pandas

import libs

```
import pandas as pd
import numpy as np
```

import custom dataset

```
weather = {
    'day':['1/1/2017', '1/2/2017', '1/3/2017', '1/4/2017', '1/5/2017', '1/6/2017'],
    'temperature': [32, 35, 28, 24, 32, 31],
    'windspeed': [6, 7, 2, 7, 4, 2],
    'event': ['Rain', 'Sunny', 'Snow', 'Snow', 'Rain', 'Sunny']
}
# creating a dict of weather data
```

pd.DataFrame(dict)

```
pd.DataFrame(weather)
# pd.DataFrame(dict)
# using dict to create a DataFrame
```

	day	temperature	windspeed	event
0	1/1/2017	32	6	Rain
1	1/2/2017	35	7	Sunny
2	1/3/2017	28	2	Snow
3	1/4/2017	24	7	Snow
4	1/5/2017	32	4	Rain
5	1/6/2017	31	2	Sunny

import dataset

```
# from google.colab import files
# uploaded = files.upload()
# nyc_weather.csv

import os
os.chdir(r'C:\Users\surya\Downloads\PG-DBDA-Mar23\Datasets')
os.getcwd()
```

```
'C:\\Users\\surya\\Downloads\\PG-DBDA-Mar23\\Datasets'
```

pd.read_csv('filename.csv')

```
df = pd.read_csv('nyc_weather.csv')
# pd.read_csv('filename.csv')
# creating DataFrame by reading csv file
df.head()
```

	EST	Temperature	DewPoint	Humidity	Sea Level PressureIn	VisibilityMiles	WindSpeedMPH	Pr
0	1/1/2016	38	23	52	30.03	10	8.0	
1	1/2/2016	36	18	46	30.02	10	7.0	
2	1/3/2016	40	21	47	29.86	10	8.0	
3	1/4/2016	25	9	44	30.05	10	9.0	
4	1/5/2016	20	-3	41	30.57	10	5.0	

```
# from google.colab import files
# uploaded = files.upload()
# weather_data1.csv

import os
os.chdir(r'C:\Users\surya\Downloads\PG-DBDA-Mar23\Datasets')
os.getcwd()
```

```
'C:\\Users\\surya\\Downloads\\PG-DBDA-Mar23\\Datasets'
```

```
df = pd.read_csv('weather_data1.csv')
df.head()
```

```
# creating DataFrame by reading csv file
```

	day	temperature	windspeed	event
0	1/1/2017	32	6	Rain
1	1/2/2017	35	7	Sunny
2	1/3/2017	28	2	Snow
3	1/4/2017	24	7	Snow
4	1/5/2017	32	4	Rain

```
df.shape
# checking the shape of DataFrame
```

```
(6, 4)
```

```
df.head()
# printing first 5 records of DataFrame
```

	day	temperature	windspeed	event
0	1/1/2017	32	6	Rain
1	1/2/2017	35	7	Sunny
2	1/3/2017	28	2	Snow
3	1/4/2017	24	7	Snow
4	1/5/2017	32	4	Rain

```
df.tail()
# printing last 5 records of DataFrame
```

	day	temperature	windspeed	event
1	1/2/2017	35	7	Sunny
2	1/3/2017	28	2	Snow
3	1/4/2017	24	7	Snow
4	1/5/2017	32	4	Rain
5	1/6/2017	31	2	Sunny

```
df.columns
# cheking the names of columns in DataFrame
```

```
Index(['day', 'temperature', 'windspeed', 'event'], dtype='object')
```

```
df.index
# checking the indices in Data Frame
```

```
RangeIndex(start=0, stop=6, step=1)
```

```
df[1:3]
# print rows 1 to 2 using row index
```

	day	temperature	windspeed	event
1	1/2/2017	35	7	Sunny
2	1/3/2017	28	2	Snow

selective columns

```
type(df['day'])
```

```
pandas.core.series.Series
```

```
df[['day', 'temperature']]
# print column day & temperature
```

	day	temperature
0	1/1/2017	32
1	1/2/2017	35
2	1/3/2017	28
3	1/4/2017	24
4	1/5/2017	32
5	1/6/2017	31

condition on temperature column

```
# find max temp
df['temperature'].max()
```

```
35
```

```
# find record where temp > 35
df[df['temperature'] > 35]
```

day	temperature	windspeed	event

```
# print days & temp column of records for which temp > 32
# df[['day', 'temperature']]
df[['day', 'temperature']][df['temperature'] > 32]
```

	day	temperature
1	1/2/2017	35

```
# print day column of record where temp is max
df['day'][df['temperature'] == df['temperature'].max()]
```

```
1    1/2/2017
Name: day, dtype: object
```

```
# population S.D.
df['temperature'].std(ddof=0)
```

```
3.496029493900505
```

```
# Sample S.D.
df['temperature'].std(ddof=1)
```

```
3.8297084310253524
```

import dataset

```
# from google.colab import files
# uploaded = files.upload()
# weather_datam.csv

```

```
import os
os.chdir(r'C:\Users\surya\Downloads\PG-DBDA-Mar23\Datasets')
os.getcwd()
```

```
'C:\\Users\\surya\\Downloads\\PG-DBDA-Mar23\\Datasets'
```

```
df = pd.read_csv('weather_datam.csv')
# reading csv into dataframe
df
```

	day	temperature	windspeed	event
0	1/1/2017	32	6	Rain
1	1/2/2017	-99999	7	Sunny
2	1/3/2017	28	-99999	Snow
3	1/4/2017	-99999	7	0
4	1/5/2017	32	-99999	Rain
5	1/6/2017	31	2	Sunny
6	1/6/2017	34	5	0

df.replace(-99999, value=np.NaN)

```
# replacing data value
df.replace(-99999, value=np.NaN)
```

	day	temperature	windspeed	event
0	1/1/2017	32.0	6.0	Rain
1	1/2/2017	NaN	7.0	Sunny
2	1/3/2017	28.0	NaN	Snow
3	1/4/2017	NaN	7.0	0
4	1/5/2017	32.0	NaN	Rain
5	1/6/2017	31.0	2.0	Sunny
6	1/6/2017	34.0	5.0	0

```
df2 = df.replace(to_replace=[-99999,10], value=np.NaN)
# replacing data value -99999, 10 with NaN
df2
```

	day	temperature	windspeed	event
0	1/1/2017	32.0	6.0	Rain
1	1/2/2017	NaN	7.0	Sunny
2	1/3/2017	28.0	NaN	Snow
3	1/4/2017	NaN	7.0	0
4	1/5/2017	32.0	NaN	Rain
5	1/6/2017	31.0	2.0	Sunny
6	1/6/2017	34.0	5.0	0

```
df.replace(0, value=111)
# replace in column
df['temperature']
```

```
0        32
1    -99999
2        28
3    -99999
4        32
5        31
6        34
Name: temperature, dtype: int64
```

HW

- replace 0 with -8
- replace -numeric values with NaN

matplotlib library

import libs

```
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
```

create X & Y

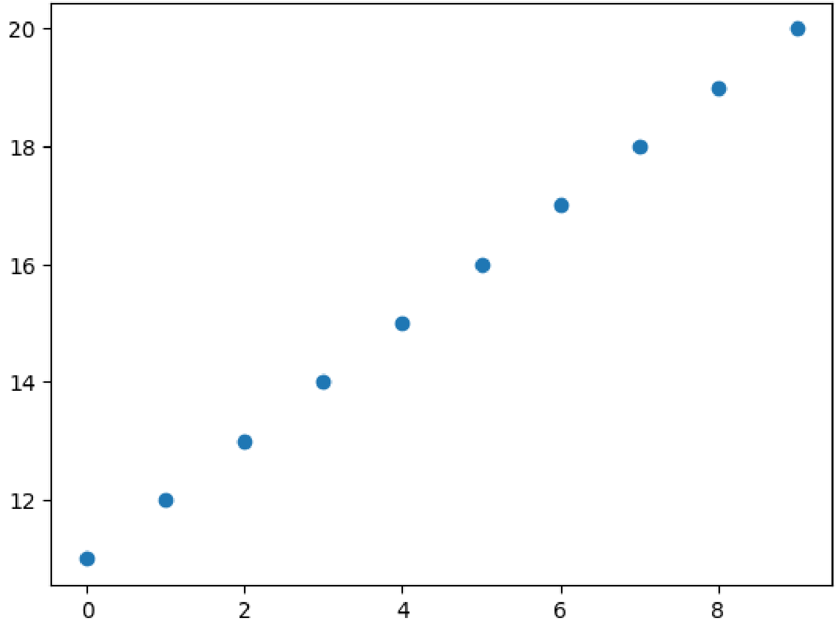
```
x = np.arange(0, 10)
y = np.arange(11, 21)
# creating ndarrays
```

scatter plots

plt.scatter(x, y)

```
plt.scatter(x, y)
# plot a scatter graph
```

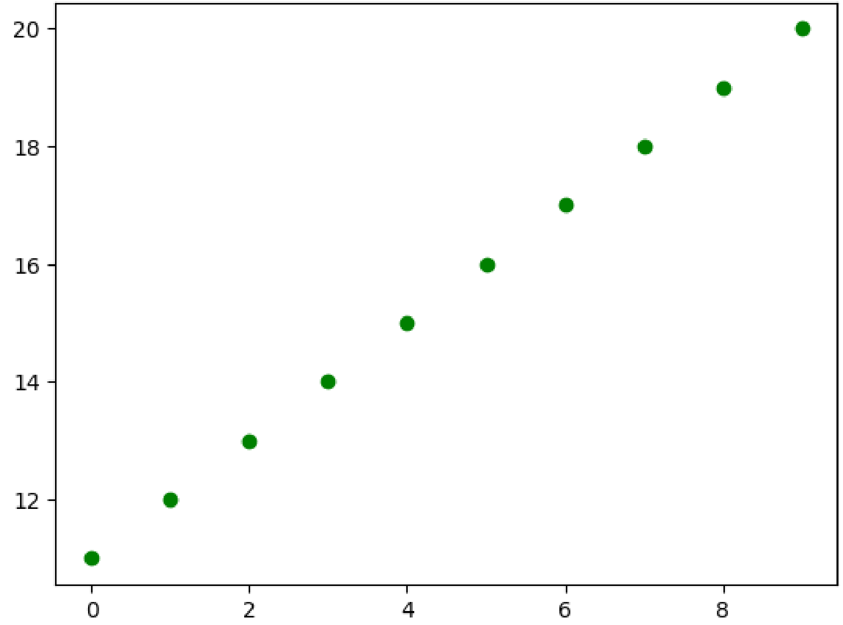
```
<matplotlib.collections.PathCollection at 0x1b20c49bc10>
```



plt.scatter(x, y, c='g')

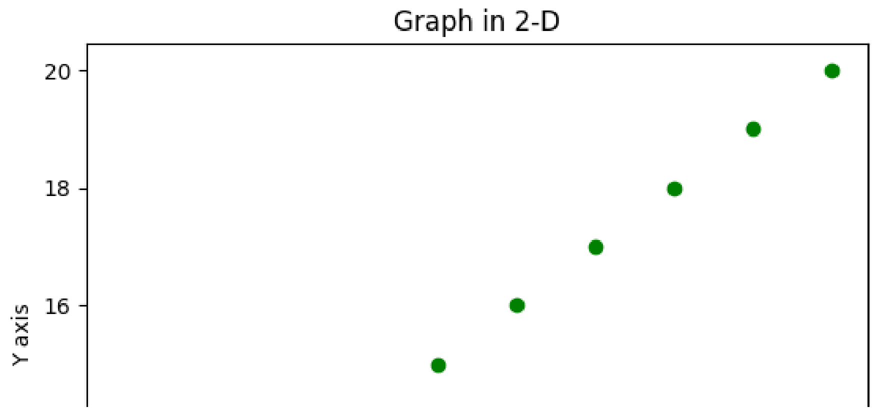
```
plt.scatter(x, y, c='g')
# plot a scatter graph with color
```

```
<matplotlib.collections.PathCollection at 0x1b20c663760>
```



```
plt.scatter(x, y, c='g')
plt.xlabel('X axis')
plt.ylabel('Y axis')
plt.title('Graph in 2-D')
# plot a scatter graph with color xlabel, ylabel & title
```

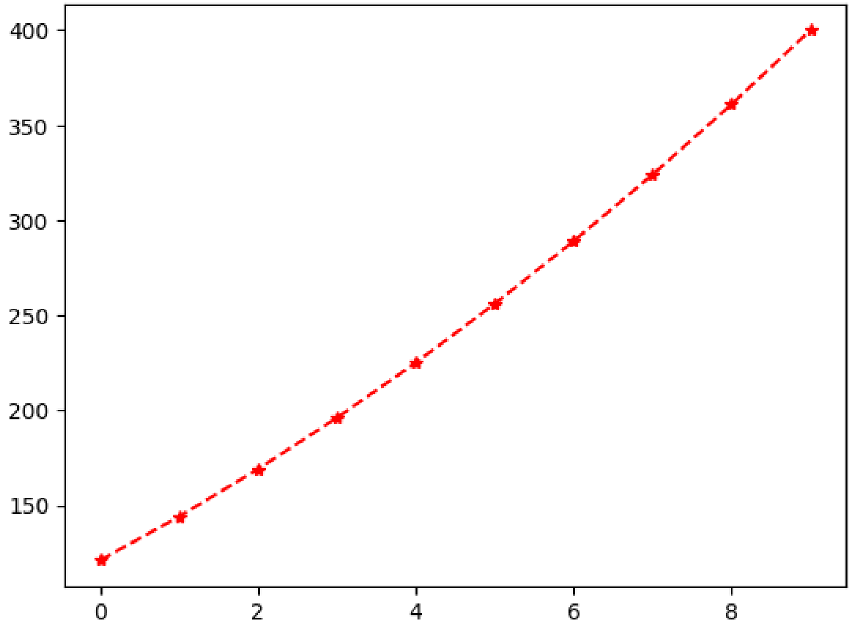
```
Text(0.5, 1.0, 'Graph in 2-D')
```



plt.plot(x, y, format, linestyle)

```
# x=np.arange(1000)
y=y**2
plt.plot(x, y, 'r*', linestyle='dashed')
```

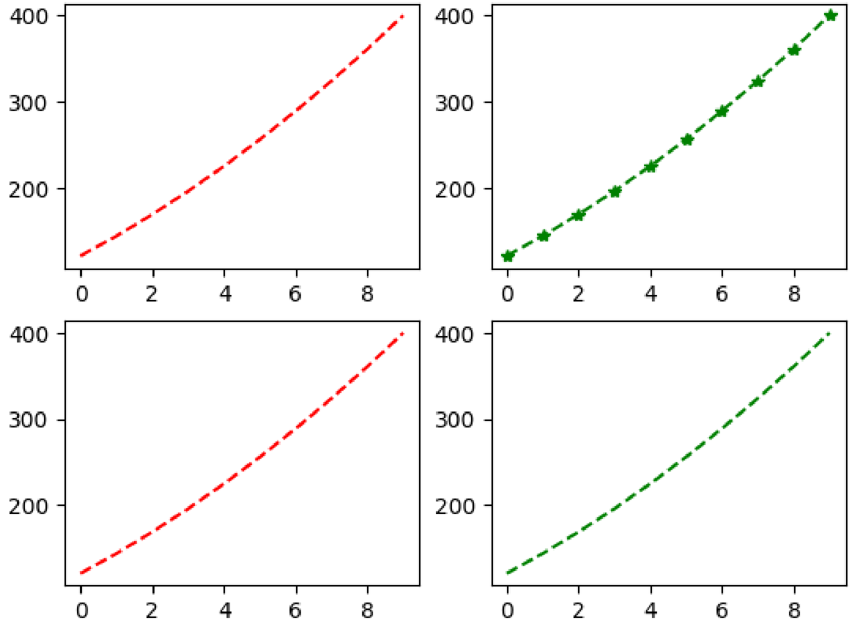
```
[<matplotlib.lines.Line2D at 0x1b20ae952b0>]
```



plt.subplot(int, int, index)

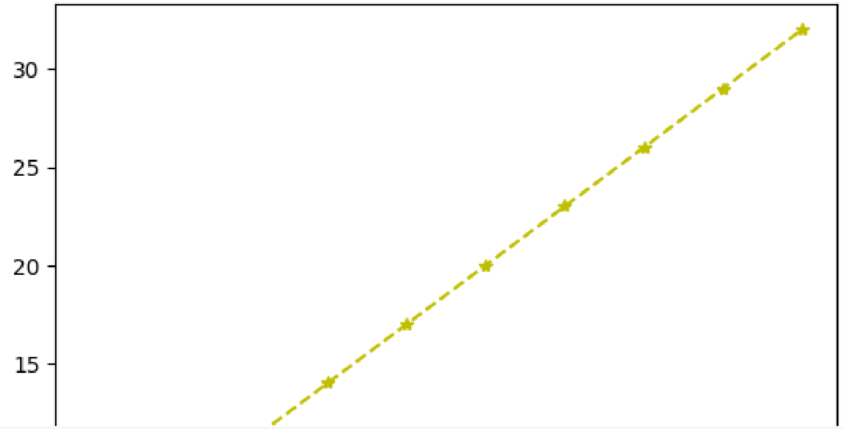
```
plt.subplot(2, 2, 1)
plt.plot(x, y, 'r--')
plt.subplot(2, 2, 2)
plt.plot(x, y, 'g*--')
plt.subplot(2, 2, 3)
plt.plot(x, y, 'r--')
plt.subplot(2, 2, 4)
plt.plot(x, y, 'g--')
```

```
[<matplotlib.lines.Line2D at 0x1b20ac384f0>]
```



```
y=3*x + 5
plt.plot(x, y, 'y*--')
```

```
[<matplotlib.lines.Line2D at 0x1b20c2f2d00>]
```



```
np.pi
```

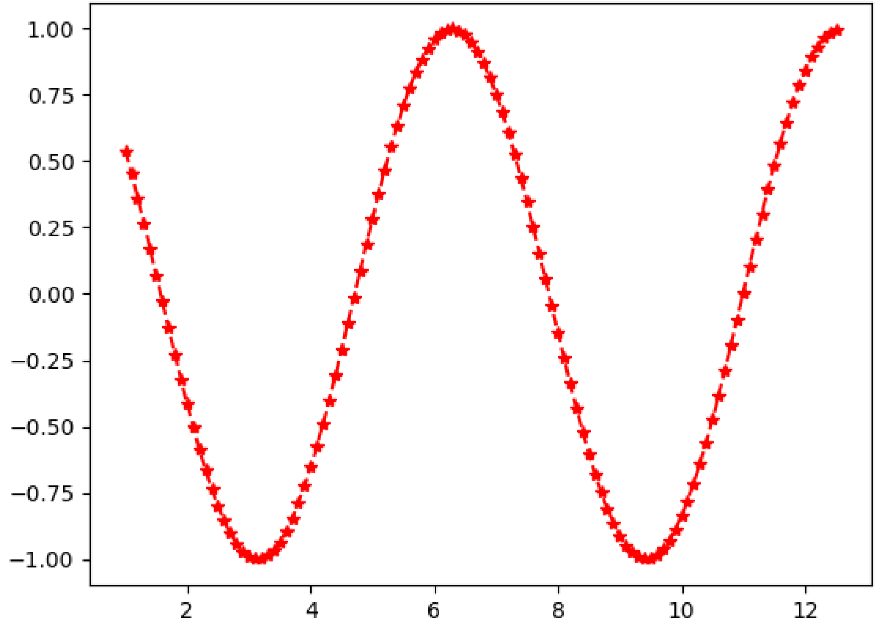
```
3.141592653589793
```

```
x = np.arange(1, 4*np.pi, 0.1)
y = np.sin(x)
plt.plot(x, y, 'r*--')
# plots sin-wave
```

```
[<matplotlib.lines.Line2D at 0x1b20c33ebb0>]
```

```
x = np.arange(1, 4*np.pi, 0.1)
y = np.cos(x)
plt.plot(x, y, 'r*--')
# plots cos-wave
```

```
[<matplotlib.lines.Line2D at 0x1b20c406df0>]
```



Distribution plots

```
# from google.colab import files
# uploaded = files.upload()
# # tips.csv

import os
os.chdir(r'C:\Users\surya\Downloads\PG-DBDA-Mar23\Datasets')
os.getcwd()
```

```
'C:\\Users\\surya\\Downloads\\PG-DBDA-Mar23\\Datasets'
```

```
df = pd.read_csv('tips.csv')
df
```

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
...	...	...	...	...	...	...	...
239	29.03	5.92	Male	No	Sat	Dinner	3
240	27.18	2.00	Female	Yes	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

244 rows × 7 columns

```
df.describe()
```

	total_bill	tip	size
count	244.000000	244.000000	244.000000
mean	19.785943	2.998279	2.569672
std	8.902412	1.383638	0.951100
min	3.070000	1.000000	1.000000
25%	13.347500	2.000000	2.000000
50%	17.795000	2.900000	2.000000
75%	24.127500	3.562500	3.000000
max	50.810000	10.000000	6.000000

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   total_bill  244 non-null    float64
 1   tip         244 non-null    float64
 2   sex         244 non-null    object 
 3   smoker      244 non-null    object 
 4   day         244 non-null    object 
 5   time        244 non-null    object 
 6   size        244 non-null    int64  
dtypes: float64(2), int64(1), object(4)
memory usage: 13.5+ KB
```

seaborn library

import libs

```
import seaborn as sns
```

import dataset from seaborn

```
df = sns.load_dataset('tips')
df.head()
```

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

```
df.shape
```

```
(244, 7)
```

```
df.describe()
```

	total_bill	tip	size
count	244.000000	244.000000	244.000000
mean	19.785943	2.998279	2.569672
std	8.902412	1.383638	0.951100
min	3.070000	1.000000	1.000000
25%	13.347500	2.000000	2.000000
50%	17.795000	2.900000	2.000000
75%	24.127500	3.562500	3.000000
max	50.810000	10.000000	6.000000

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype   
---  ------      --------------  -----   
 0   total_bill  244 non-null    float64 
 1   tip         244 non-null    float64 
 2   sex         244 non-null    category
 3   smoker      244 non-null    category
 4   day         244 non-null    category
 5   time        244 non-null    category
 6   size        244 non-null    int64   
dtypes: category(4), float64(2), int64(1)
memory usage: 7.4 KB
```

relations / correlations

df.corr()

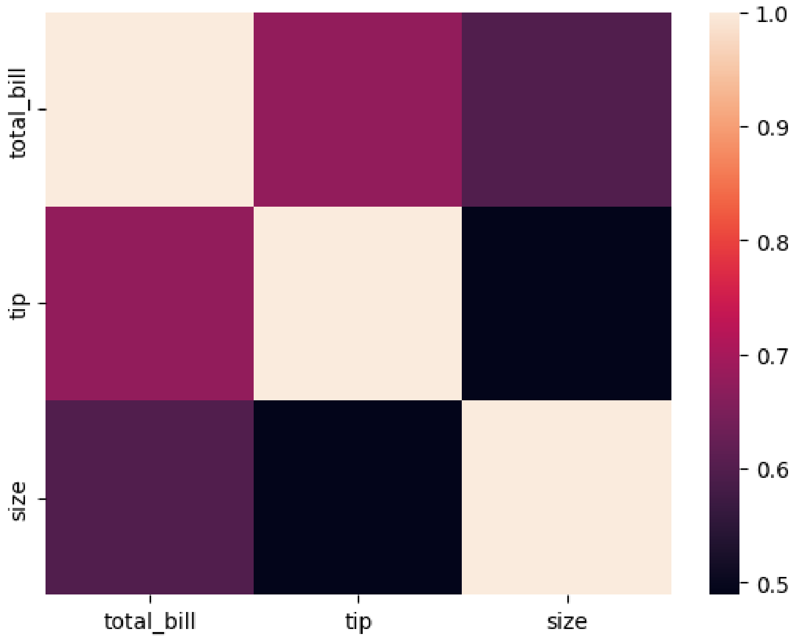
```
df.iloc[ : , [0, 1, 6]].corr()
# correlation matrix
```

	total_bill	tip	size
total_bill	1.000000	0.675734	0.598315
tip	0.675734	1.000000	0.489299

heatmap(correlation)

```
sns.heatmap(df.iloc[ : , [0, 1, 6]].corr())
# heatmap
```

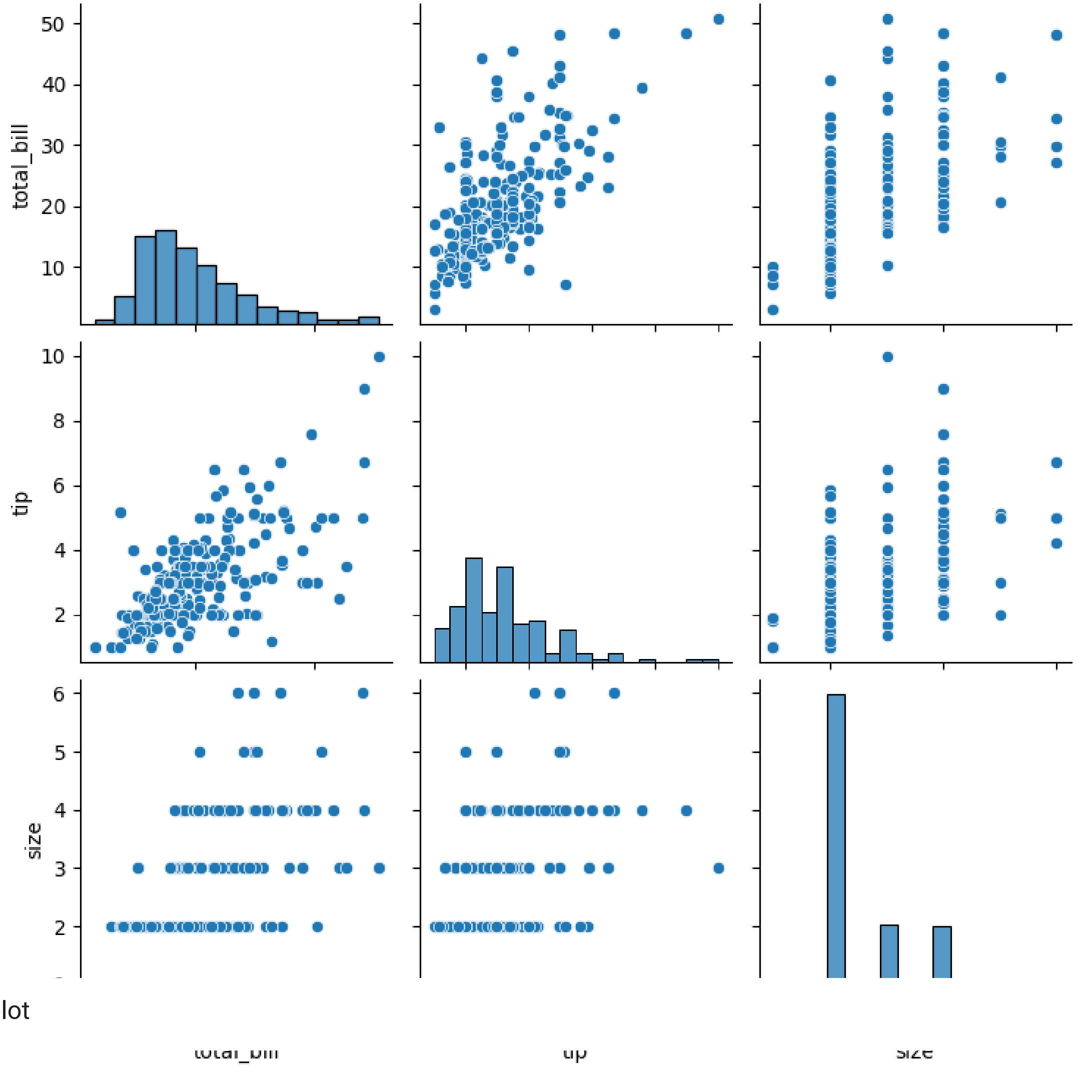
```
<Axes: >
```



pairplot

```
sns.pairplot(df)
```

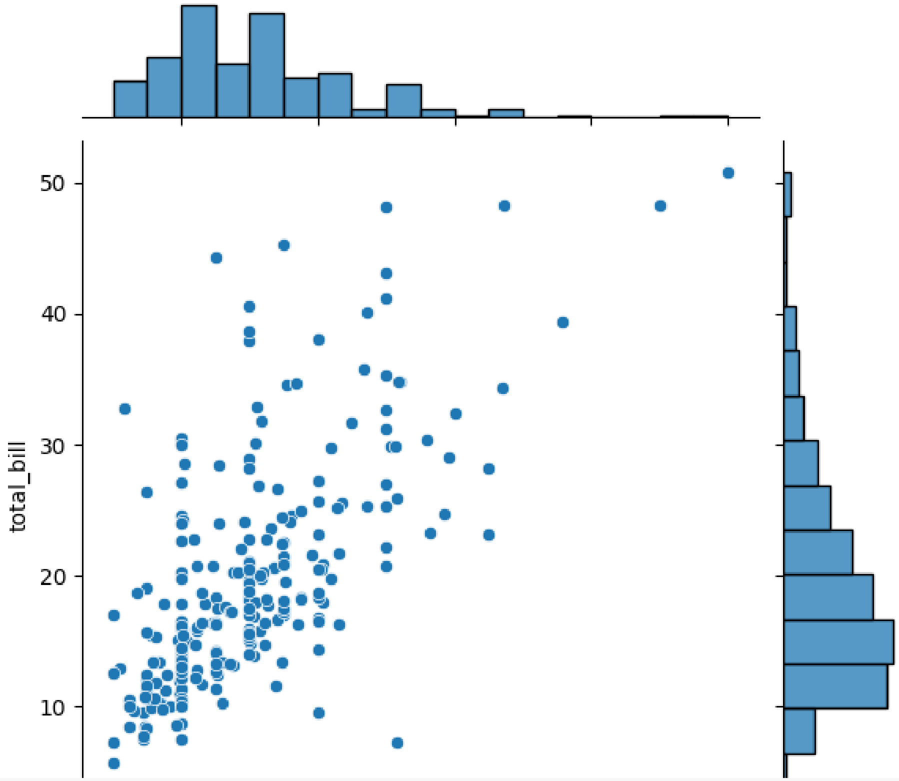
```
c:\users\surya\appdata\local\programs\python\python39\lib\site-packages\seaborn\axisgrid.p
  self._figure.tight_layout(*args, **kwargs)
<seaborn.axisgrid.PairGrid at 0x1b212063460>
```



jointplot

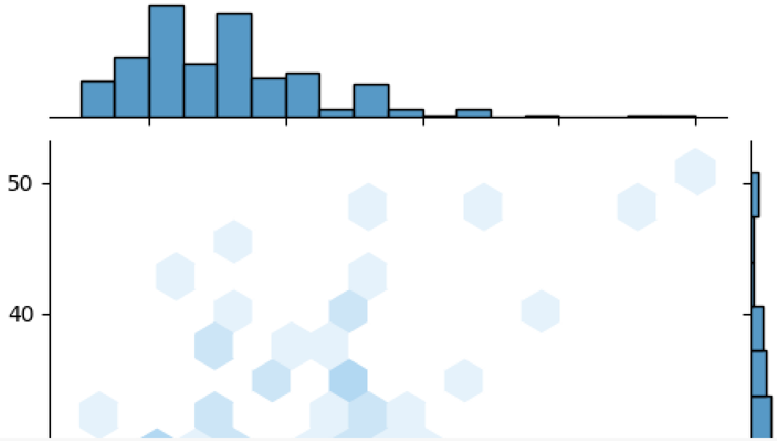
```
sns.jointplot(x='tip', y='total_bill', data=df)
# sns.jointplot(x, y, data=df)
```

```
<seaborn.axisgrid.JointGrid at 0x1b212949cd0>
```



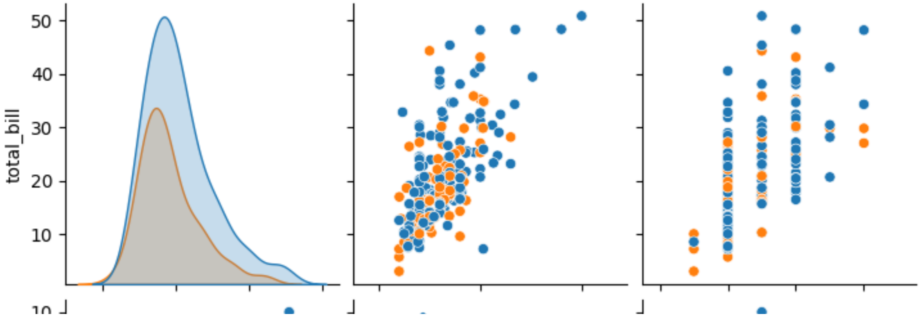
```
sns.jointplot(x='tip', y='total_bill', data=df, kind='hex')
# sns.jointplot(x, y, data=df, kind='hex')
```

```
<seaborn.axisgrid.JointGrid at 0x1b212df8e50>
```



```
sns.pairplot(df, hue='sex')
```

```
c:\users\surya\appdata\local\programs\python\python39\lib\site-packages\seaborn\axisgrid.p
  self._figure.tight_layout(*args, **kwargs)
<seaborn.axisgrid.PairGrid at 0x1b2130669d0>
```



distribution plots

distplot

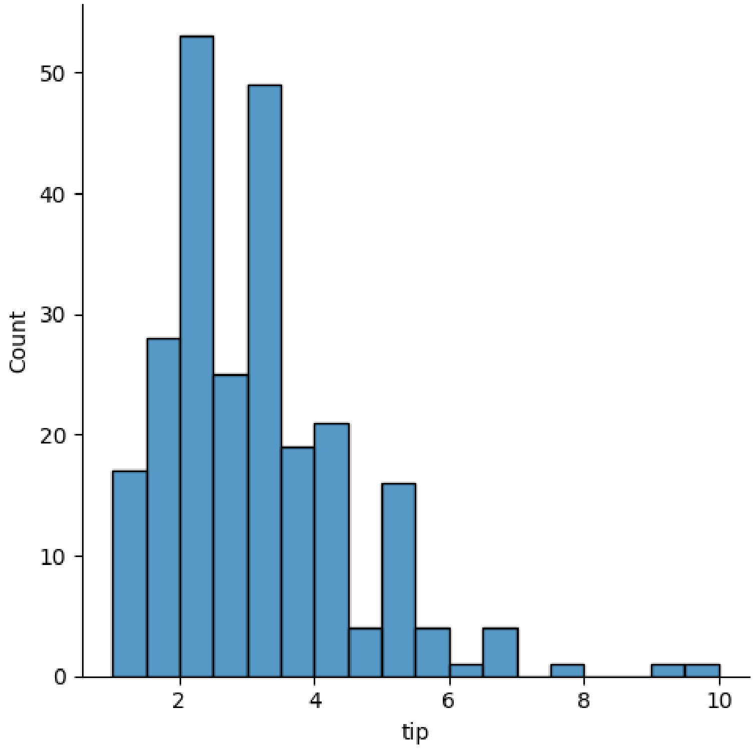
```
# distplot
sns.distplot(df['tip'])
```

```
C:\Users\surya\AppData\Local\Temp\ipykernel_31444\3837117233.py:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['tip'])
<Axes: xlabel='tip', ylabel='Density'>
```

displot

```
# displot
sns.displot(df['tip'])
```

```
c:\users\surya\appdata\local\programs\python\python39\lib\site-packages\seaborn\axisgrid.p
  self._figure.tight_layout(*args, **kwargs)
<seaborn.axisgrid.FacetGrid at 0x1b212e03190>
```

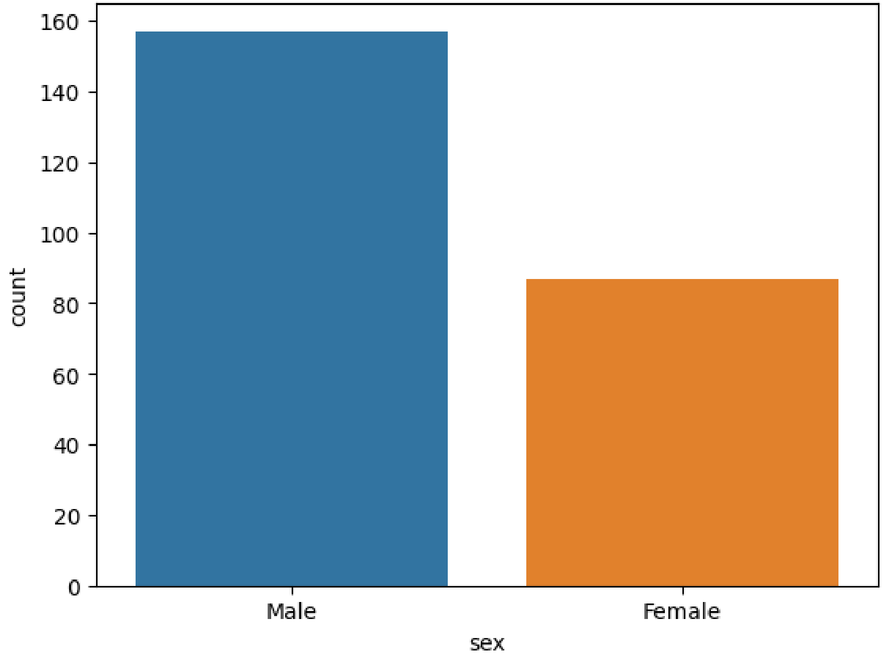


Categorical plots

countplot()

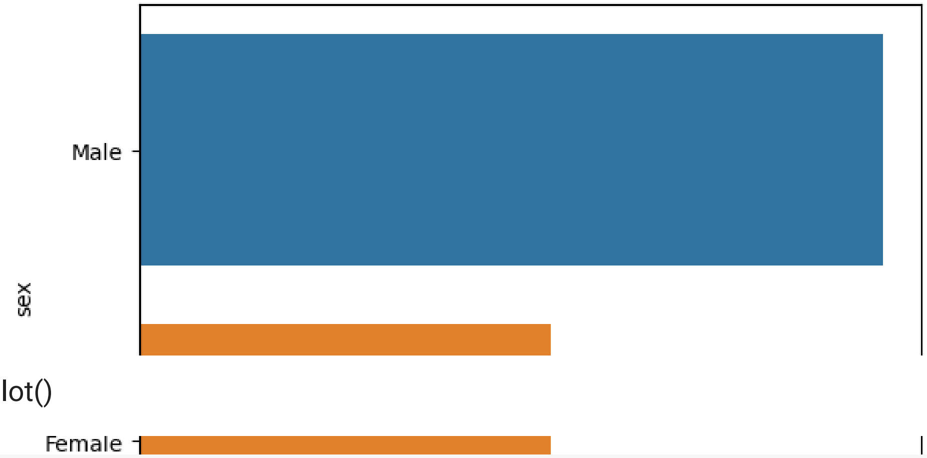
```
sns.countplot(x='sex', data=df)
```

```
<Axes: xlabel='sex', ylabel='count'>
```



```
sns.countplot(y='sex', data=df)
```

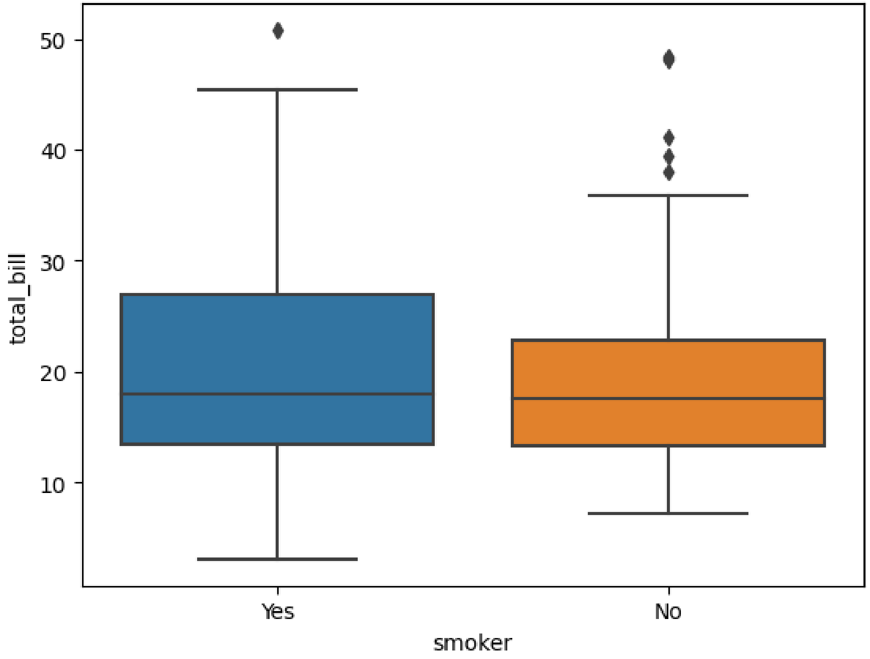
```
<Axes: xlabel='count', ylabel='sex'>
```



boxplot()

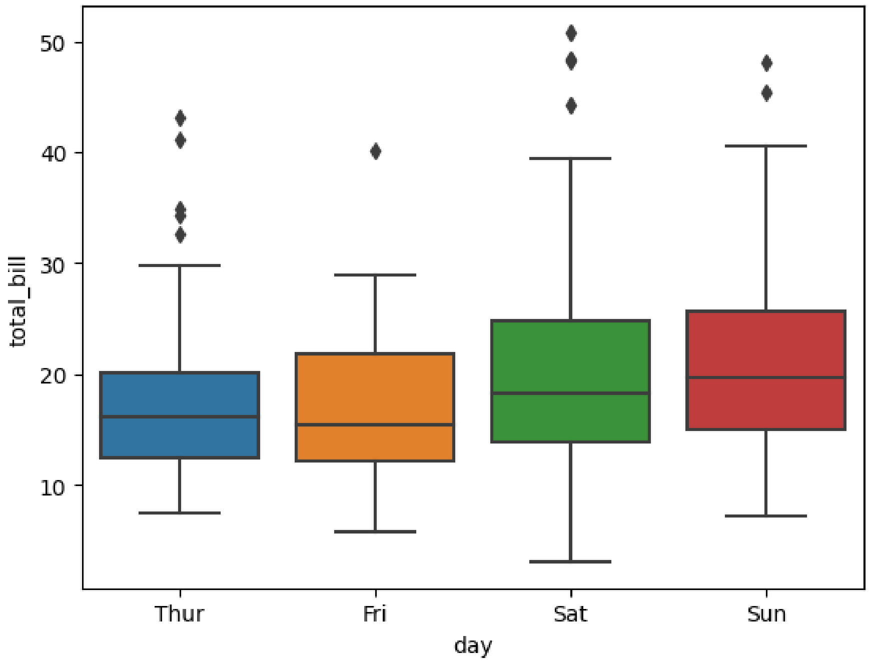
```
# box plot
sns.boxplot(x='smoker', y='total_bill', data=df)
# sns.boxplot(x='a', y='b', data=df)
```

```
<Axes: xlabel='smoker', ylabel='total_bill'>
```



```
sns.boxplot(x='day', y='total_bill', data=df)
# sns.boxplot(x='c', y='b', data=df)
```

```
<Axes: xlabel='day', ylabel='total_bill'>
```

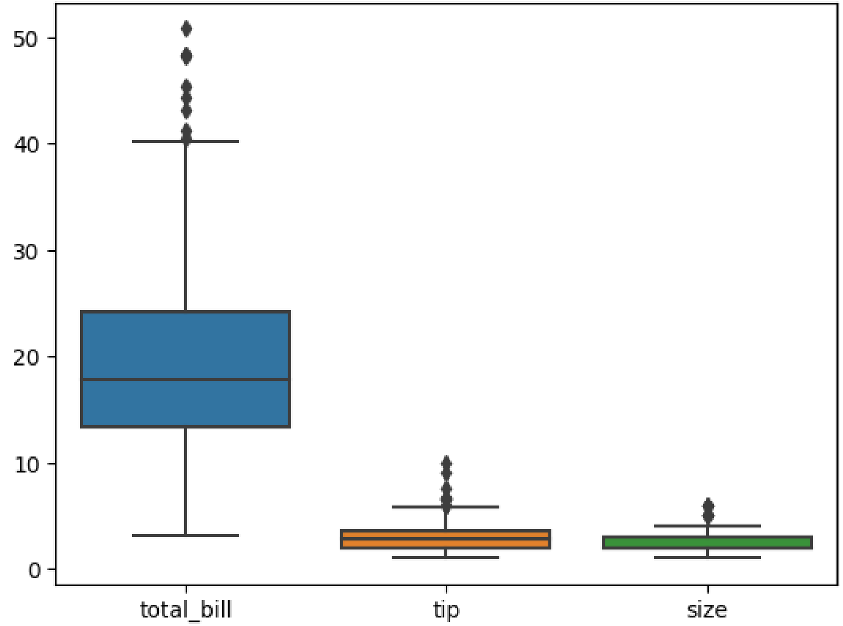


```
sns.boxplot(x='day', y='total_bill', data=df, palette ='rainbow')
# sns.boxplot(x='a', y='b', data=df, palette ='rainbow')
```

```
<Axes: xlabel='day', ylabel='total_bill'>
```

```
sns.boxplot(data=df, orient='v')
# sns.boxplot(data=df, orient='v')
```

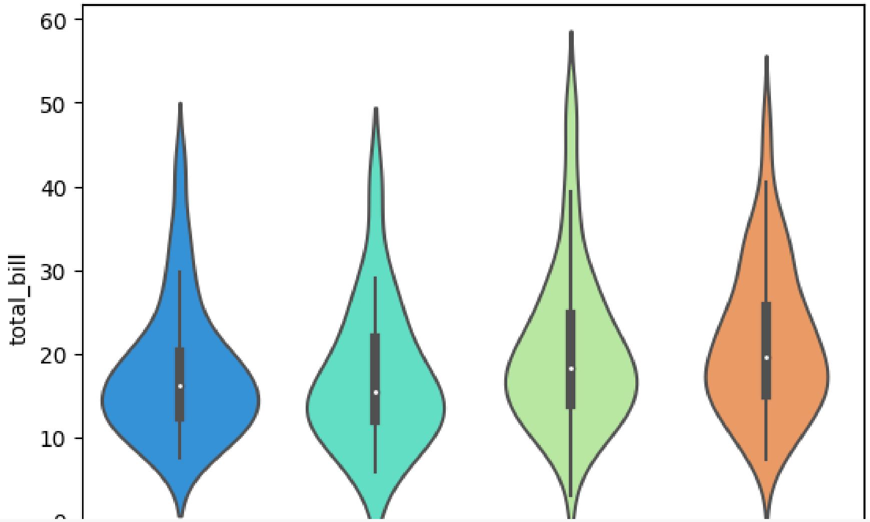
```
<Axes: >
```



violinplot()

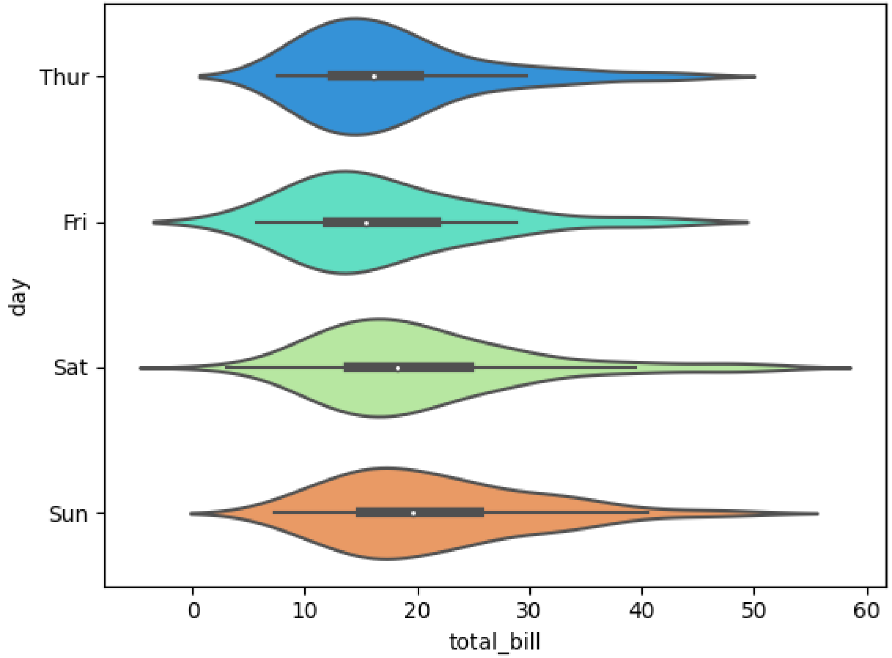
```
sns.violinplot(x='day', y='total_bill', data=df, palette='rainbow')
# sns.violinplot(x='a', y='b', data=df, palette='rainbow')
```

```
<Axes: xlabel='day', ylabel='total_bill'>
```



```
sns.violinplot(x='total_bill', y='day', data=df, palette='rainbow')
# sns.violinplot(x='b', y='a', data=df, palette='rainbow')
```

```
<Axes: xlabel='total_bill', ylabel='day'>
```



HW

- barplot

HW

- bar graph , pie chart, box plot, on character data

HW

- analyse the iris dataset with different plots