is only possible to purchase items if their **total cost (cost including tax minus discount) is under the budget**. Return a np.array of all the items that can be purchased. Each item corresponds to the same index for all arrays.

There is a one–line maximum requirement.

Python Test Cases

```
>>> items = np.array(['apple pie',
           'ben and jerry's mint chocolate chunk',
           'root beer float',
           'tiramisu',
           'cannoli',
           'trader joe's coffee ice cream',
           'chocolate chip cookie',
           'oatmeal raisin cookie'])
>>> cost = np.array([15, 6, 6, 12, 7, 7, 6, 6])
>>> tax = np.array([0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1])
>>> discount = np.array([2,3,2,1,0,1,2,3])
>>> random_sale(items, cost, tax, 5)
np.array(["ben and jerry's mint chocolate chunk" 'root beer float'
 'chocolate chip cookie' 'oatmeal raisin cookie'])
```

Function name: `no_null`
Parameter(s): `data` (np.array)
Return Type: np.array

Description:
Write a function that takes in `data` (a np.array of the scores). The function should return a np.array where the np.nan values **are replaced with the mean of all values that are NOT np.nan**.

Hint: Use `np.isnan()`

There is a one–line maximum requirement.

Python Test Case

```
>>> data = np.array([92, np.nan, 99, np.nan, np.nan,
                     80, np.nan, 100, np.nan, 78])
>>> no_null(data)
np.array([92.,  89.8,  99.,   89.8,  89.8,
          80.,  89.8,  100.,  89.8,  78.])

>>> data = np.array([84, 270, 94, np.nan, np.nan,
                     82, np.nan, 100, np.nan, 18])
>>> no_null(data)
np.array([84., 270.,  94., 108., 108.,
```

```
           82., 108., 100., 108.,  18.])
```

Function name: `wordle`
Parameter(s): player1 (np.array), player2 (np.array)
Return Type: bool
Description: You and your friend never miss the daily wordle. In order to see who is the superior player, you decide to compare your lowest weekly total guesses. Given `player1` and `player2` (np.arrays with each row representing a different week and each column representing the number of guesses it took per day), return True if your (`player1`'s) lowest total weekly guesses is less than your friend's (`player2`'s) and False otherwise.

There is a one–line maximum requirement.

Python Test Ca

```
>>> player1 = np.array([[4,4,4,2,6,6,6],
                        [3,2,5,5,5,2,2],
                        [1,2,1,5,4,3,3]])
>>> player2 = np.array([[6,2,3,3,5,4,6],
                        [5,3,3,2,1,4,2],
                        [1,5,4,3,6,2,4]])
>>> wordle(player1, player2)
    True
```

Function name: `csv_parser`
Parameters: `filename` (str)
Return Type: pd.DataFrame
Description: Write a function that takes in the `filename` as a string and returns the Pandas DataFrame representing the corresponding csv file.

There is a one–line maximum requirement.
Do not hardcode the filename. The csv file that will be entered as a parameter is titled cars.csv and contains the following information about each car:

Column name	Data Type	Description
vehicle_id	string	The vehicle identification number is a collection of 17 characters (digits and capital letters)
price	int	Sale price of vehicle in ad
brand	string	Brand of car

model	string	Model of car
year	int	Car registration year
condition	string	Either ‘clean vehicle’ or ‘salvage insurance’
mileage	float	Miles the car has traveled
color	string	Color of the vehicle
lot	int	Identification number assigned to a particular quantity or lot of material from a single manufacturer.
state	string	The state in which the car is being available for purchase
country	string	The country in which the car is being available for purchase. Either ‘usa’ or ‘canada’.
bidding_time	string	Time remaining to bid on the vehicle

Python Test Case:

```
>>> df = csv_parser("cars.csv")
>>> df
              vehicle_id  price      brand  ...       state  country   bidding_time
0       jtezu11f88k007763   6300     toyota  ...  new jersey      usa   10 days left
1       2fmdk3gc4bbb02217   2899       ford  ...   tennessee      usa    6 days left
2       3c4pdcgg5jt346413   5350      dodge  ...     georgia      usa    2 days left
3       1ftfw1et4efc23745  25000       ford  ...    virginia      usa  22 hours left
4       3gcpcrec2jg473991  27700  chevrolet  ...     florida      usa  22 hours left
...                   ...    ...        ...  ...         ...      ...           ...
2494    3n1cn7ap9kl880319   7800     nissan  ...  california      usa    1 days left
2495    3n1cn7ap5jl884088   9200     nissan  ...     florida      usa  21 hours left
2496    3n1cn7ap9jl884191   9200     nissan  ...     florida      usa  21 hours left
2497    3n1cn7ap3jl883263   9200     nissan  ...     florida      usa    2 days left
2498    3n1cn7ap4jl884311   9200     nissan  ...     florida      usa  21 hours left

>>> df.shape
(2499, 12)
```

Function name: `add_new_col_1`
Parameters: `df` (pd.DataFrame)
Return Type: NoneType
Description: Write a function that takes in the Pandas DataFrame corresponding to the output from `csv_parser` and returns the DataFrame with an added column titled `score`, which is equal to *`(-1/30000) * mileage + (year/2022)`* rounded to **three decimal places** for each car.

Python Test Case:

```
>>> df = csv_parser("cars.csv")
>>> df2 = add_new_col_1(df)
>>> df2
              vehicle_id  price      brand  ... country   bidding_time  score
0       jtezu11f88k007763   6300     toyota  ...     usa   10 days left -8.144
1       2fmdk3gc4bbb02217   2899       ford  ...     usa    6 days left -5.357
2       3c4pdcgg5jt346413   5350      dodge  ...     usa    2 days left -0.322
3       1ftfw1et4efc23745  25000       ford  ...     usa  22 hours left -1.142
4       3gcpcrec2jg473991  27700  chevrolet  ...     usa  22 hours left  0.776
...                   ...    ...        ...  ...     ...           ...    ...
2494    3n1cn7ap9kl880319   7800     nissan  ...     usa    1 days left  0.212
2495    3n1cn7ap5jl884088   9200     nissan  ...     usa  21 hours left -0.154
2496    3n1cn7ap9jl884191   9200     nissan  ...     usa  21 hours left -0.055
2497    3n1cn7ap3jl883263   9200     nissan  ...     usa    2 days left -0.087
2498    3n1cn7ap4jl884311   9200     nissan  ...     usa  21 hours left -0.048
```

Function name: `add_new_col_2`
Parameters: `df` (pd.DataFrame)
Return Type: pd.DataFrame
Description: You've developed a very specific taste for cars. You are only interested in a car if its brand is "`chevrolet`", "`dodge`", or "`ford`" and if its color is either "`blue`" or "`silver`". Additionally, you would like to have at least one day for bidding.

Write a function that takes in the Pandas DataFrame corresponding to the output from `csv_parser` and returns the Pandas DataFrame with an added column titled `relative_price`. If the requirements above are not satisfied for a specific car, this column should have the value "Not Interested". Otherwise, you should calculate the signed difference `price - avg.price` truncated **as type integer**. (Where `avg.price` represents the mean price of ALL cars in the list).

Python Test Case:

```
>>> df = csv_parser("cars.csv")
>>> df3 = add_new_col_2(df)
>>> df3

        vehicle_id         price  ...   bidding_time  relative_price
0       jtezu11f88k007763   6300  ...   10 days left  Not Interested
1       2fmdk3gc4bbb02217   2899  ...    6 days left          -15868
2       3c4pdcgg5jt346413   5350  ...    2 days left          -13417
3       1ftfw1et4efc23745  25000  ...  22 hours left  Not Interested
4       3gcpcrec2jg473991  27700  ...  22 hours left  Not Interested
...                   ...    ...  ...            ...             ...
2494    3n1cn7ap9kl880319   7800  ...    1 days left  Not Interested
2495    3n1cn7ap5jl884088   9200  ...  21 hours left  Not Interested
2496    3n1cn7ap9jl884191   9200  ...  21 hours left  Not Interested
2497    3n1cn7ap3jl883263   9200  ...    2 days left  Not Interested
2498    3n1cn7ap4jl884311   9200  ...  21 hours left  Not Interested

>>> len(df3[df3["relative_price"] != "Not Interested"])
189
```

Function name: `year_data`
Parameters: `df` (pd.DataFrame)
Return Type: pd.DataFrame
Description: You want to see how many different years are available for each make and model of car. For example, if there are only 2008 and 2009 Acura Doors available, the number of years available for that make and model is 2.

Hint: Use .agg() or .aggregate()
Note: You should return a **pd.DataFrame**, not **pd.Series.**

There is a one–line maximum requirement.

Python Test Case

```
>>> df = csv_parser("cars.csv")
>>> df4 = year_data(df)
>>> pd.set_option('display.max_rows', df4.shape[0]+1)
>>> df4
                          year
brand           model
acura           door         2
                mdx          1
audi            5            1
                door         2
                q5           1
bmw             coupe        1
                door         5
                m            1
                series       8
                x3           2
...
peterbilt       truck        4
ram             door         1
toyota          cruiser      1

>>> df4.shape
(180,1)
```

Function name: `cannot_a_ford`
Parameters: `df` (pd.DataFrame)
Return Type: pd.DataFrame
Description: Retrieve the average price of the ford cars per year of production. Convert the average to integers.

Hint: Use .agg() or .aggregate()
Note: You should return a **pd.DataFrame**, not **pd.Series.**
There is a one–line maximum requirement.

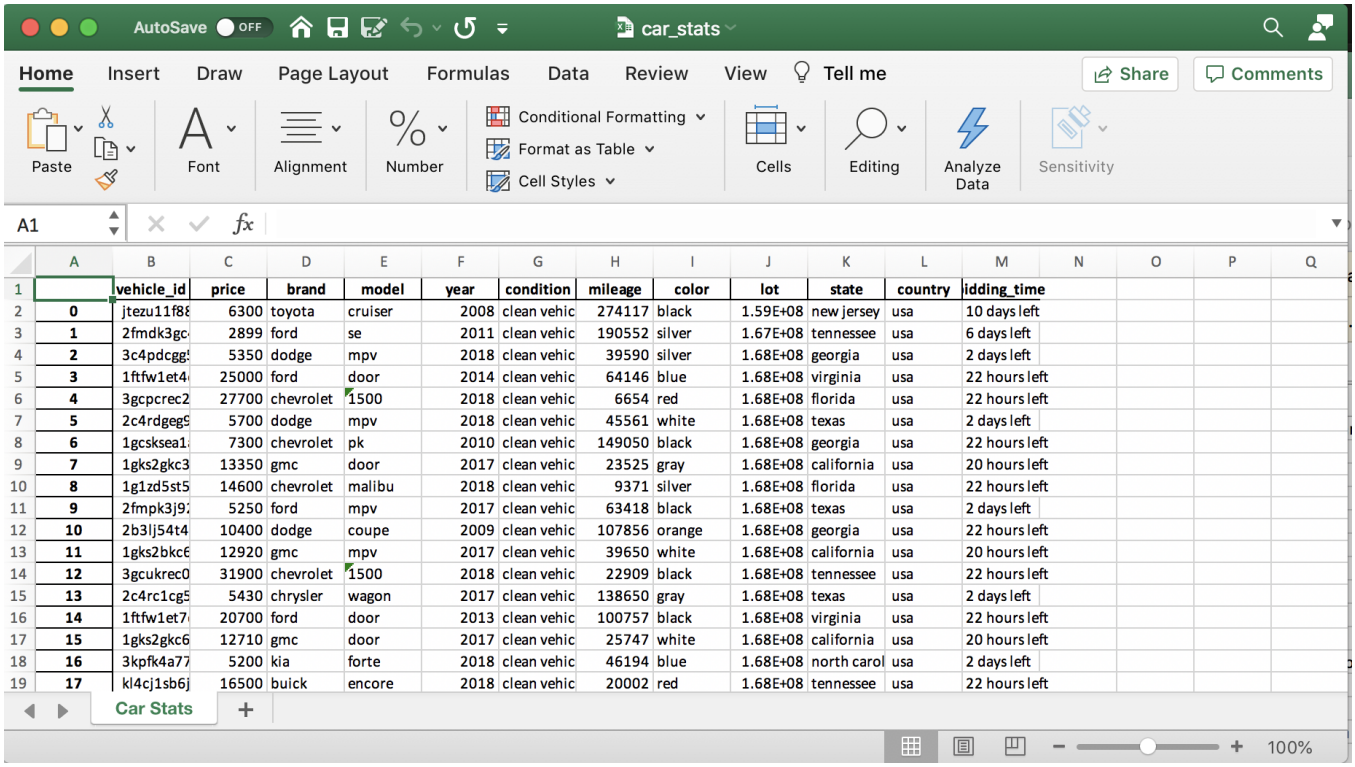
Python Test Case:

```
>>> df = csv_parser("cars.csv")
>>> ford_df = cannot_a_ford(df)
>>> print(ford_df)
      price
year
1984     25
1994     12
1996      0
1997      0
1998      8
...
2017  20286
2018  25315
2019  28045
2020  27088
```

Function name: `export_excel`
Parameters: `df` (pd.DataFrame), `filename` (str), `sheetname` (str)
Return Type: NoneType
Description: Write a function that takes in the given DataFrame called `df` and writes to a new excel file with the given `filename`. You should then name the corresponding sheet name to the name given by `sheetname`.

Python Test Case:

```
>>> df = csv_parser("cars.csv")
>>> export_excel(df, "car_stats.xlsx", "Car Stats")
```



AutoSave OFF car_stats

Home Insert Draw Page Layout Formulas Data Review View Tell me Share Comments

Paste Font Alignment Number Conditional Formatting Format as Table Cell Styles Cells Editing Analyze Data Sensitivity

A1

	A	B	C	D	E	F	G	H	I	J	K	L	M
1		vehicle_id	price	brand	model	year	condition	mileage	color	lot	state	country	idding_time
2	0	jtezu11f88	6300	toyota	cruiser	2008	clean vehic	274117	black	1.59E+08	new jersey	usa	10 days left
3	1	2fmdk3gc	2899	ford	se	2011	clean vehic	190552	silver	1.67E+08	tennessee	usa	6 days left
4	2	3c4pdcgg5	5350	dodge	mpv	2018	clean vehic	39590	silver	1.68E+08	georgia	usa	2 days left
5	3	1ftfw1et4	25000	ford	door	2014	clean vehic	64146	blue	1.68E+08	virginia	usa	22 hours left
6	4	3gcpcrec2	27700	chevrolet	1500	2018	clean vehic	6654	red	1.68E+08	florida	usa	22 hours left
7	5	2c4rdgeg9	5700	dodge	mpv	2018	clean vehic	45561	white	1.68E+08	texas	usa	2 days left
8	6	1gcsksea1	7300	chevrolet	pk	2010	clean vehic	149050	black	1.68E+08	georgia	usa	22 hours left
9	7	1gks2gkc3	13350	gmc	door	2017	clean vehic	23525	gray	1.68E+08	california	usa	20 hours left
10	8	1g1zd5st5	14600	chevrolet	malibu	2018	clean vehic	9371	silver	1.68E+08	florida	usa	22 hours left
11	9	2fmpk3j92	5250	ford	mpv	2017	clean vehic	63418	black	1.68E+08	texas	usa	2 days left
12	10	2b3lj54t4	10400	dodge	coupe	2009	clean vehic	107856	orange	1.68E+08	georgia	usa	22 hours left
13	11	1gks2bkc6	12920	gmc	mpv	2017	clean vehic	39650	white	1.68E+08	california	usa	20 hours left
14	12	3gcukrec0	31900	chevrolet	1500	2018	clean vehic	22909	black	1.68E+08	tennessee	usa	22 hours left
15	13	2c4rc1cg5	5430	chrysler	wagon	2017	clean vehic	138650	gray	1.68E+08	texas	usa	2 days left
16	14	1ftfw1et7	20700	ford	door	2013	clean vehic	100757	black	1.68E+08	virginia	usa	22 hours left
17	15	1gks2gkc6	12710	gmc	door	2017	clean vehic	25747	white	1.68E+08	california	usa	20 hours left
18	16	3kpfk4a77	5200	kia	forte	2018	clean vehic	46194	blue	1.68E+08	north carol	usa	2 days left
19	17	kl4cj1sb6j	16500	buick	encore	2018	clean vehic	20002	red	1.68E+08	tennessee	usa	22 hours left

Car Stats

100%

Gradescope Requirements

- NO PRINT STATEMENTS this will break the autograder
- NO FUNCTION CALLS outside of function definitions this will also break the autograder
- Make sure the file submitted is titled **HW03.py**
- Do not import any modules, packages, or libraries other than **numpy, pandas, pprint**
- Only submit the HW03.py file to Gradescope

Grading Rubric

spring_weather	10 pts
random_sale	10 pts
no_null	10 pts
wordle	10 pts
csv_parser	5 pts
add_new_col_1	10 pts
add_new_col_2	15 pts
year_data	10 pts
cannot_a_ford	15 pts
export_excel	5 pts
Total	100 pts