## NUMPY

**Create array**

import numpy as np

a=np.array([10,20,30])

**create 2d array**

a=np.array([[10,20,30],[20,30,40],[60,70,80]])

**create a array from range**

a=np.arange(1,11) # it will create an array from 1 to 10

**create array of zeroes**

b=np.zeros(5) # it will 1d create array of zeros which contain 5 zeroes

**create array of one**

b=np.ones(5) # it will 1d create array of ones which contain 5 ones

**create 2d array of zero**

b=np.zeros((5,5))

**create 2d array of zero in int data type**

b=np.zeros((5,5),dtype=np.int16)

**print dimension of array**

print(b.ndim) # it will show the dimension of array such as (5x5 ) or accordingly

**print size of array**

print(b.size) # it shows number of elements in array

**print size of element of array in byte**

print(b.itemsize)

**print shape of array**

print(b.shape)

**reshape array**

b=a.reshape(3,3) #arranging in 3x3 format in this case

b=a.reshape(3,2) # arranging in 3x2 format in this case

**create empty array (may create garbage values)**

a=np.empty([2,3]) #in this case empty array will be of 2x3 format

**addition of numpy array**

print(a+a) # both array should be of same format

**subtraction of numpy array**

print(a-a) # both array should be of same format

**multiplication of numpy array**

print(a\*a) # both array should be of same format

**power of numpy array**

print(a\*\*a) # “\*\*” is used to denote power

**print index range**

print(a[2:6]) # 2 to 5 print action

**print index**

print(a[4])

**print 1d array in reverse order using index**

print(a[::-1])

**print index with skipping**

print(a[1:7:2]) # it will print index 1 then index 3 then index 5

**print index in 2d array**

b[0]

**print element in a index in 2d array**

b[2,1] # it will print element with 1 index of 2 index

**taking index in variable and changing value**

a=np.array([0,1,2,3,4,5,6,7,8,9]) # it will change the value of specific index in b as well as a

b=a[1:5]

b[1]=400

**taking index in variable and changing value without changing value in original array**

a=np.array([0,1,2,3,4,5,6,7,8,9]) #it will change the value of b but not of a

b=a[1:5].copy()

b[1]=400

**sum of array**

a.sum()

**sum of 2d array in vertical direction**

a.sum(axis=0)

**sum of 2d array in horizontal direction**

a.sum(axis=1)

**transpose array**

c.transpose() # rows and columns interchange

**matrix multiplication of arrays**

c.dot(b)

**sort in increasing order**

np.sort(b) #sort in horizontal way

**sort in increasing order**

np.sort(b,axis=0) #sort in vertical way

**sort array in increasing order and display its index**

g.argsort() #arrange array in increasing order and display the index of numbers

**find minimum in array and show its index**

np.argmin(h) # find the minimum number in array and display its index

**create a array of random number**

z=np.random.rand(5,4) #array of 5x4 of random numbers will generate

## PANDAS

**Import panda module**

import pandas as pd

**create series of array**

b=pd.Series(a) # here array is named as a

a=np.array([11,22,33,44,55])

b=pd.Series(a)

b

ANSWER

0 11

1 22

2 33

3 44

4 55

dtype: int32

**create series and put user required index**

b=pd.Series(a,index) # index is another array

# number of element in array a and array of index should be same

index=np.array(['a','b','c','d','e'])

a=np.array([11,22,33,44,55])

b=pd.Series(a,index)

b

ANSWER

a 11

b 22

c 33

d 44

e 55

dtype: int32

**addition of series**

series1+series2 # only the index common to both series will perform the function

**subtraction of series**

series1-series2 # only the index common to both series will perform the function

**multiplication of series**

series1\*series2 # only the index common to both series will perform the function

**devision of series**

series1/series2 # only the index common to both series will perform the function

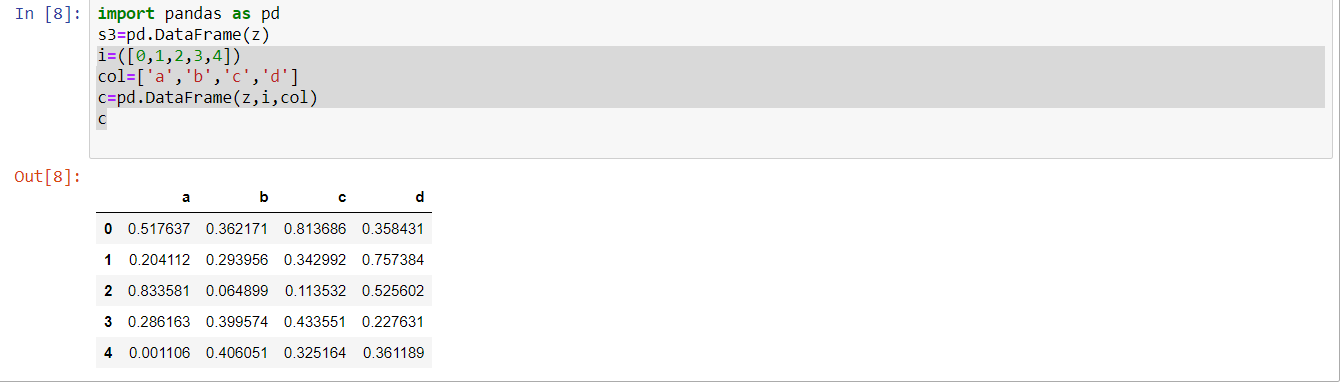
**create data frame**

s3=pd.DataFrame(z) # here s3 is data frame name and z is name of series

**create data frame and put user required index and column name**

c=pd.DataFrame(z,i,col) # here z is series , i is index ,col is column

*example*



**Convert the dataframe to excel sheet**

c.to\_excel("datascience1.xlsx") # c is data frame name

# CSV WORKING

**UPLOAD CSV FILE**

**READ CSV FILE**

import pandas as pd

df=pd.read\_csv("data.csv")

df

**SHOW TOP 5 DATA OF FILE**

df.head()

**SHOW DOWN 5 DATA OF FILE**

df.tail()

**SHOW REQUIRED NUMBER OF DATA FROM TOP**

df.head(10)

**SHOW REQUIRED NUMBER OF DATA FROM DOWN**

df.tail(10)

**READ PARTICULAR EXCEL SHEET**

df1=pd.read\_excel("OFFLINE SEO.xlsx","CLASSIFIED ADS")

df1

**MATPLOTLIB**

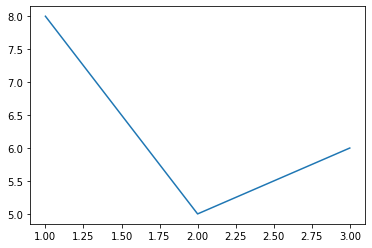
**IMPORT MATPLOTLIB**

from matplotlib import pyplot as plt

**PLOTTING AND PRINTING**

plt.plot([1,2,3],[8,5,6])

plt.show()

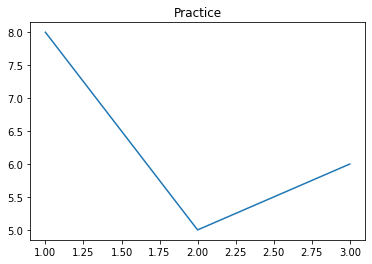


**TITLE**

plt.plot([1,2,3],[8,5,6])

plt.title("Practice")

plt.show()



**LABEL**

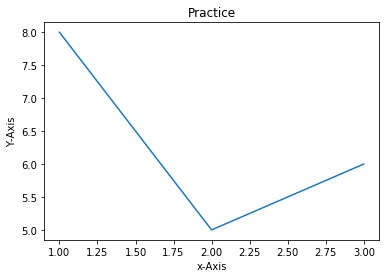
plt.plot([1,2,3],[8,5,6])

plt.title("Practice")

plt.xlabel("x-Axis")

plt.ylabel("Y-Axis")

plt.show()



**PLOTTING TWO GRAPH**

x1=[5,4,8]

y1=[6,4,3]

x2=[8,4,5]

y2=[8,5,6]

plt.plot(x1,y1)

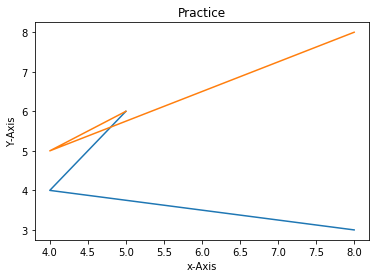
plt.plot(x2,y2)

plt.title("Practice")

plt.xlabel("x-Axis")

plt.ylabel("Y-Axis")

plt.show()



**IMPORT STYLE**

from matplotlib import style

**USING DIFFERENT STYLE**

style.use("ggplot")

x1=[5,4,8]

y1=[6,2,3]

x2=[8,4,5]

y2=[5,5,6]

x3=[2,4,6,8,10]

y3=[6,7,8,2,4]

plt.plot(x1,y1,label="First",color="r",linewidth=3)

plt.scatter(x2,y2,label="Second",color="g")

plt.bar(x3,y3,label="third",color="b")

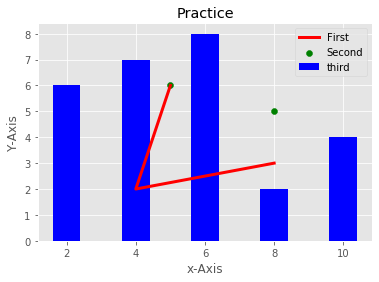
plt.title("Practice")

plt.xlabel("x-Axis")

plt.ylabel("Y-Axis")

plt.legend()

plt.show()



x4=[1,3,5,7,9]

y4=[7,8,2,4,2]

x3=[2,4,6,8,10]

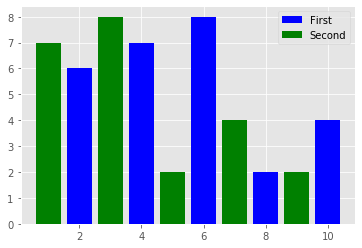
y3=[6,7,8,2,4]

plt.bar(x3,y3,label="First",color="b")

plt.bar(x4,y4,label="Second",color="g")

plt.legend()

plt.show()



**USING FOR LOOP**

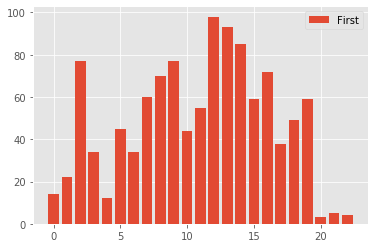
population\_ages=[14,22,77,34,12,45,34,60,70,77,44,55,98,93,85,59,72,38,49,59,3,5,4]

id=[i for i in range(len(population\_ages))]

plt.bar(id,population\_ages,label="First")

plt.legend()

plt.show()

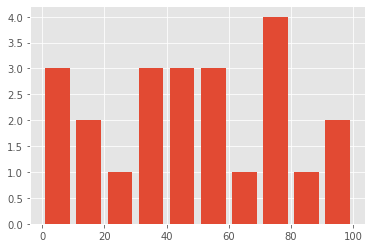


**USING LIST**

range1=[0,10,20,30,40,50,60,70,80,90,100]

plt.hist(population\_ages,range1,histtype='bar',rwidth=0.8)

plt.show()



**STACKPLOT**

day=[1,2,3,4,5]

sleeping=[7,8,6,11,7]

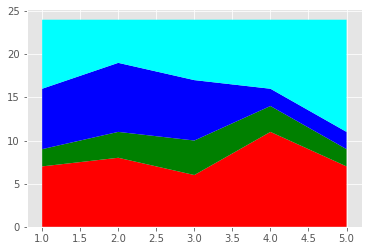
eating=[2,3,4,3,2]

working=[7,8,7,2,2]

playing=[8,5,7,8,13]

plt.stackplot(day,sleeping,eating,working,playing,colors=['red','green','blue','cyan'])

plt.show()



day=[1,2,3,4,5]

sleeping=[7,8,6,11,7]

eating=[2,3,4,3,2]

working=[7,8,7,2,2]

playing=[8,5,7,8,13]

plt.plot([],[],label="sleeping",linewidth=5)

plt.plot([],[],label="eating",linewidth=5)

plt.plot([],[],label="working",linewidth=5)

plt.plot([],[],label="playing",linewidth=5)

plt.stackplot(day,sleeping,eating,working,playing,colors=['blue','yellow','green','cyan'])

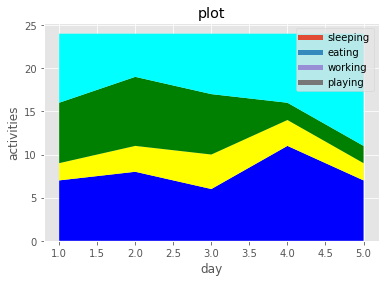
plt.xlabel("day")

plt.ylabel("activities")

plt.title("plot")

plt.legend()

plt.show()



**PIE PLOT**

slice=[7,2,2,13]

activities=['sleeping','eating','working','playing']

cols=['red','green','blue','yellow']

plt.pie(

slice,

labels=activities,

colors=cols,

startangle=90,

explode=(0,0,0.1,0),

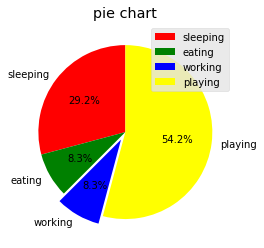
autopct="%1.1f%%"

)

plt.title("pie chart")

plt.legend()

plt.show()



**DIFFERENT PLOT IN ONE**

from matplotlib import pyplot as plt

plt.plot([1,4,7],[3,5,10],label="first sem",linewidth=3,color='r')

plt.scatter([9,7,8],[9,8,5], label="second sem",linewidth=1,color='black')

plt.bar([5,6,9],[4,6,8], label="third sem")

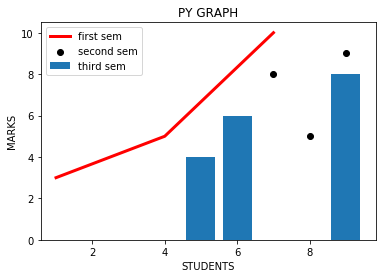
plt.title("PY GRAPH")

plt.xlabel("STUDENTS")

plt.ylabel("MARKS")

plt.legend()

plt.show()



**SEABORN**

**USING SEABORN**

import seaborn as sns

import numpy as np

import pandas as pd

from matplotlib import pyplot as plt

**PRINTING SEABORN VERSION**

print(sns.\_\_version\_\_)

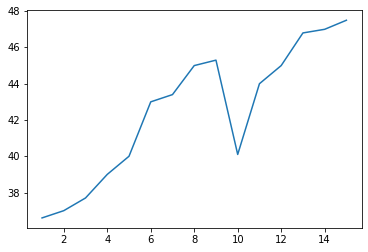
**#LINEPLOT**

**PRINT LINEPLOT**

days=[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]

temperature=[36.6,37,37.7,39,40,43,43.4,45,45.3,40.1,44,45,46.8,47,47.5]

sns.lineplot(x=days,y=temperature)



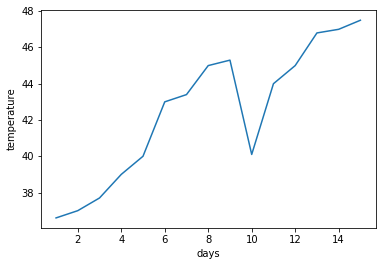
**LINEPLOT USING DATAFRAME**

days=[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]

temperature=[36.6,37,37.7,39,40,43,43.4,45,45.3,40.1,44,45,46.8,47,47.5]

d1=pd.DataFrame({"days":days,"temperature":temperature})

sns.lineplot(x="days",y="temperature" ,data=d1)



**LOAD DATASET EXISTING IN SEABORN**

tips=sns.load\_dataset('tips')

**PRINT DATASET**

Tips

**LAST 5 DATASET ENTRY**

tips.tail()

**TOP 10 DATASEY ENTRY**

tips.head(10)

**LAST 10 DATASET ENTRY**

tips.tail(10)

**DATASET INFORMATION**

tips.info()

**PRINT ONE COLUMN OF DATASET**

tips['tip']

**FINDING MAX VALUE IN ONE COLUMN**

tips['total\_bill'].max()

**FINDING A PARTICULAR VALUE OF DATASET**

tips[tips['total\_bill']==50.81]

**SUM OF COLUMN**

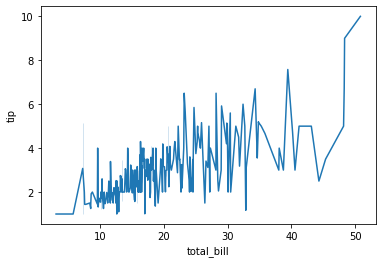
tips['size'].sum()

**SEPERATING DIFFERENT VALUE AND THEIR QUANTITY**

tips['size'].value\_counts()

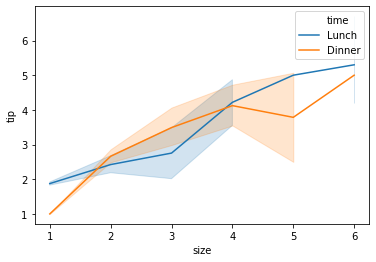
**LINEPLOT FROM DATASET**

sns.lineplot(x="total\_bill" ,y="tip" ,data=tips)



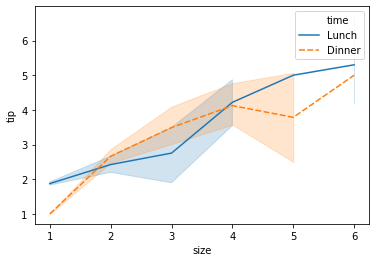
**USING HUE**

sns.lineplot(x="size" ,y="tip" ,data=tips ,hue='time')



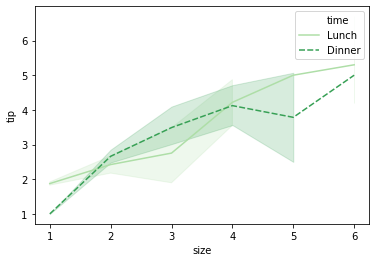
**USING STYLE**

sns.lineplot(x="size" ,y="tip" ,data=tips ,hue='time' , style='time')



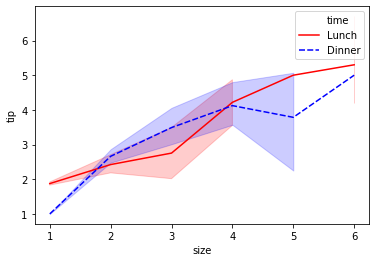
**PALETTE**

sns.lineplot(x="size" ,y="tip" ,data=tips ,hue='time' , style='time' ,palette="Greens")



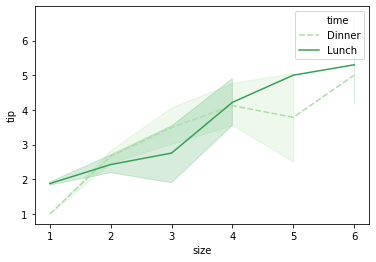
**TWO COLOURS IN PALETTE**

sns.lineplot(x="size" ,y="tip" ,data=tips ,hue='time' , style='time' ,palette=['red','blue'])



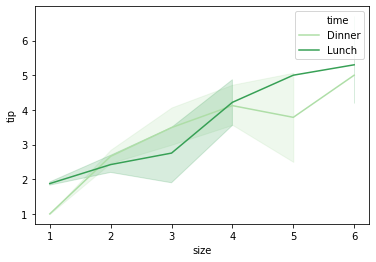
**HUE ORDER (LEGEND ORDER)**

sns.lineplot(x="size" ,y="tip" ,data=tips ,hue='time' , style='time' ,palette="Greens",hue\_order=['Dinner','Lunch'])



**DASHES FALSE**

sns.lineplot(x="size" ,y="tip" ,data=tips ,hue='time' , style='time' ,palette="Greens",hue\_order=['Dinner','Lunch'],dashes=False)



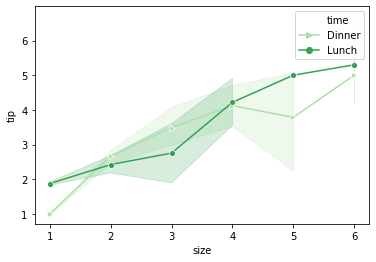
**MARKERS(MARKERS IN LEGEND)**

sns.lineplot(x="size" ,y="tip" ,data=tips ,hue='time' , style='time' ,palette="Greens",hue\_order=['Dinner','Lunch']

,dashes=False,

markers=['o','>']

)



**HIDING LEGEND**

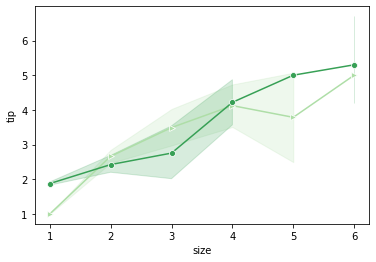
sns.lineplot(x="size" ,y="tip" ,data=tips ,hue='time' , style='time' ,palette="Greens",hue\_order=['Dinner','Lunch']

,dashes=False,

markers=['o','>'],

legend=False

)



**LABEL AND TITLE**

sns.lineplot(x="size" ,y="tip" ,data=tips ,hue='time' , style='time' ,palette="Greens",hue\_order=['Dinner','Lunch']

,dashes=False,

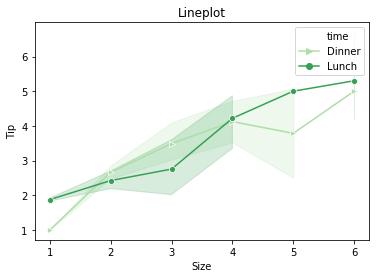
markers=['o','>']

)

plt.title("Lineplot")

plt.xlabel("Size")

plt.ylabel("Tip")



**FONT SIZE OF LABELS AND TITLE**

sns.lineplot(x="size" ,y="tip" ,data=tips ,hue='time' , style='time' ,palette="Greens",hue\_order=['Dinner','Lunch']

,dashes=False,

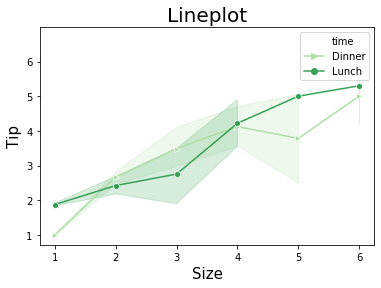
markers=['o','>']

)

plt.title("Lineplot",fontsize=20)

plt.xlabel("Size",fontsize=15)

plt.ylabel("Tip",fontsize=15)



**SEABORN BACKGROUNG STYLE**

sns.set(style='darkgrid')

sns.lineplot(x="size" ,y="tip" ,data=tips ,hue='time' , style='time' ,palette="Greens",hue\_order=['Dinner','Lunch']

,dashes=False,

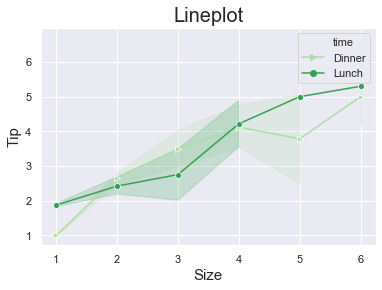
markers=['o','>']

)

plt.title("Lineplot",fontsize=20)

plt.xlabel("Size",fontsize=15)

plt.ylabel("Tip",fontsize=15)



**SIZE OF GRAPH**

plt.figure(figsize=(16,9))

sns.set(style='darkgrid',font\_scale=1.3)

sns.lineplot(x="size" ,y="tip" ,data=tips ,hue='time' , style='time' ,palette="Greens",hue\_order=['Dinner','Lunch']

,dashes=False,

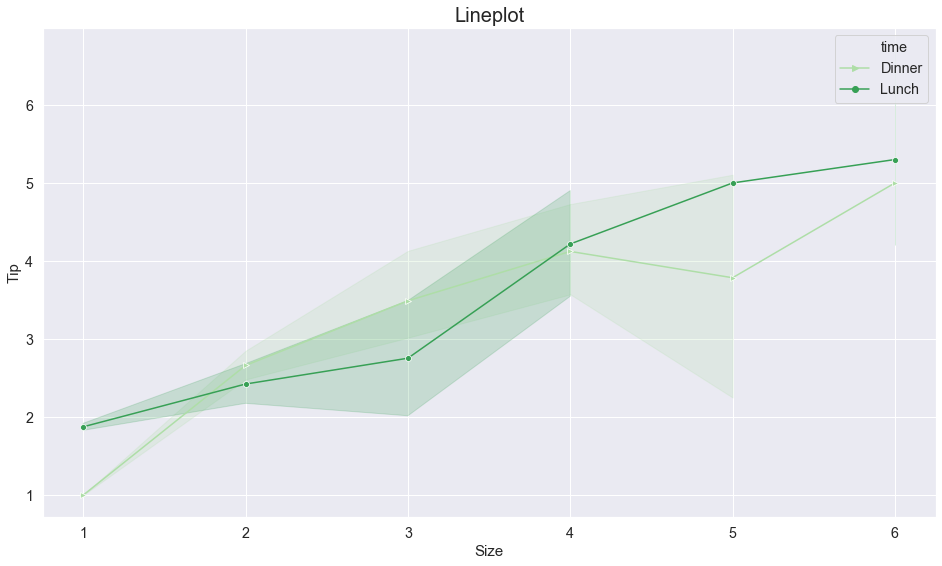
markers=['o','>']

)

plt.title("Lineplot",fontsize=20)

plt.xlabel("Size",fontsize=15)

plt.ylabel("Tip",fontsize=15)



.

**SAVING PLOT**

sns.set(style='darkgrid',font\_scale=1.3)

sns.lineplot(x="size" ,y="tip" ,data=tips ,hue='time' , style='time' ,palette="Greens",hue\_order=['Dinner','Lunch']

,dashes=False,

markers=['o','>']

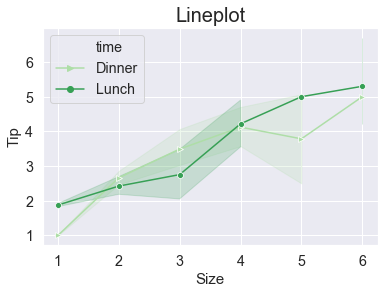
)

plt.title("Lineplot",fontsize=20)

plt.xlabel("Size",fontsize=15)

plt.ylabel("Tip",fontsize=15)

plt.savefig("lineplot1")



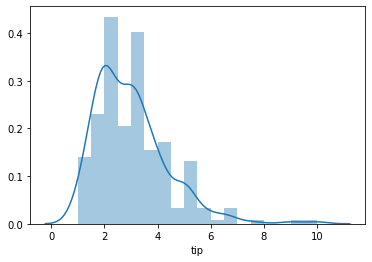
**#DISTPLOT**

**LOAD DATASET**

tip\_df=sns.load\_dataset('tips')

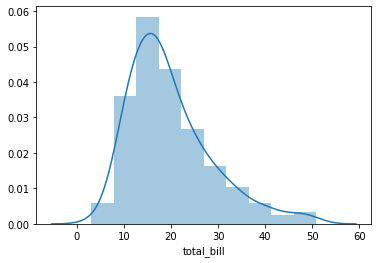
**DISTPLOT**

sns.distplot(tip\_df['tip']) #kde=kernel density estimator(line)

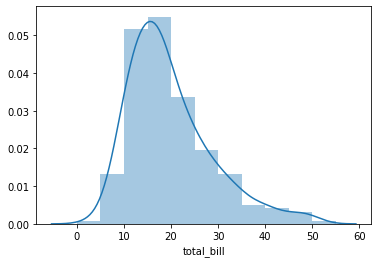


**BINS**

sns.distplot(tip\_df['total\_bill'],bins=10)

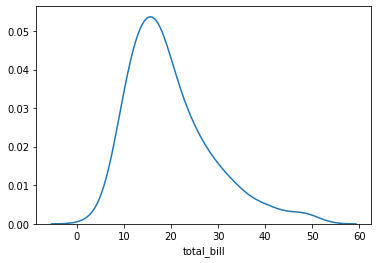


sns.distplot(tip\_df['total\_bill'],bins=[0,5,10,15,20,25,30,35,40,45,50,55])



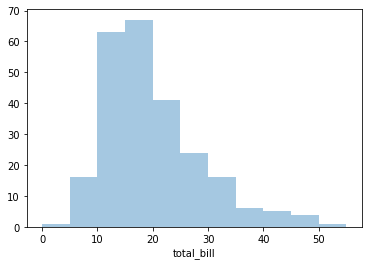
**HISTOGRAM FALSE**

sns.distplot(tip\_df['total\_bill'],bins=[0,5,10,15,20,25,30,35,40,45,50,55],hist=False)



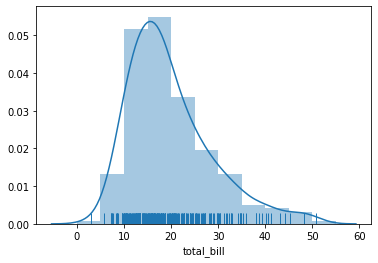
**KDE FALSE**

sns.distplot(tip\_df['total\_bill'],bins=[0,5,10,15,20,25,30,35,40,45,50,55],kde=False).



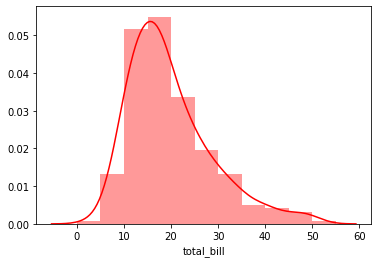
**RUG**

sns.distplot(tip\_df['total\_bill'],bins=[0,5,10,15,20,25,30,35,40,45,50,55],rug=True)



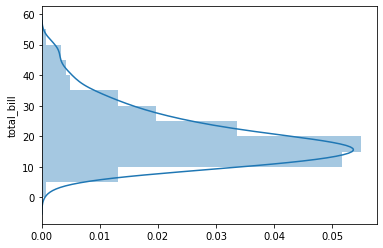
**COLOR OF HISTOGRAM**

sns.distplot(tip\_df['total\_bill'],bins=[0,5,10,15,20,25,30,35,40,45,50,55],color='r')



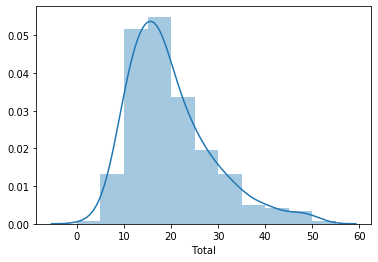
**CHANGING ORIENTATION**

sns.distplot(tip\_df['total\_bill'],bins=[0,5,10,15,20,25,30,35,40,45,50,55],vertical=True)



**HORIZONTAL LABEL**

sns.distplot(tip\_df['total\_bill'],bins=[0,5,10,15,20,25,30,35,40,45,50,55],axlabel="Total")

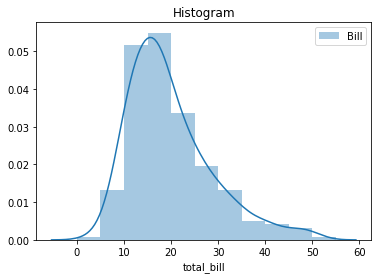


**LEGEND TITLE**

sns.distplot(tip\_df['total\_bill'],bins=[0,5,10,15,20,25,30,35,40,45,50,55],label="Bill")

plt.legend()

plt.title("Histogram")



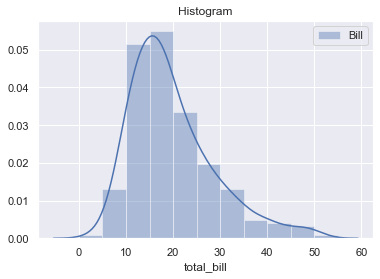
**SET FUNCTION**

sns.set()

sns.distplot(tip\_df['total\_bill'],bins=[0,5,10,15,20,25,30,35,40,45,50,55],label="Bill")

plt.legend()

plt.title("Histogram")



**FIGURE SIZE (GRAPH SIZE**)

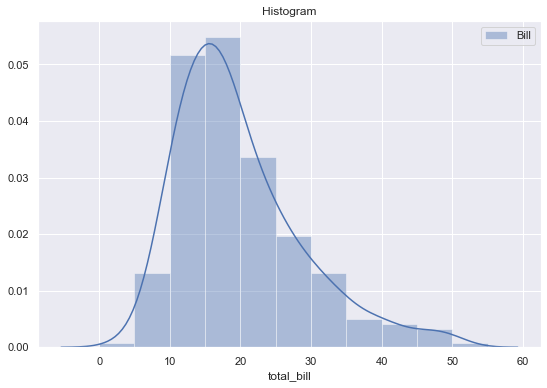
plt.figure(figsize=(9,6))

sns.set()

sns.distplot(tip\_df['total\_bill'],bins=[0,5,10,15,20,25,30,35,40,45,50,55],label="Bill")

plt.legend()

plt.title("Histogram")



**HISTOGRAM KWS**

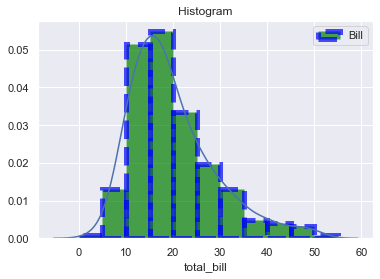
sns.set()

sns.distplot(tip\_df['total\_bill'],bins=[0,5,10,15,20,25,30,35,40,45,50,55],

label="Bill",hist\_kws={'color':'green','edgecolor':'blue','linewidth':5,'linestyle':'--','alpha':0.7})

plt.legend()

plt.title("Histogram")



**KDE KWS**

sns.set()

sns.distplot(tip\_df['total\_bill'],bins=[0,5,10,15,20,25,30,35,40,45,50,55],

label="Bill",

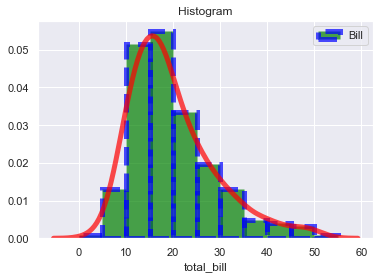
hist\_kws={'color':'green','edgecolor':'blue','linewidth':5,'linestyle':'--','alpha':0.7},

kde\_kws={'color':'red','linewidth':5,'linestyle':'-','alpha':0.7}

)

plt.legend()

plt.title("Histogram")



**RUG KWS**

sns.set()

sns.distplot(tip\_df['total\_bill'],bins=[0,5,10,15,20,25,30,35,40,45,50,55],

label="Bill",rug=True,

hist\_kws={'color':'green','edgecolor':'blue','linewidth':5,'linestyle':'--','alpha':0.7},

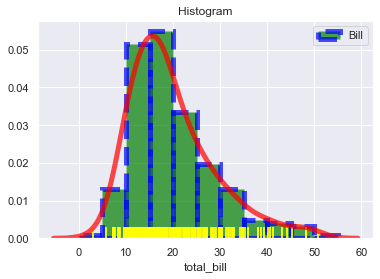
kde\_kws={'color':'red','linewidth':5,'linestyle':'-','alpha':0.7},

rug\_kws={'color':'black','edgecolor':'yellow','linewidth':2,'linestyle':'-','alpha':0.9}

)

plt.legend()

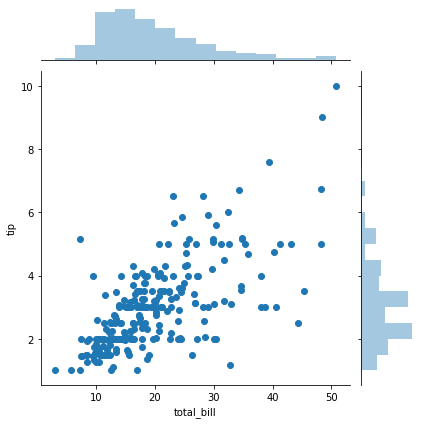
plt.title("Histogram")



**# JOINTPLOT**

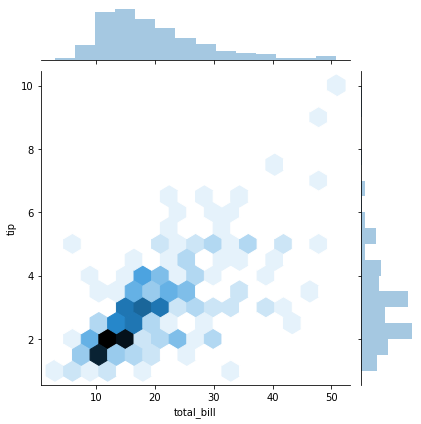
**PRINT JOINT PLOT**

sns.jointplot(tip['total\_bill'],tip['tip'],tip)



**KIND (STYLE)**

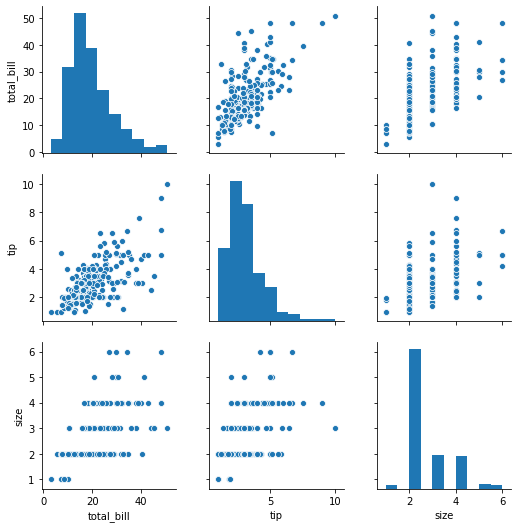
sns.jointplot(tip['total\_bill'],tip['tip'],tip,kind='hex')



**#PAIRPLOT**

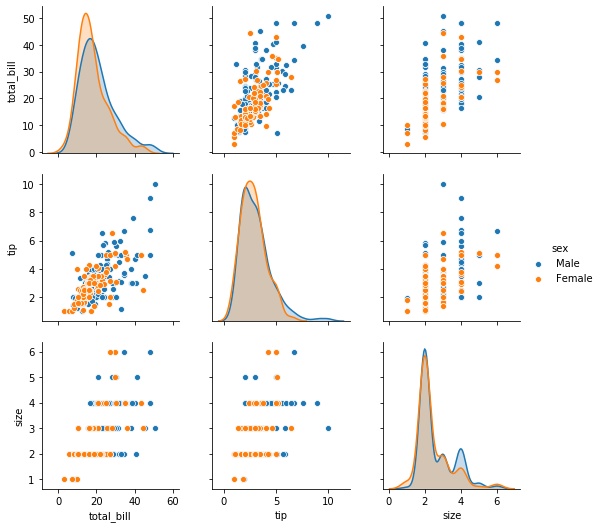
**PRINT PAIRPLOT**

sns.pairplot(tip)

****

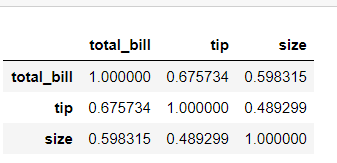
**HUE**

sns.pairplot(tip,hue='sex')



**CORRELATION**

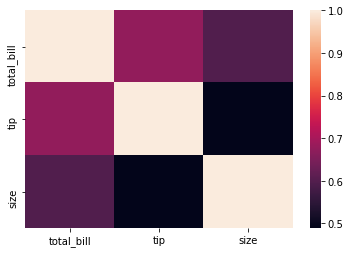
tip.corr() #correlation (maximum 1)



**#HEATMAP**

**PRINT HEATMAP**

sns.heatmap(tip.corr())



**#COUNTPLOT**

**PRINT COUNTPLOT**

sns.countplot(x=tip['sex'],data=tip)

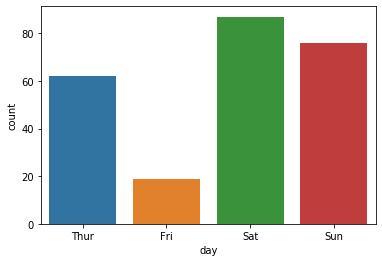


**COUNTING VALUES**

tip['sex'].value\_counts()

**DATA ASSIGNING**

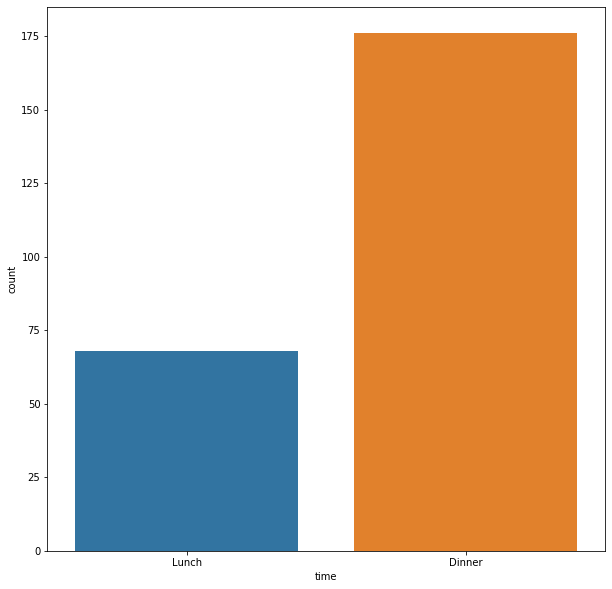
tip['sex'].value\_counts()

****

**FIGURE SIZE**

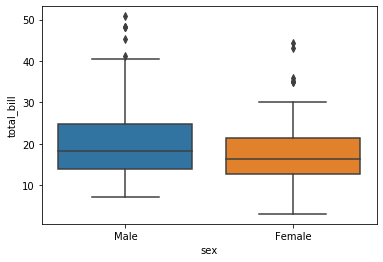
plt.figure(figsize=(10,10))

sns.countplot(x=tip['time'],data=tip)



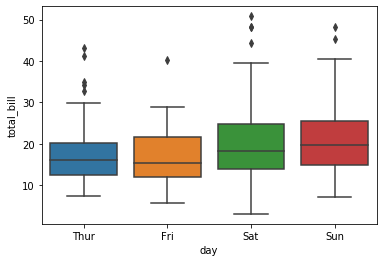
**#BOXPLOT**

sns.boxplot(x='sex',y='total\_bill',data=tip)



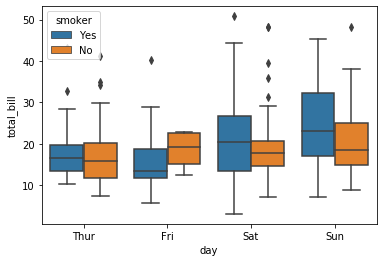
**DATA ASSIGNING**

sns.boxplot(x='day',y='total\_bill',data=tip)

****

**HUE**

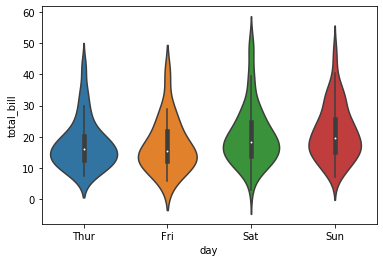
sns.boxplot(x='day',y='total\_bill',data=tip,hue='smoker')



**#VIOLINPLOT**

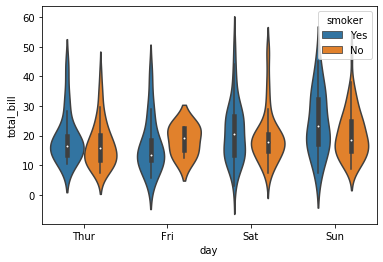
**PRINT VIOLINPLOT**

sns.violinplot(x='day',y='total\_bill',data=tip)



**HUE**

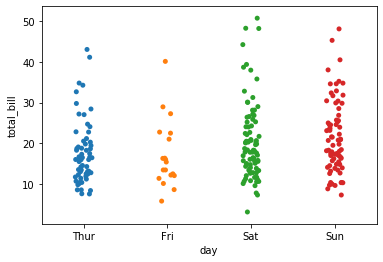
sns.violinplot(x='day',y='total\_bill',data=tip,hue='smoker')



**# STRIPPLOT**

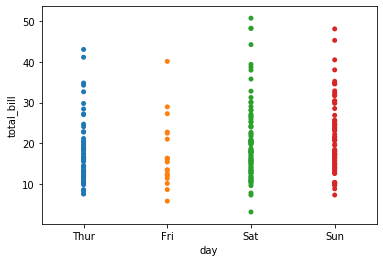
**PRINT STRIPPLOT**

sns.stripplot(x=tip['day'],y=tip['total\_bill'])



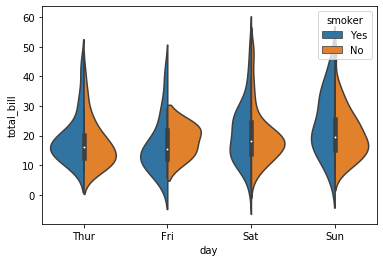
**JITTER**

sns.stripplot(x=tip['day'],y=tip['total\_bill'],jitter=False)

****

**SPLIT**

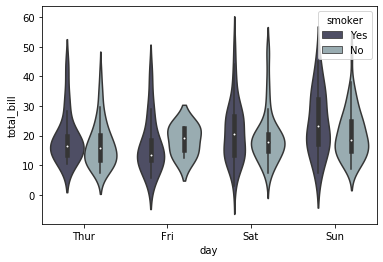
sns.violinplot(x='day',y='total\_bill',data=tip,hue='smoker',split=True)



**PALATTE**

sns.violinplot(x='day',y='total\_bill',data=tip,hue='smoker',palette='bone',plit=True)

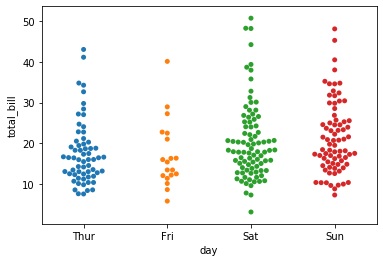
plt.figure(figsize=(10,10))



**#SWARM PLOT**

**PRINT SWARM PLOT**

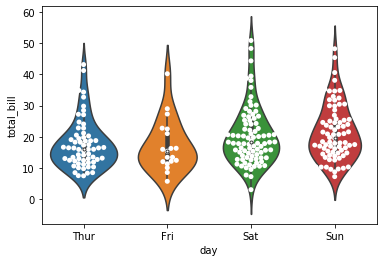
sns.swarmplot(x='day',y='total\_bill',data=tip)



**VIOLIN PLOT AND SWARM PLOT**

sns.swarmplot(x='day',y='total\_bill',data=tip,color='white')

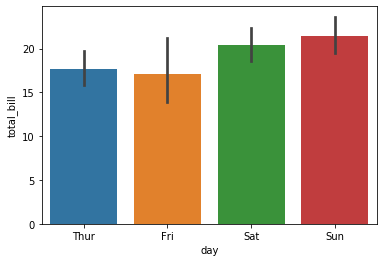
sns.violinplot(x='day',y='total\_bill',data=tip)



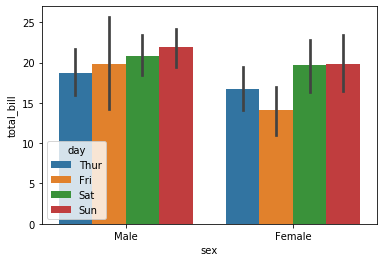
**#BARPLOT**

**PRINT BARPLOT**

sns.barplot(x=tips\_df['day'],y=tips\_df['total\_bill'])

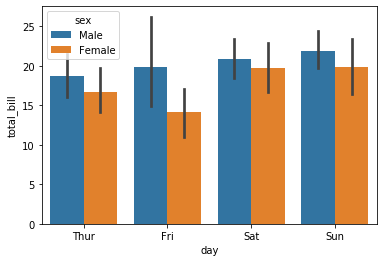


sns.barplot(x='sex',y='total\_bill',hue='day',data=tips\_df)

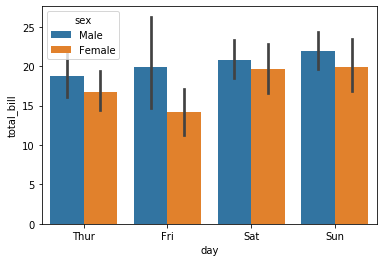


**HUE**

sns.barplot(x=tips\_df['day'],y=tips\_df['total\_bill'],hue=tips\_df['sex'])



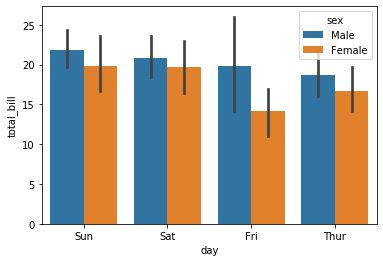
sns.barplot(x='day',y='total\_bill',hue=tips\_df['sex'],data=tips\_df)



**ORDER X-AXIS**

ord=['Sun','Sat','Fri','Thur']

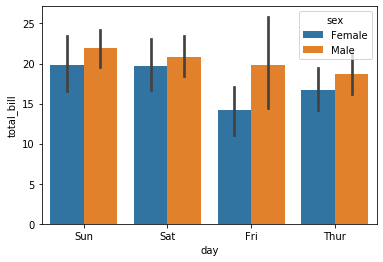
sns.barplot(x='day',y='total\_bill',hue=tips\_df['sex'],data=tips\_df,order=ord)



**HUE ORDER**

ord=['Sun','Sat','Fri','Thur']

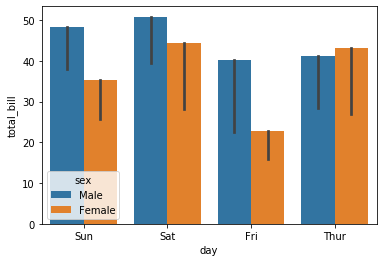
sns.barplot(x='day',y='total\_bill',hue='sex',data=tips\_df,order=ord,hue\_order=['Female','Male'])



**ESTIMATOR**

ord=['Sun','Sat','Fri','Thur']

sns.barplot(x='day',y='total\_bill',hue=tips\_df['sex'],data=tips\_df,order=ord,estimator=np.max)



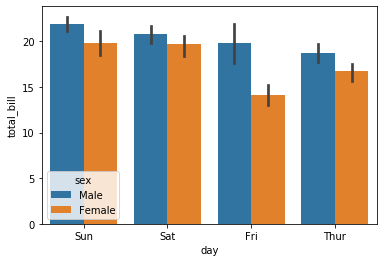
**ESTIMATOR MEAN**

ord=['Sun','Sat','Fri','Thur']

sns.barplot(x='day',y='total\_bill',hue=tips\_df['sex'],data=tips\_df,order=ord,estimator=np.mean

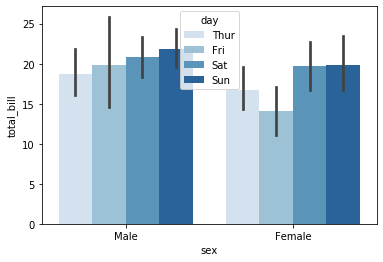
,ci=52

)



**PALATTE**

sns.barplot(x='sex',y='total\_bill',hue='day',data=tips\_df,palette='Blues')



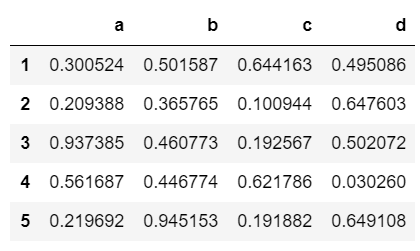
**Fill na value**

**import numpy as np**

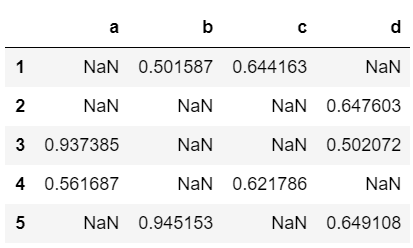
**import pandas as pd**

**df=pd.DataFrame(np.random.rand(5,4),index=[1,2,3,4,5],columns=["a","b","c","d"])**

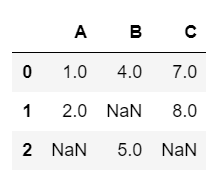
**df**



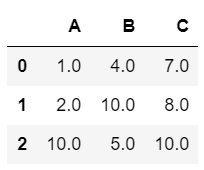
**df[df>0.5]**



**df2=pd.DataFrame({'A':[1,2,np.nan],'B':[4,np.nan,5],'C':[7,8,np.nan]})**



**df2.fillna(10)**



**df2['A'].fillna(df2['A'].mean())**

