HW 1

Canvas assignment link for submission and due date.

Objective:

- familiarize with the use of basic functions in python packages
- familiarize with the use of python notebook: compile all cells to show results

Assessment:

- successful run of the Markdown and Python notebook cells.
- succesfull submission on Canvas link by the due date.

Keep the following in mind for **all** notebooks you develop:

1. Structure your notebook. Use headings with meaningful levels in Markdown cells, and explain the questions each piece of code is to answer or the reason it is there.
2. Make sure your notebook can always be rerun from top to bottom.
3. **DO NOT** erase notebook cells provided.

Setup

This section loads the relevant Python modules and does any configuration needed for the notebook to work.

Lets import python packages we will use in this homework:

- numpy - scientific computing package
- pandas - python data analysis package
- seaborn - statistical data visualization package

In [2]:
```
import numpy as  np
import pandas as pd
import seaborn as sns
```

Introduction to Pandas

In this chapter you will use pandas commands

Q1: Read the data file using Pandas. **Note** When we run your experiment to test for correctness, we assume that the data.csv is in the ../data/ folder relative to your HW1.ipynb.

1. **Download the Capital Bike Share data set** from https://archive.ics.uci.edu/ml/datasets/bike+sharing+dataset. Click 'Data Folder', download the zip file, and extract the `day.csv`

file.

- get used to downloading data files and saving them to correct hierarchy.
- big part of projct and source versioning practice

2. Read the Data File: use Pandas read_csv[] function to read the file into `bikes` dataframe.

- Our data file does not have column headers, so we need to specify the names.

A1 Replace the ? mark with your answer in the python cell below.

In [3]:
```
bikes= pd.read_csv('day.csv')
```

Q2: Use `head` to show the first few rows of the table:

- brief preview is a safety check you are exploring the correct data frame

A2 Replace the ? mark with your answer

In [4]:
```
bikes.head()
```

Out[4]:

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	at
0	1	2011-01-01	1	0	1	0	6	0	2	0.344167	0.363
1	2	2011-01-02	1	0	1	0	0	0	2	0.363478	0.353
2	3	2011-01-03	1	0	1	0	1	1	1	0.196364	0.189
3	4	2011-01-04	1	0	1	0	2	1	1	0.200000	0.212
4	5	2011-01-05	1	0	1	0	3	1	1	0.226957	0.229

Q3: Use `info` to show a description of the columns, along with the shape and memory use of the data frame:

- `.info()` or `.head()` can be called in the same cell as data load
- we separate them out in this notebook so that we can discuss them in the markdown cells, but we can combine them in the future.

A3 Replace the ? mark with your answer in the python cell below

In [5]:
```
bikes.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 731 entries, 0 to 730
Data columns (total 16 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   instant     731 non-null    int64  
 1   dteday      731 non-null    object 
 2   season      731 non-null    int64  
 3   yr          731 non-null    int64  
 4   mnth        731 non-null    int64  
 5   holiday     731 non-null    int64
 6   weekday     731 non-null    int64
 7   workingday  731 non-null    int64
 8   weathersit  731 non-null    int64
 9   temp        731 non-null    float64
 10  atemp       731 non-null    float64
 11  hum         731 non-null    float64
 12  windspeed   731 non-null    float64
 13  casual      731 non-null    int64
 14  registered  731 non-null    int64
 15  cnt         731 non-null    int64
dtypes: float64(4), int64(11), object(1)
```

Q4: Pandas provide a very useful function for exploring statistical properties of dataframe, and allow us to see data composition for numerical columns. Use pandas build-in function and show statistical information for columns.

A4: Replace the ? mark with your answer in the python cell below

In [6]:

```
bikes.describe()
```

Out[6]:

	instant	season	yr	mnth	holiday	weekday	workingday	weathers
count	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.0000
mean	366.000000	2.496580	0.500684	6.519836	0.028728	2.997264	0.683995	1.3953
std	211.165812	1.110807	0.500342	3.451913	0.167155	2.004787	0.465233	0.5448
min	1.000000	1.000000	0.000000	1.000000	0.000000	0.000000	0.000000	1.0000
25%	183.500000	2.000000	0.000000	4.000000	0.000000	1.000000	0.000000	1.0000
50%	366.000000	3.000000	1.000000	7.000000	0.000000	3.000000	1.000000	1.0000
75%	548.500000	3.000000	1.000000	10.000000	0.000000	5.000000	1.000000	2.0000
max	731.000000	4.000000	1.000000	12.000000	1.000000	6.000000	1.000000	3.0000

Q5: Use pandas functions for filtering the dataframe rows based on some column values. Find out number of rows where the value of the **temp** column is more than the average value of **temp** for the dataset.

Steps:

1. Find out the mean value of the temp column
2. Filter the rows where temp is grater than the mean value
3. Get the number of rows in the filtered dataframe.

A5: Replace the ? mark with your answer in python cell below

In [7]:
```
mean_value= bikes.temp.mean()
filtered_dataframe= bikes[bikes.temp > mean_value]
row_count= len(filtered_dataframe)

print(row_count)
```

```
367
```

Numpy

We now use Numpy for doing some mathematical calculation on the dataset.

Q6: Now, use numpy for working with below tasks

1. At first convert the dataframe into a numpy array
2. Print the shape of the numpy n-dimensional array
3. select and print rows from *100 to 105*

A6: Replace the ? mark with your answer

In [8]:
```
num_array=bikes.to_numpy() # conver the dataframe
print(num_array) #print the shape
print(num_array[[100,101,102,103,104,105], :]) # print rows from 100 to 105
```

```
[[1 '2011-01-01' 1 ... 331 654 985]
 [2 '2011-01-02' 1 ... 131 670 801]
 [3 '2011-01-03' 1 ... 120 1229 1349]
 ...
 [729 '2012-12-29' 1 ... 159 1182 1341]
 [730 '2012-12-30' 1 ... 364 1432 1796]
 [731 '2012-12-31' 1 ... 439 2290 2729]]
[[101 '2011-04-11' 2 0 4 0 1 1 2 0.595652 0.565217 0.716956 0.324474 855
  2493 3348]
 [102 '2011-04-12' 2 0 4 0 2 1 2 0.5025 0.493054 0.739167 0.274879 257
  1777 2034]
 [103 '2011-04-13' 2 0 4 0 3 1 2 0.4125 0.417283 0.819167 0.250617 209
  1953 2162]
 [104 '2011-04-14' 2 0 4 0 4 1 1 0.4675 0.462742 0.540417 0.1107 529 2738
  3267]
 [105 '2011-04-15' 2 0 4 1 5 0 1 0.446667 0.441913 0.67125 0.226375 642
  2484 3126]
 [106 '2011-04-16' 2 0 4 0 6 0 3 0.430833 0.425492 0.888333 0.340808 121
  674 795]]
```

Q7: Lets put it all together

1. Create a new numpy array selecting column number 10 - 13.
2. Sort the numpy array in ascending order based on the 2nd column of our new numpy array.
3. Print first 5 rows of the sorted numpy array

A7: Replace the ? mark with your answer in the cell below

In [9]:
```
new_array= num_array[:,10:13]

sorted_array= new_array[np.argsort(new_array[:,1])]

print(sorted_array[1:5, :])
```

```
[[0.391404 0.187917 0.507463]
 [0.426129 0.254167 0.274871]
 [0.492425 0.275833 0.232596]
 [0.315654 0.29 0.187192]]
```

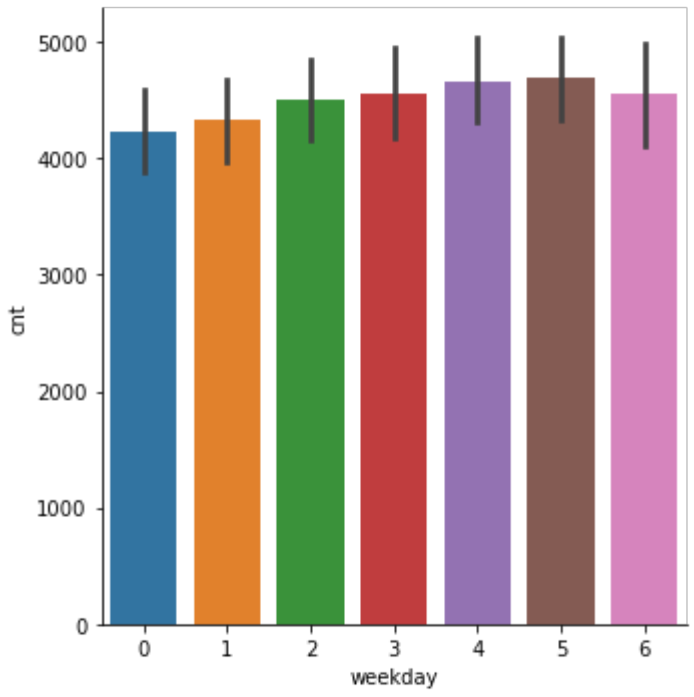
Seaborn Plotting 1

Seaborn package is used to plot the data. For every question in this chapter use the **bikes dataframe** for answering the questions.

Q8: Make a bar plot showing the mean number of riders (y-axis) per weekday (x-axis) using seaborn `catplot` method.

A8: Replace the ? mark with your answer

In [20]:
```
mean_riders = sns.catplot(x='weekday', y='cnt', kind = 'bar', data = bikes)
```



Analysis

The X-axis labels are not very helpful as day 0 is not clear. This is a question about how the data is *coded*. We'll talk more about data encoding next week. Unfortunately, the data documentation doesn't actually say how weekdays are coded! But we can infer from the data in this case: first data point is January 1, 2011, which was a Saturday, coded as weekday 6; it then resets to 0 for the next day, and starts counting up. Often, we will not be able to infer the data encoding from the data itself - we need to consult the codebook or data set description. We got lucky this time. But looking at the data can help us make sense of the codebook.

Lesson here is to always look at your data.
Q9: Turn these weekday numbers into a *categorical* variable so Pandas knows how to label them. Hint: use pandas.Categorical.from_codes().

A9: Replace the ? mark with your answer in the python cell below

In [40]:
```
codes = pd.CategoricalDtype(['m','t','w','th','f','s','sd'], ordered=True)

bikes['day_names'] = pd.Categorical.from_codes(codes = [0,1,2,4,5,6], codes=co
bikes.head()
```

```
  File "C:\Users\pedro\AppData\Local\Temp/ipykernel_3164/3885310286.py", line
3
    bikes['day_names'] = pd.Categorical.from_codes(codes = [0,1,2,4,5,6], code
s=codes )
                                                                          ^
SyntaxError: keyword argument repeated: codes
```

A10: Plot new data using seaborn `catplot`, where data=bikes, x-axis is `day_names` and y-axis is `cnt`

A10: Replace the ? mark with your answer in the python cell below

In [41]:
```
mean_riders = sns.catplot(x='day_names',y='cnt',kind='bar',data=bikes)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_3164/3108684414.py in <module>
----> 1 mean_riders = sns.catplot(x='day_names',y='cnt',kind='bar',data=bikes)

~\anaconda3\lib\site-packages\seaborn\_decorators.py in inner_f(*args, **kwarg
s)
     44             )
     45         kwargs.update({k: arg for k, arg in zip(sig.parameters, arg
s)})
---> 46         return f(**kwargs)
     47     return inner_f
     48

~\anaconda3\lib\site-packages\seaborn\categorical.py in catplot(x, y, hue, dat
a, row, col, col_wrap, estimator, ci, n_boot, units, seed, order, hue_order, r
ow_order, col_order, kind, height, aspect, orient, color, palette, legend, leg
end_out, sharex, sharey, margin_titles, facet_kws, **kwargs)
   3790     p = _CategoricalPlotter()
   3791     p.require_numeric = plotter_class.require_numeric
-> 3792     p.establish_variables(x_, y_, hue, data, orient, order, hue_order)
   3793     if (
   3794         order is not None

~\anaconda3\lib\site-packages\seaborn\categorical.py in establish_variables(se
lf, x, y, hue, data, orient, order, hue_order, units)
```

```
                        if isinstance(var, str):
                            err = "Could not interpret input '{}'".format(var)
--> 153                     raise ValueError(err)

                    # Figure out the plotting orientation
```

You have now now plotted the average rides per day.
Note: When we do not tell `catplot` what to do with multiple points for the same value (in this case the weekday name), it computes the mean and a bootstrapped 95% confidence interval.

Seaborn Polotting 2: View Data over Time

Lets explore how did rides-per-day change over the course of the data set?

- This kind of data - a sequence of data points associated with times - is called a *time series*.
- This data set gives us an `instant` column that records the data number since the start of the data set

Q11: Use seaborn.lineplot() where data=bikes, x-axis is `instant` and y-axis is `cnt` value.

A11: Replace the ? mark with your answer in the python cell below

In [26]:
```
sns.lineplot(?)
```

```
  File "C:\Users\tesic\AppData\Local\Temp/ipykernel_17456/913567343.py", line
1
    sns.lineplot(?)
                 ^
SyntaxError: invalid syntax
```

Lets view this graph for actual times on x-axis. The `dteday` column records the date. We can transform `dteday` column to the actual date comlumn using pandas.to_datetime() method on the column:

In [27]:
```
bikes['dt'] = pd.to_datetime(bikes['dteday'])
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_17456/281172278.py in <module>
----> 1 bikes['dt'] = pd.to_datetime(bikes['dteday'])

NameError: name 'bikes' is not defined
```

Q12: Now create a plot using seaborn.lineplot() where data=bikes, x-axis is `dt` and y-axis is `cnt` value.

A12: Replace the ? mark with your answer

In [28]:
```
sns.lineplot(?)
```

```
  File "C:\Users\tesic\AppData\Local\Temp/ipykernel_17456/913567343.py", line
```

```
1
    sns.lineplot(?)
                 ^
SyntaxError: invalid syntax
```

Next, plot the *weekly* rides by resampling. Right now, our `bikes` data is indexed by row number in the CSV file. We can change its index to another column, such as our `dt` column with the date, which then lets us do things like resample by week:

In [29]:
```
bikes.set_index('dt')['cnt'].resample('1W').sum().plot()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_17456/2181500339.py in <module>
----> 1 bikes.set_index('dt')['cnt'].resample('1W').sum().plot()

NameError: name 'bikes' is not defined
```

Q13: Save the modified dataframe in csv format using pandas *to_csv()* function. Give the file name as **day_output**

A13: Replace the ? mark with your answer

In [30]:
```
bikes.?
```

```
  File "C:\Users\tesic\AppData\Local\Temp/ipykernel_17456/3457342399.py", line
1
    bikes.?
          ^
SyntaxError: invalid syntax
```

What that code did, in one line, is:

1. Set the data frame's index to `dt` (`bikes.set_index('dt')`), returning a new DF
2. Select the count column (`['cnt']`), returning a series
3. Resample the series by week (`.resample('1W')`)
4. Combine measurements within each sample by summing them (`.sum()`)
5. Plotting the results using Pandas' defaults (`.plot()`)

Pandas default plotting functions are useful for quick plots to see what's in a data frame or series. They often are difficult to use to turn in to publication-ready charts.

Submission Instructions

1. Run all cells in `HW1.ipynb` and make sure there are no errors
2. Print `HW1.ipynb` to pdf file
3. Upload `HW1.ipynb` , `HW1.pdf` and `day_output.csv` file to Canvas assignment link before the deadline.