WTF23 DATA SCIENCE AND ARTIFICAIL INTELLIGENCE¶

GROUP C SUBGROUP 1

CLASSWORK ON PYTHON- NUMPY, PANDAS AND MATPLOTLIP

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
```

Q1: Write a NumPy program to get help on the add function.

In [2]:
```
help(np.add)
```

```
Help on ufunc:

add = <ufunc 'add'>
    add(x1, x2, /, out=None, *, where=True, casting='same_kind', order='K', dtype=Non
e, subok=True[, signature, extobj])

    Add arguments element-wise.

    Parameters
    ----------
    x1, x2 : array_like
        The arrays to be added.
        If ``x1.shape != x2.shape``, they must be broadcastable to a common
        shape (which becomes the shape of the output).
    out : ndarray, None, or tuple of ndarray and None, optional
        A location into which the result is stored. If provided, it must have
        a shape that the inputs broadcast to. If not provided or None,
        a freshly-allocated array is returned. A tuple (possible only as a
        keyword argument) must have length equal to the number of outputs.
    where : array_like, optional
        This condition is broadcast over the input. At locations where the
        condition is True, the `out` array will be set to the ufunc result.
        Elsewhere, the `out` array will retain its original value.
        Note that if an uninitialized `out` array is created via the default
        ``out=None``, locations within it where the condition is False will
        remain uninitialized.
    **kwargs
        For other keyword-only arguments, see the
        :ref:`ufunc docs <ufuncs.kwargs>`.

    Returns
    -------
    add : ndarray or scalar
        The sum of `x1` and `x2`, element-wise.
        This is a scalar if both `x1` and `x2` are scalars.

    Notes
    -----
    Equivalent to `x1` + `x2` in terms of array broadcasting.

    Examples
    --------
    >>> np.add(1.0, 4.0)
    5.0
    >>> x1 = np.arange(9.0).reshape((3, 3))
    >>> x2 = np.arange(3.0)
    >>> np.add(x1, x2)
    array([[  0.,   2.,   4.],
           [  3.,   5.,   7.],
           [  6.,   8.,  10.]])

    The ``+`` operator can be used as a shorthand for ``np.add`` on ndarrays.

    >>> x1 = np.arange(9.0).reshape((3, 3))
    >>> x2 = np.arange(3.0)
    >>> x1 + x2
    array([[ 0.,  2.,  4.],
           [ 3.,  5.,  7.],
           [ 6.,  8., 10.]])
```

In [3]:
```
np.info(np.add) #### specific to numpy
```

```
add(x1, x2, /, out=None, *, where=True, casting='same_kind', order='K', dtype=None, s
ubok=True[, signature, extobj])

Add arguments element-wise.

Parameters
----------
x1, x2 : array_like
    The arrays to be added.
    If ``x1.shape != x2.shape``, they must be broadcastable to a common
    shape (which becomes the shape of the output).
out : ndarray, None, or tuple of ndarray and None, optional
    A location into which the result is stored. If provided, it must have
    a shape that the inputs broadcast to. If not provided or None,
    a freshly-allocated array is returned. A tuple (possible only as a
    keyword argument) must have length equal to the number of outputs.
where : array_like, optional
    This condition is broadcast over the input. At locations where the
    condition is True, the `out` array will be set to the ufunc result.
    Elsewhere, the `out` array will retain its original value.
    Note that if an uninitialized `out` array is created via the default
    ``out=None``, locations within it where the condition is False will
    remain uninitialized.
**kwargs
    For other keyword-only arguments, see the
    :ref:`ufunc docs <ufuncs.kwargs>`.

Returns
-------
add : ndarray or scalar
    The sum of `x1` and `x2`, element-wise.
    This is a scalar if both `x1` and `x2` are scalars.

Notes
-----
Equivalent to `x1` + `x2` in terms of array broadcasting.

Examples
--------
>>> np.add(1.0, 4.0)
5.0
>>> x1 = np.arange(9.0).reshape((3, 3))
>>> x2 = np.arange(3.0)
>>> np.add(x1, x2)
array([[  0.,   2.,   4.],
       [  3.,   5.,   7.],
       [  6.,   8.,  10.]])

The ``+`` operator can be used as a shorthand for ``np.add`` on ndarrays.

>>> x1 = np.arange(9.0).reshape((3, 3))
>>> x2 = np.arange(3.0)
>>> x1 + x2
array([[ 0.,  2.,  4.],
       [ 3.,  5.,  7.],
       [ 6.,  8., 10.]])
```

Q2: Write a NumPy program to create an array with values ranging from 12 to 38.

In [4]:
```
np.linspace(12, 38) ##generates float
```

Out[4]:
```
array([12.        , 12.53061224, 13.06122449, 13.59183673, 14.12244898,
       14.65306122, 15.18367347, 15.71428571, 16.24489796, 16.7755102 ,
       17.30612245, 17.83673469, 18.36734694, 18.89795918, 19.42857143,
       19.95918367, 20.48979592, 21.02040816, 21.55102041, 22.08163265,
       22.6122449 , 23.14285714, 23.67346939, 24.20408163, 24.73469388,
       25.26530612, 25.79591837, 26.32653061, 26.85714286, 27.3877551 ,
       27.91836735, 28.44897959, 28.97959184, 29.51020408, 30.04081633,
       30.57142857, 31.10204082, 31.63265306, 32.16326531, 32.69387755,
       33.2244898 , 33.75510204, 34.28571429, 34.81632653, 35.34693878,
       35.87755102, 36.40816327, 36.93877551, 37.46938776, 38.        ])
```

In [5]:
```
np.linspace(12, 38, 5)
```

Out[5]:
```
array([12. , 18.5, 25. , 31.5, 38. ])
```

In [6]:
```
np.random.randint(12, 38, 10) ###to generate random integers
```

Out[6]:
```
array([16, 23, 18, 12, 31, 29, 19, 14, 22, 17])
```

In [7]:
```
np.arange(12, 39) ### generates integers
```

Out[7]:
```
array([12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28,
       29, 30, 31, 32, 33, 34, 35, 36, 37, 38])
```

Q3: Write a NumPy program to create a 3x3 matrix with values ranging from 2 to 10.

In [8]:
```
np.arange(2,11).reshape(3,3)
```

Out[8]:
```
array([[ 2,  3,  4],
       [ 5,  6,  7],
       [ 8,  9, 10]])
```

In [9]:
```
np.linspace(2,10,9)
```

Out[9]:
```
array([ 2.,  3.,  4.,  5.,  6.,  7.,  8.,  9., 10.])
```

In [10]:
```
np.linspace(2,10,9).reshape(3,3) ## this produces a float matrix
```

Out[10]:
```
array([[ 2.,  3.,  4.],
       [ 5.,  6.,  7.],
       [ 8.,  9., 10.]])
```

Q4: Write a NumPy program to create a 2d array with 1 on the border and 0 inside

In [11]:
```
a= np.array([[1,0,1], [1,0,1]])### manual imputation
a
```

Out[11]:
```
array([[1, 0, 1],
       [1, 0, 1]])
```

In [12]:
```
#### We can specify any dimension of array
a = np.ones([5,5])
print("original array by Oluchi")
print(a)

print("1 on the border and 0 inside in the array")
a[1:-1, 1:-1] = 0
print(a)
```

```
original array by Oluchi
[[1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]]
1 on the border and 0 inside in the array
[[1. 1. 1. 1. 1.]
 [1. 0. 0. 0. 1.]
 [1. 0. 0. 0. 1.]
 [1. 0. 0. 0. 1.]
 [1. 1. 1. 1. 1.]]
```

Q5 : Write a NumPy program to append values to the end of an array.

In [13]:
```
b =np.linspace(2,10,9).reshape(3,3)
#np.append((a, b), axis=0)
#### We can specify any dimension of array
a = np.ones([3,3])
print("original array by Oluchi")
print(a)

print("1 on the border and 0 inside in the array")
a[1:-1, 1:-1] = 0
print(a)
print(b)
```

```
original array by Oluchi
[[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]
1 on the border and 0 inside in the array
[[1. 1. 1.]
 [1. 0. 1.]
 [1. 1. 1.]]
[[ 2.  3.  4.]
 [ 5.  6.  7.]
 [ 8.  9. 10.]]
```

In [14]:
```
np.append(a, b, axis=0)
```

Out[14]:
```
array([[ 1.,  1.,  1.],
       [ 1.,  0.,  1.],
       [ 1.,  1.,  1.],
       [ 2.,  3.,  4.],
       [ 5.,  6.,  7.],
       [ 8.,  9., 10.]])
```

Q6: Write a NumPy program to generate five random numbers from the normal distribution

In [15]:
```
np.random.normal(size=5)
```

Out[15]:
```
array([-1.27587076, -0.07686806, -0.66495225, -0.6710439 , -1.37175471])
```

In [16]:
```
np.random.normal(5)### singular value
```

Out[16]:
```
5.6790799523945426
```

In [17]:
```
np.random.normal(5)
```

Out[17]:
```
4.207636177024182
```

Q7: Write a NumPy program to get the n largest values of an array

In [18]:
```
arr= np.arange(2,11)
print(arr)
np.max(arr)
```

Out[18]:
```
[ 2  3  4  5  6  7  8  9 10]
10
```

In [19]:
```
arr= np.arange(2,11)
print(arr)
sorted_index_array = np.argsort(arr)
sorted_array = arr[sorted_index_array]
print("Sorted array:", sorted_array)
n = 2
rslt = sorted_array[-n : ]
print("{} largest value:".format(n),
    rslt)
#np.max(arr)
```

```
[ 2  3  4  5  6  7  8  9 10]
Sorted array: [ 2  3  4  5  6  7  8  9 10]
2 largest value: [ 9 10]
```

In [20]:
```
b= np.arange(10)
print ("Original array by Oluchi:")
print(b)
#np.random.shuffle(b)
n=3
print(b[np.argsort(b)[-n:]])
```

```
Original array by Oluchi:
[0 1 2 3 4 5 6 7 8 9]
[7 8 9]
```

Q8: 10 matplotlib functions with examples

MATPLOTLIB

Matploitlib is a Python Library used for plotting, this python library provides and objected-oriented APIs for integrating plots into applications.

Matplotlib.pyplot is a plotting library used for 2D graphics in python programming language. It can be used in python scripts, shell, web application servers and other graphical user interface toolkits.

Below are 10 matplotlib functions with corresponding examples

Function	Uses	Syntax

.show()|displays the plot in the canvas| plt.show() .scatter()|plots a scatter plot|plt.scatter(x,y) .plot()| Produces line graph|plt.plot() .bar()|Produces bar charts|plt.bar() .legend()|Add legend to the plot|plt.legend() .cla()|Clear an axis | plt.cla() .clf()|Clear an entire figure |plt.clf() .close()|Close a graphical window| plt.close() .hist2d()|Used to make two dimensional histogram plot with default bin value of 10 |plt.hist2d(x,y, bins=n) .violinplot()|To make a violin plot|plt.violinplot() .savefig('file.png')|Used to save a figure|plt.savefig('file.png')

EXAMPLES WITH MATPLOTLIB FUNCTONS

In [21]:
```
df = pd.read_csv('data.csv')
```

In [22]:
```
df.head()
```

Out[22]:

	Make	Model	Year	Engine Fuel Type	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	Number of Doors	Marke
0	BMW	1 Series M	2011	premium unleaded (required)	335.0	6.0	MANUAL	rear wheel drive	2.0	Tuner,L P
1	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,P
2	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	L P
3	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,P
4	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	

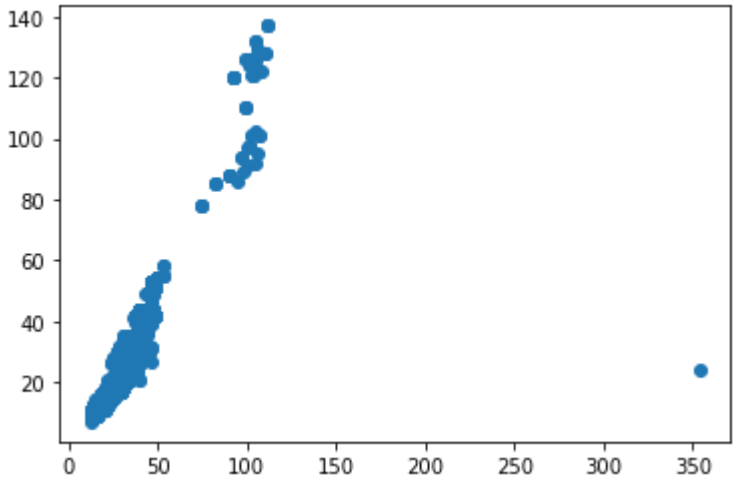
In [23]:
```
list(df)
```

Out[23]:
```
['Make',
 'Model',
 'Year',
 'Engine Fuel Type',
 'Engine HP',
 'Engine Cylinders',
 'Transmission Type',
 'Driven_Wheels',
 'Number of Doors',
 'Market Category',
 'Vehicle Size',
 'Vehicle Style',
 'highwayMPG',
 'citympg',
 'Popularity',
 'MSRP']
```

1: Scatterplot

In [24]:
```
plt.scatter('highwayMPG', 'citympg', data=df)
```

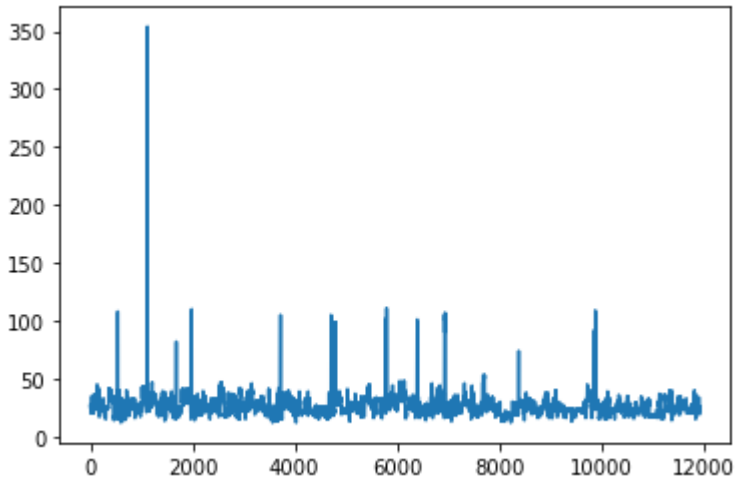
Out[24]:
```
<matplotlib.collections.PathCollection at 0x29cb837ae50>
```



2:PLOT

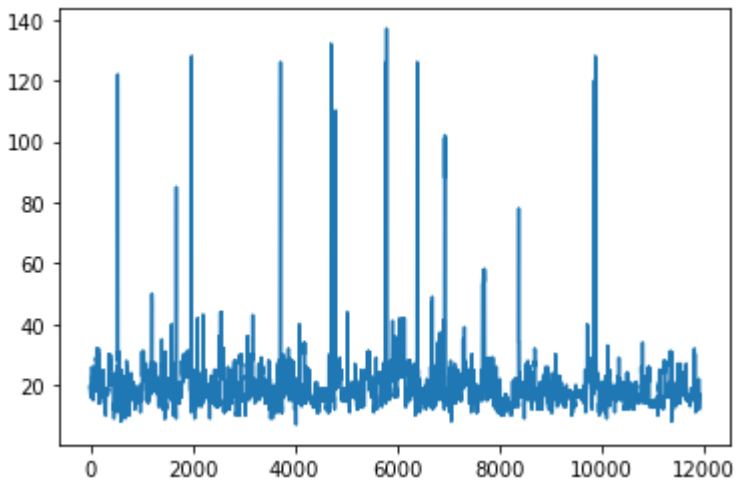
In [25]:
```
plt.plot('highwayMPG', data=df)
```

Out[25]:
```
[<matplotlib.lines.Line2D at 0x29cb8b2d430>]
```



In [26]:
```
plt.plot('citympg', data=df)
```

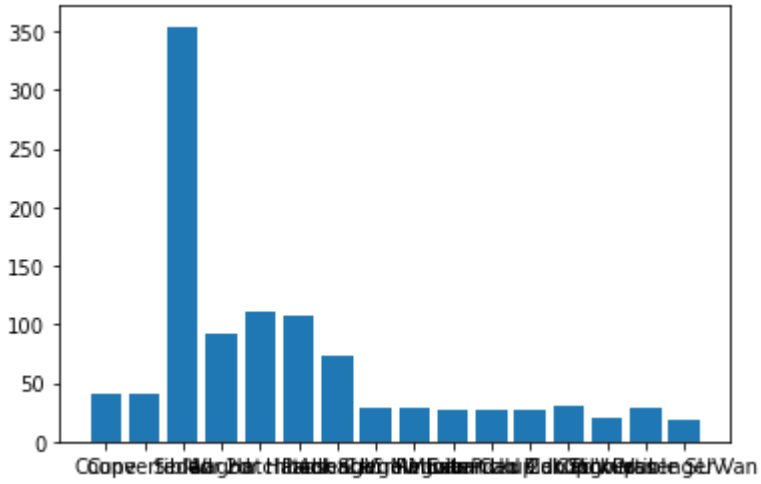
Out[26]:
```
[<matplotlib.lines.Line2D at 0x29cb8b9ae50>]
```



3:Bar Plot*

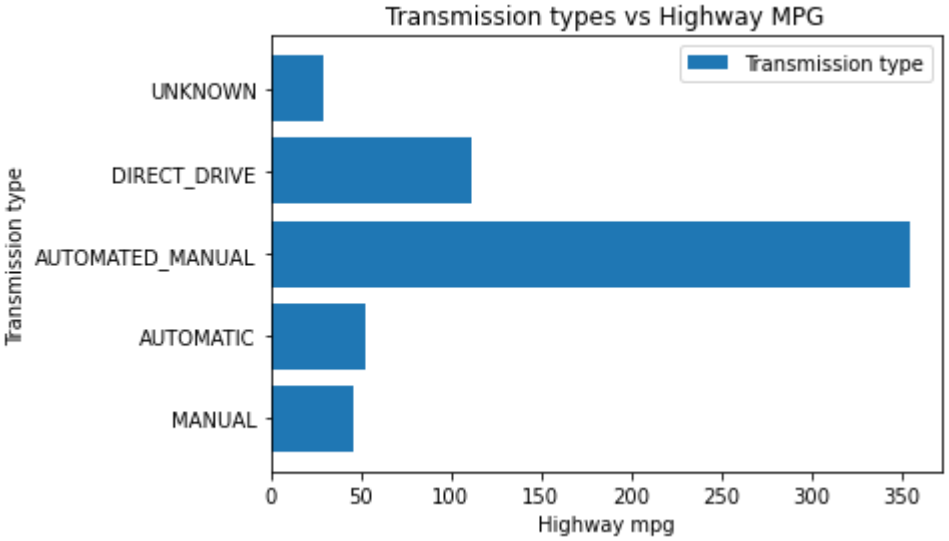
In [29]:
```
plt.bar('Vehicle Style','highwayMPG', data=df)
```

Out[29]:
```
<BarContainer object of 11914 artists>
```



4: Bar plot, Legend, ylabel, xlabel, show() and Title

In [28]:
```
a = plt.barh('Transmission Type','highwayMPG', label='Transmission type', data=df)
plt.legend()
plt.xlabel('Highway mpg')
plt.ylabel('Transmission type')
plt.title('Transmission types vs Highway MPG')
plt.show()
plt.savefig('a.jpg')
```




```
<Figure size 432x288 with 0 Axes>
```

9: 10 seaborn functions with examples

SEABORN

Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures.

Seaborn helps you explore and understand your data. Its plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots. Its dataset-oriented, declarative API lets you focus on what the different elements of your plots mean, rather than on the details of how to draw them.

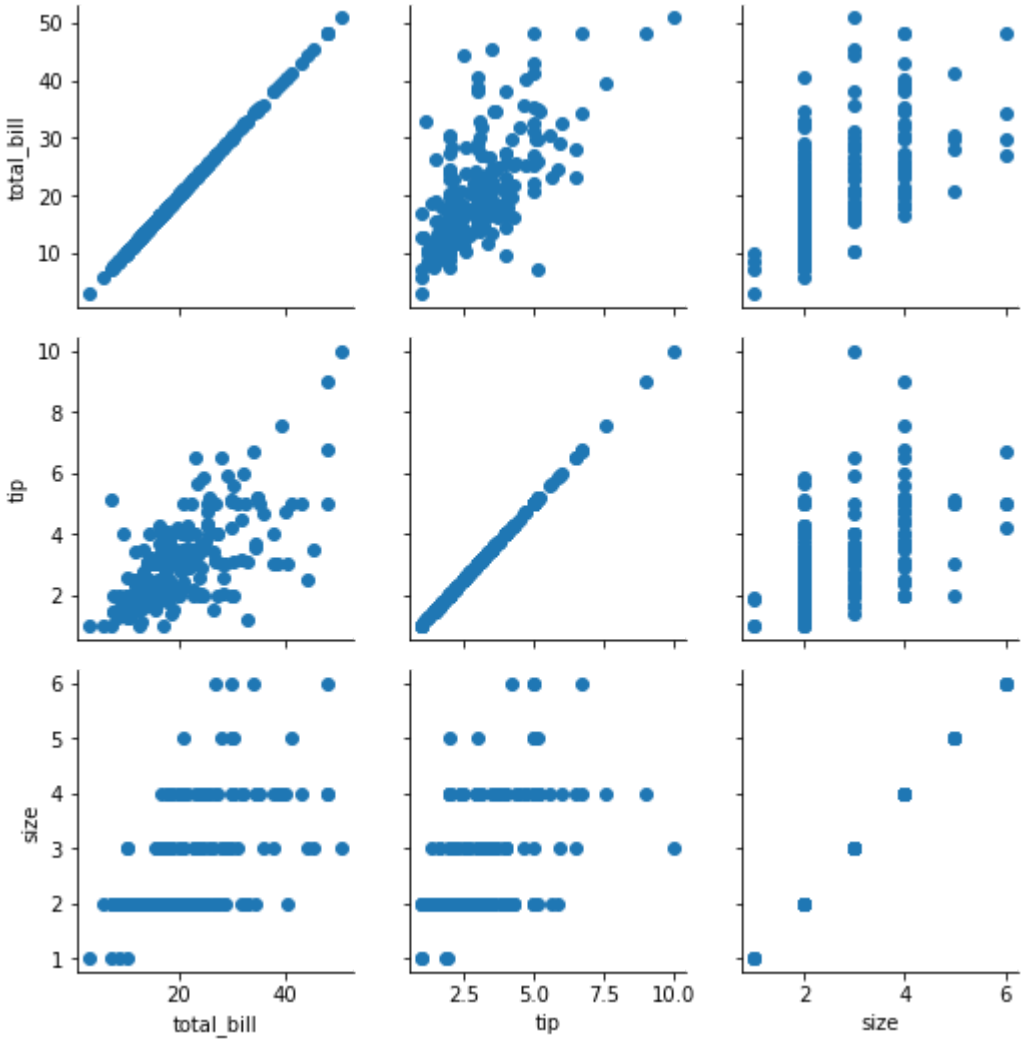
Function	Uses	Syntax

.countplot(x, data)|plots the count of a categorical variable|sb.countplot(x, data)| .set_style('style')|use to set the background style of the plot|sb.set_style('style')| .pairplot(data)|Plots pair wise relationship between variables in the dataset|sb.pairplot(data)| .scatterplot()| Produces scatter plots|sns.scatterplot()| .boxplot()|Produces boxplot|sns.boxplot()| .barplot()|Produces bar plots|sns.barplot()| .heatmap()| Used for correlation matrix plot|sns.heatmap()| .lineplot()| Used for line plot|sns.lineplot()| stripplot()|used when one of the variable under study is categorical. It represents the data in sorted order along any one of the axis.|sns.stripplot()| .PairGrid|use to draw a grid of subplots using the same plot type to visualize data| sns.pairgrid|

EXAMPLES WITH SEABORN FUNCTONS

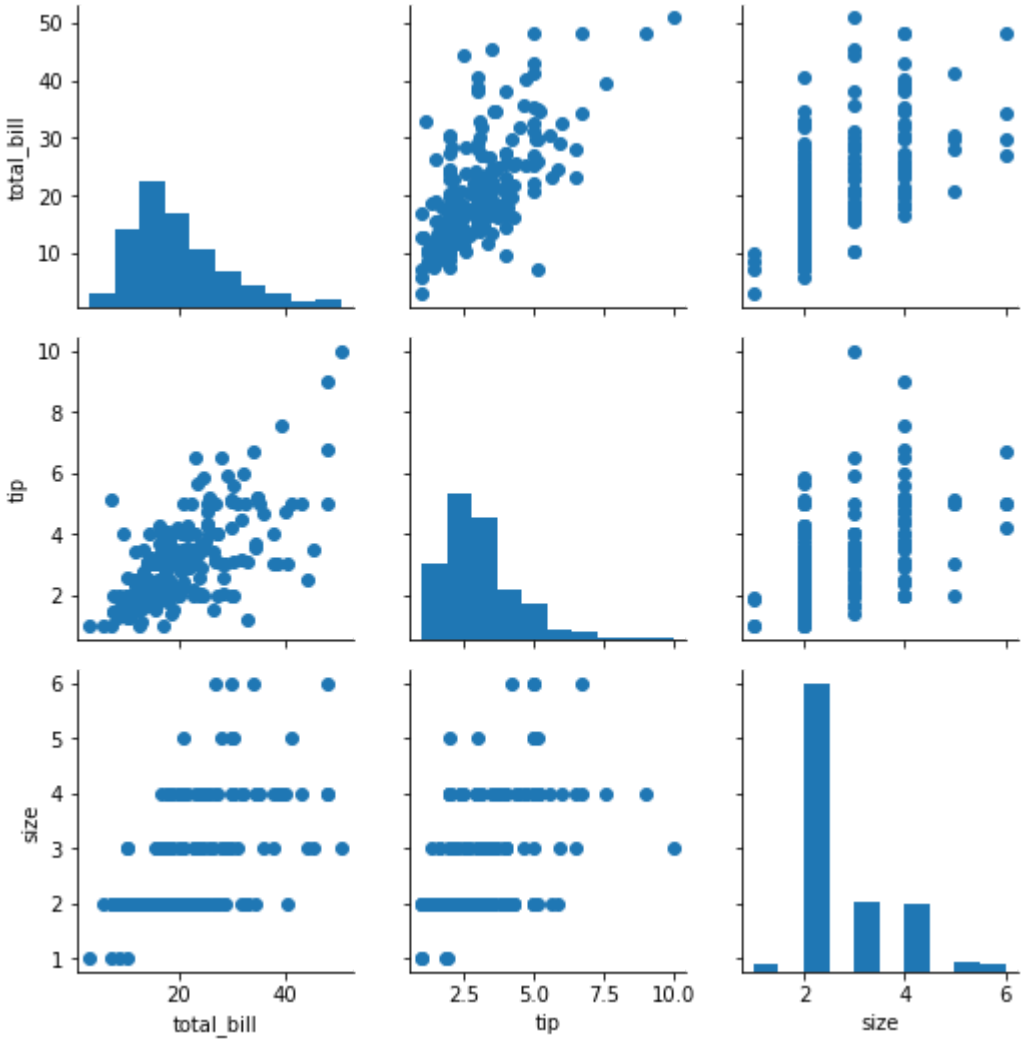
PAIRGRID AND SCATTER PLOT

In [30]:
```
dfs = sns.load_dataset('tips')
g = sns.PairGrid(dfs)
g.map(plt.scatter);
plt.show()
```



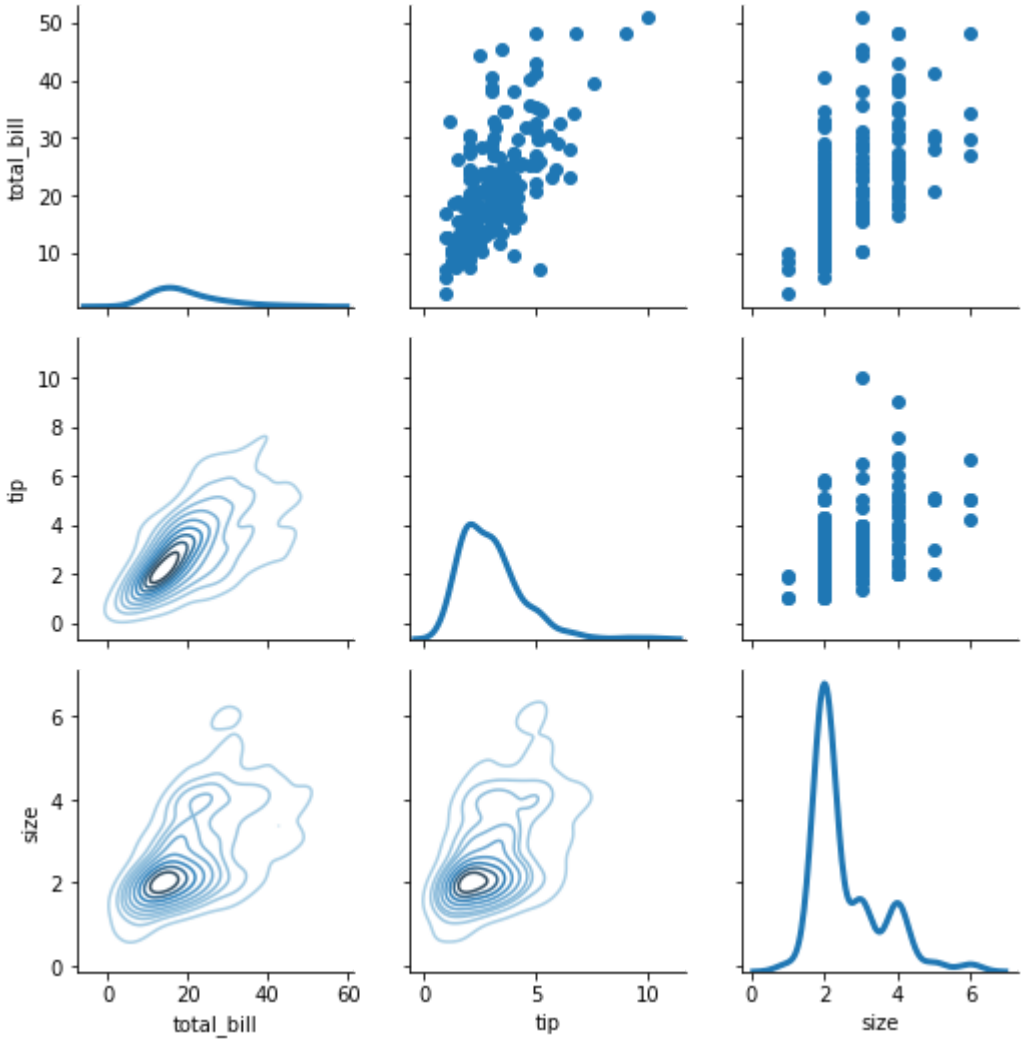
HISTOGRAM AND SCATTER PLOT

In [31]:
```
g = sns.PairGrid(dfs)
g.map_diag(plt.hist)
g.map_offdiag(plt.scatter);
plt.show()
```



HISTOGRAM, LINE AND SCATTER PLOT

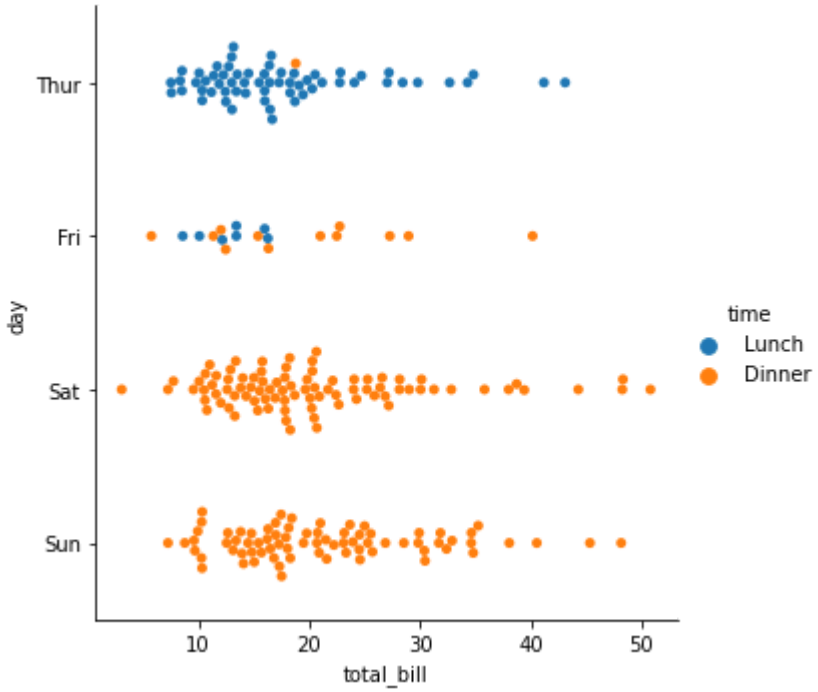
In [32]:
```
g = sns.PairGrid(dfs)
g.map_upper(plt.scatter)
g.map_lower(sns.kdeplot, cmap = "Blues_d")
g.map_diag(sns.kdeplot, lw = 3, legend = False);
plt.show()
```



CATEGORICAL PLOT WITH SEABORN

In [33]:
```
sns.catplot(data=dfs, x="total_bill", y="day", hue="time", kind="swarm")
```

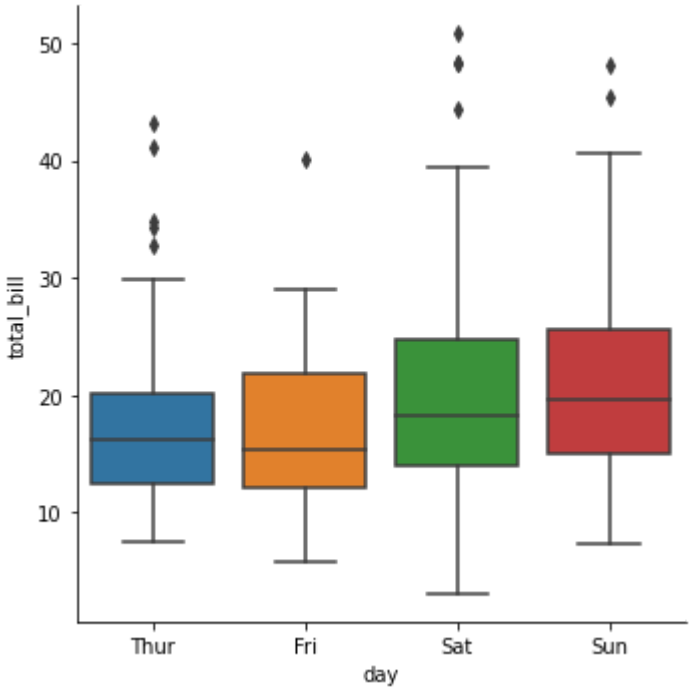
Out[33]:
```
<seaborn.axisgrid.FacetGrid at 0x29cd4e7dfa0>
```



BOXPLOT WITH SEABORN

In [34]:
```
sns.catplot(data=dfs, x="day", y="total_bill", kind="box")
```

Out[34]:
```
<seaborn.axisgrid.FacetGrid at 0x29cd4ff6be0>
```



CONCLUSION

There are several matplotlib and seaborn functions used for specific purposes. A comprehensive list can be found on the documentation page

Matplotlib

Seaborn

CONTRIBUTORS

Margaret Oluwadare

Loveth Osuagwu

Oluchi Okoro (Oluchi Oluchi)

Monsurat Onabajo

Mariam Anishere

Olubusayo Solola

Olayemi Ibiloye

Maryann Amaefula

Olayemi Oloyede

Oluwadunsin Olajide

Type Markdown and LaTeX: $\alpha 2$