INFOSYS SPRING BOARD INTERNSHIP 6.0

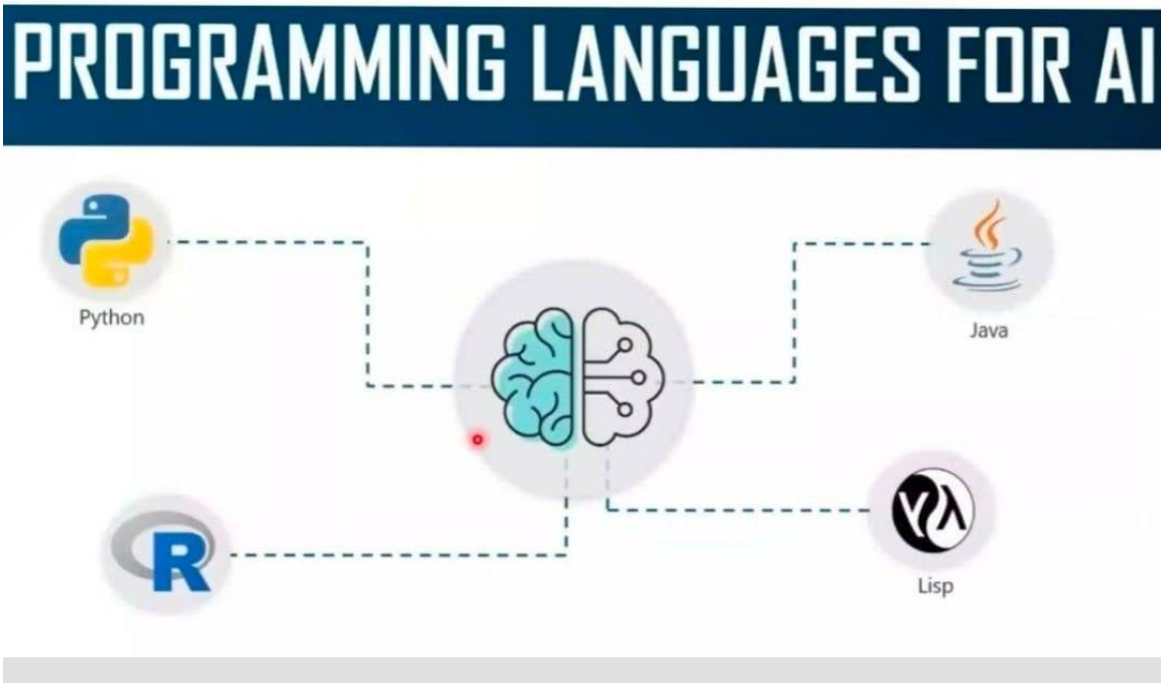
"BUDGETWISE AI BASED EXPENSE FORECASTING TOOL"

ARTIFICIAL INTELLIGENCE (AI):

John McCarthy is the father of AI in the year 1955.It is defined as the science and engineering of making intelligent machines, especially intelligent computer programs.

APPLICATIONS OF AI:

1. Healthcare
2. Robotics
3. Agriculture
4. Social media

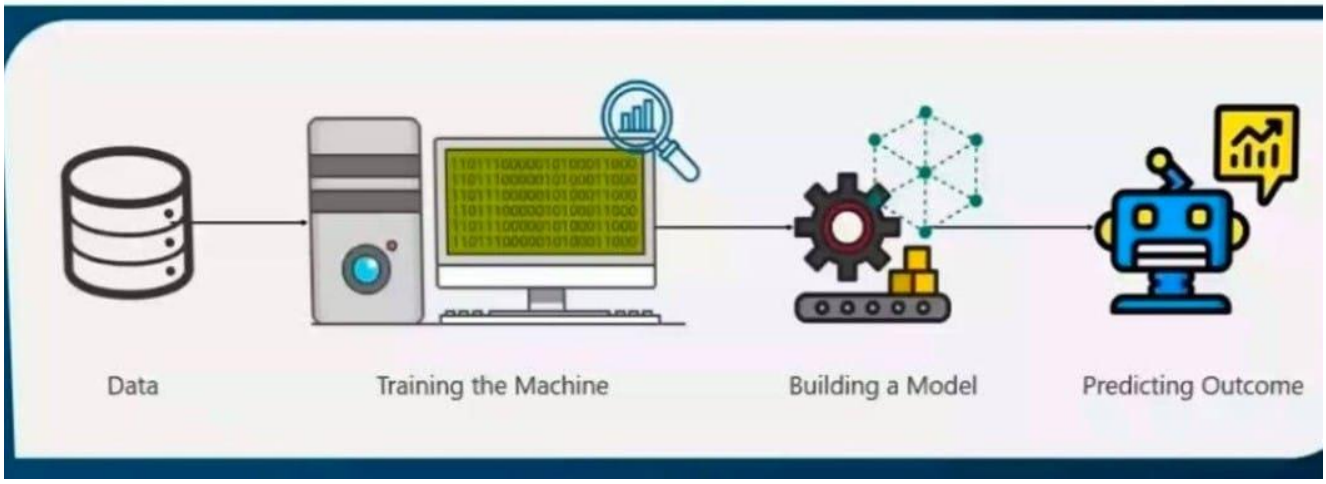


MACHINE LEARNING:

Machine Learning ML was first coined by Arthur Samuel in the year 1959. Machine learning is a subset of AI which provides machines the ability to learn automatically and improve from experience without being explicitly programmed.

APPLICATIONS OF ML:

1. Image recognition
2. Speech recognition
3. Self driving cars
4. Email spam



NUMPY MATRIX:

Numpy stands for "Numerical Python" and is the core library for numeric and scientific computing. It consists of multi dimensional array objects and a collection of routines for processing those arrays.

SINGLE DIMENSIONAL ARRAY:

```
n1=np.array([10,20,30,40,50])
n1
```

O/P: array([10, 20, 30, 40, 50])

MULTIDIMENSIONAL ARRAY:

```
import numpy as np
n2=np.array([[10,20,30],[40,50,60]])
n2
```

O/P: array([[10, 20, 30],
[40, 50, 60]])

INITIALIZING NUMPY ARRAY WITH ZEROS:

```
n1=np.zeros((1,2))
n1
```

O/P: array([[0., 0.]])

- n1=np.zeros((5,5))
  n1

O/P: array([[0., 0., 0., 0., 0.],
[0., 0., 0., 0., 0.],
[0., 0., 0., 0., 0.],
[0., 0., 0., 0., 0.],
[0., 0., 0., 0., 0.]])

INITIALIZING NUMPY ARRAY WITH SAME NUMBERS:

- n1=np.full((2,2),10)
  n1

O/P: array([[10, 10],

[10, 10]])

INITIALIZING NUMPY ARRAY:

- n1=np.arange(10,20)

  n1

O/P: array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19])

- import numpy as np

  n1=np.arange(10,50,5)

  n1

O/P: array([10, 15, 20, 25, 30, 35, 40, 45])

INITIALISING NUMPY ARRAY WITH RANDOM NUMBERS:

- n1=np.random.randint(1,100,5)

  n1

O/P: array([69, 68, 94, 66, 72])

NUMPY SHAPE:

- n1=np.array([[1,2,3],[4,5,6]])

  n1.shape

O/P: (2, 3)

- n1.shape=(3,2)

  n1.shape

O/P: (3, 2)

JOINING NUMPY ARRAY:

#VSTACK()

```
n1=np.array([10,20,30])
n2=np.array([40,50,60])
np.vstack((n1,n2))
```

O/P: array([[10, 20, 30],
[40, 50, 60]])

#HSTACK()

```
n1= np.array([10, 20, 30])
n2=np.array([40,50,60])
np.hstack((n1,n2))
```

O/P: array([10, 20, 30, 40, 50, 60])

#COLUMN_STACK()

```
n1=np.array([10,20, 30])
n2=np.array([40,50,60])
np.column_stack((n1,n2))
```

O/P: array([[10, 40],
[20, 50],
[30, 60]])

NUMPY TRANSPOSE:

#TRANSPOSE()

import numpy as np

n1.transpose()

O/P: array([10, 20, 30])

MATRIX MULTIPLICATION:

n1=np.array([[1,2,3],[4,5,6],[7,8,9]])

n1

O/P: array([[1, 2, 3],
[4, 5, 6],
[7, 8, 9]])

- import numpy as np
  n2 = np.array([[1, 2], [3, 4], [5, 6]])
  n1.dot(n2)

O/P: array([[22, 28],
[49, 64],
[76, 100]])

- n2=np.array([[7,8,9],[6,5,4],[5,4,3]])
  n2

O/P: array([[7, 8, 9],
[6, 5, 4],
[5, 4, 3]])

- import numpy as np

```
n1= np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
n2=np.array([[7,8,9],[6,5,4],[5,4,3]])
n2.dot(n1.T)
```

O/P: array([[50, 122, 194],
[28, 73, 118],
[22, 58, 94]])

PANDAS:

Pandas stand for panel Data and is the core library for data analysis. It consists of single & multidimensional data structures for data manipulation.

PANDAS SERIES OBJECT:

```
#SERIES OBJECT IS ONE-DIMENSIONAL LABELED ARRAY
import pandas as pd
s1=pd.Series([9,7,6,4,5])
s1
```

O/P:

	0
0	9
1	7

	0
2	6
3	4
4	5

dtype: int64

CHANGING INDEX:

```
import pandas as pd
s1=pd.Series([1,2,3],index=['x','y','z'])
s1
```

O/P:

	0
x	1
y	2
z	3

dtype: int64

SERIES OBJECT FROM DICTIONARY: You can also create a series object from a dictionary!!

```
import pandas as pd
pd.Series({'a':10,'b':20,'c':30})
```

O/P:

```
0
a  10
b  20
c  30
dtype: int64
```

CHANGING INDEX POSITION: We can Change the index positions.

```
import pandas as pd
pd.Series({'a':10,'b':20,'c':30},index=['b','a','g'])
```

O/P:

```
0
b  20.0
```

0

a 10.0

g NaN

dtype: float64

EXTRACTING INDIVIDUAL ELEMENTS:

#EXTRACTING A SINGLE ELEMENT

s1=pd.Series([1,2,3,4,5,6,7,8,9,10])

s1[5]

O/P: np.int64(6)

#EXTRACTING ELEMENTS FROM BACK

s1=pd.Series([1,2,3,4,5,6,7,8,9,10])

s1[-3:]

O/P:

0

7 8

0

8 9

9 10

dtype: int64

#EXTRACTING A SEQUENCE OF ELEMENTS

s1=pd.Series([1,2,3,4,5,6,7,8,9,10])

s1[:5]

O/P:

	0
0	1
1	2
2	3
3	4
4	5

dtype: int64

#SUM

```
import numpy as np
n1=np.array([10,20])
n2=np.array([30,40])
np.sum([n1,n2])
```
O/P: np.int64(100)
```
#column wise
np.sum([n1,n2],axis=0)
```
O/P: array([40, 60])
```
#row wise
np.sum([n1,n2],axis=1)
```
O/P: array([30, 70])
```
#ADDITION
import numpy as np
n1=np.array([10,20,30])
n1=n1+1
n1
```
O/P: array([11, 21, 31])
```
#MULTIPLICATION
import numpy as np
n1=np.array([10,20,30])
n1=n1*2
```

n1

O/P: array([20, 40, 60])

#SUBTRACTION

```
import numpy as np
n1=np.array([10,20,30])
n1=n1-1
n1
```

O/P: array([9, 19, 29])

#DIVISION

```
import numpy as np
n1=np.array([10,20,30])
n1=n1/2
n1
```

O/P: array([5., 10., 15.])

#MEAN

```
import numpy as np
n1=np.array([10,20,30,40,50,60])
np.mean(n1)
```

O/P: np.float64(35.0)

#standard deviation

```
import numpy as np
```

n1=np.array([1,5,3,100,4,48])

np.std(n1)

O/P: np.float64(36.59424666377065)

#MEDIAN

import numpy as np

n1=np.array([11,45,5,96,67,85])

np.median(n1)

O/P: np.float64(56.0)

NUMPY MATRIX:

n1[0]

O/P: array([1, 2, 3])

n1[1]

O/P: array([4, 5, 6])

n1[:,1]

O/P: n1[:,1]

n1[:,2]

O/P: array([3, 6, 9])

BASIC MATH OPERATIONS ON SERIES:

#Adding two series objects

import pandas as pd

s1 = pd.Series([1,2,3,4,5,6,7,8,9])

s2 = pd.Series([10,20,30,40,50,60,70,80,90])

s1+s2

O/P:

	0
0	11
1	22
2	33
3	44
4	55
5	66
6	77
7	88
8	99

dtype: int64

#Adding a scalar value to series elements

s1 +5

O/P:

	0
0	6
1	7
2	8
3	9
4	10
5	11
6	12
7	13
8	14

dtype: int64

DATAFRAME:

This is how we can create a dataframe.Dataframe is a 2-dimensional labelled data structure. A dataframe comprises of rows and columns.

```
import pandas as pd
pd.DataFrame({"Name":['bob','sam','anne'],'Marks':[76,25,92]})
```

O/P:

	Name	Marks
0	bob	76
1	sam	25
2	anne	92

- df=pd.DataFrame({"Name":["sam","anne","jennifer"],"Marks":[50,60,70]})

  df

O/P:

	Name	Marks
0	sam	50
1	anne	60
2	jennifer	70

- type(df)

O/P: **pandas.core.frame.DataFrame**

def __init__(data=None, index: Axes | None=None, columns: Axes

| None=None, dtype: Dtype | None=None, copy: bool | None=None) -> None

UPLOADING IRIS DATASET:

iris=pd.read_csv('Iris (1).csv')

from google.colab import files

uploaded = files.upload()

for fn in uploaded.keys():

print('User uploaded file "{name}" with length {length} bytes'.format(

name=fn, length=len(uploaded[fn])))

O/P: Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Saving Iris.csv to Iris (1).csv

User uploaded file "Iris (1).csv" with length 5107 bytes

- iris.head()

O/P:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

- import pandas as pd
- ds = pd.read_csv('Iris (1).csv')
- ds.head()

O/P:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

- ds.head(20)

O/P:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
5	6	5.4	3.9	1.7	0.4	Iris-setosa
6	7	4.6	3.4	1.4	0.3	Iris-setosa
7	8	5.0	3.4	1.5	0.2	Iris-setosa
8	9	4.4	2.9	1.4	0.2	Iris-setosa
9	10	4.9	3.1	1.5	0.1	Iris-setosa
10	11	5.4	3.7	1.5	0.2	Iris-setosa
11	12	4.8	3.4	1.6	0.2	Iris-setosa
12	13	4.8	3.0	1.4	0.1	Iris-setosa
13	14	4.3	3.0	1.1	0.1	Iris-setosa
14	15	5.8	4.0	1.2	0.2	Iris-setosa
15	16	5.7	4.4	1.5	0.4	Iris-setosa
16	17	5.4	3.9	1.3	0.4	Iris-setosa
17	18	5.1	3.5	1.4	0.3	Iris-setosa
18	19	5.7	3.8	1.7	0.3	Iris-setosa
19	20	5.1	3.8	1.5	0.3	Iris-setosa

- ds.tail()

O/P:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

- ds.tail(10)

O/P:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
140	141	6.7	3.1	5.6	2.4	Iris-virginica
141	142	6.9	3.1	5.1	2.3	Iris-virginica
142	143	5.8	2.7	5.1	1.9	Iris-virginica
143	144	6.8	3.2	5.9	2.3	Iris-virginica
144	145	6.7	3.3	5.7	2.5	Iris-virginica
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

- ds.shape

O/P:

(150, 6)

- ds.describe()

O/P:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

ILOC:

iris.iloc[0:3,0:2]

O/P:

	Id	SepalLengthCm
0	1	5.1
1	2	4.9
2	3	4.7

iris.iloc[0:9,0:8]

O/P:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
5	6	5.4	3.9	1.7	0.4	Iris-setosa
6	7	4.6	3.4	1.4	0.3	Iris-setosa
7	8	5.0	3.4	1.5	0.2	Iris-setosa
8	9	4.4	2.9	1.4	0.2	Iris-setosa

ds.iloc[0:3,2:5]

O/P:

	SepalWidthCm	PetalLengthCm	PetalWidthCm
0	3.5	1.4	0.2
1	3.0	1.4	0.2
2	3.2	1.3	0.2

DROPPING COLUMNS:

iris.drop('SepalLengthCm',axis=1)

O/P:

	Id	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	3.5	1.4	0.2	Iris-setosa
1	2	3.0	1.4	0.2	Iris-setosa
2	3	3.2	1.3	0.2	Iris-setosa
3	4	3.1	1.5	0.2	Iris-setosa
4	5	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	146	3.0	5.2	2.3	Iris-virginica
146	147	2.5	5.0	1.9	Iris-virginica
147	148	3.0	5.2	2.0	Iris-virginica
148	149	3.4	5.4	2.3	Iris-virginica
149	150	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

ds.loc[1:5,('SepalLengthCm','PetalLengthCm')]

O/P:

	SepalLengthCm	PetalLengthCm
1	4.9	1.4
2	4.7	1.3
3	4.6	1.5
4	5.0	1.4
5	5.4	1.7

LOC:

iris.loc[0:3,("SepalLengthCm","PetalLengthCm")]

O/P:

	SepalLengthCm	PetalLengthCm
0	5.1	1.4
1	4.9	1.4
2	4.7	1.3
3	4.6	1.5

DROPPING ROWS:

iris.drop([1,2,3],axis=0)

O/P:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
5	6	5.4	3.9	1.7	0.4	Iris-setosa
6	7	4.6	3.4	1.4	0.3	Iris-setosa
7	8	5.0	3.4	1.5	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

147 rows × 6 columns

MORE PANDAS FUNCTIONS:

iris.mean(numeric_only=True)

O/P:

	0
Id	75.500000
SepalLengthCm	5.843333
SepalWidthCm	3.054000
PetalLengthCm	3.758667
PetalWidthCm	1.198667

dtype: float64

iris.median(numeric_only=True)

O/P:

	0
Id	75.50
SepalLengthCm	5.80
SepalWidthCm	3.00

	0
PetalLengthCm	4.35
PetalWidthCm	1.30

dtype: float64

iris.min()

O/P:

	0
Id	1
SepalLengthCm	4.3
SepalWidthCm	2.0
PetalLengthCm	1.0
PetalWidthCm	0.1
Species	Iris-setosa

dtype: object

iris.max()

O/P:

	0
Id	150
SepalLengthCm	7.9
SepalWidthCm	4.4
PetalLengthCm	6.9
PetalWidthCm	2.5
Species	Iris-virginica

dtype: object

LINE PLOT:

```
import numpy as np
from matplotlib import pyplot as plt
x=np.arange(1,11)
y= 2*x
y
```

O/P:

```
array([ 2,  4,  6,  8, 10, 12, 14, 16, 18, 20])
```

x=np.arange(1,11)

x

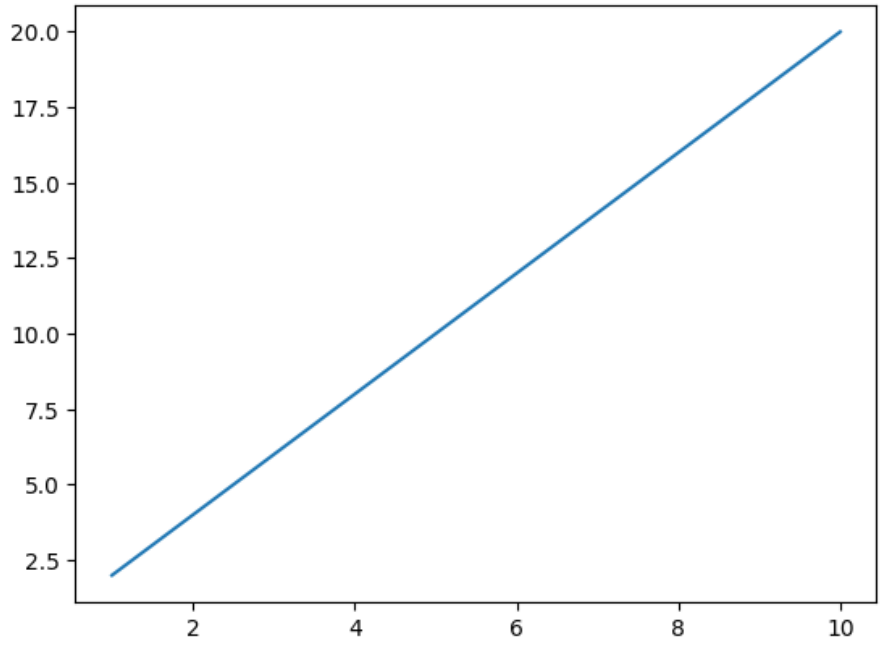
O/P:

array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])

plt.plot(x,y)

plt.show()

O/P:



LINE PLOT: Adding titiles & labels

plt.plot(x,y)

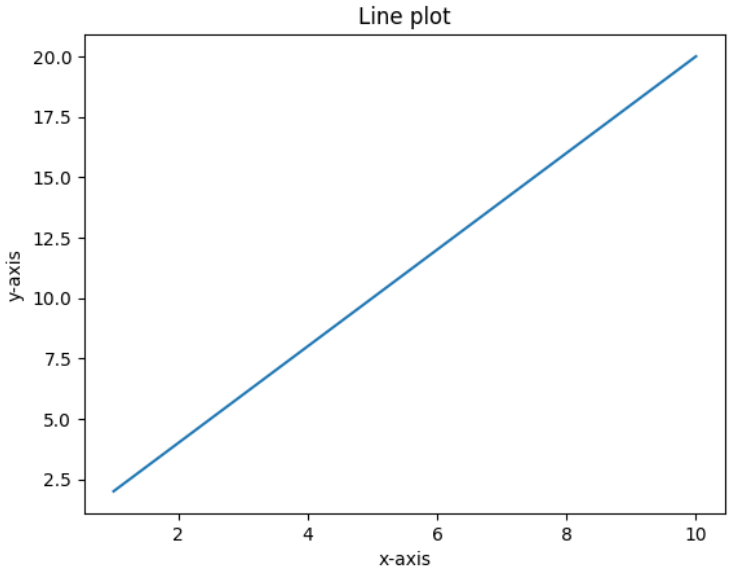
plt.title("Line plot")

plt.xlabel("x-axis")

plt.ylabel("y-axis")

plt.show()

O/P:

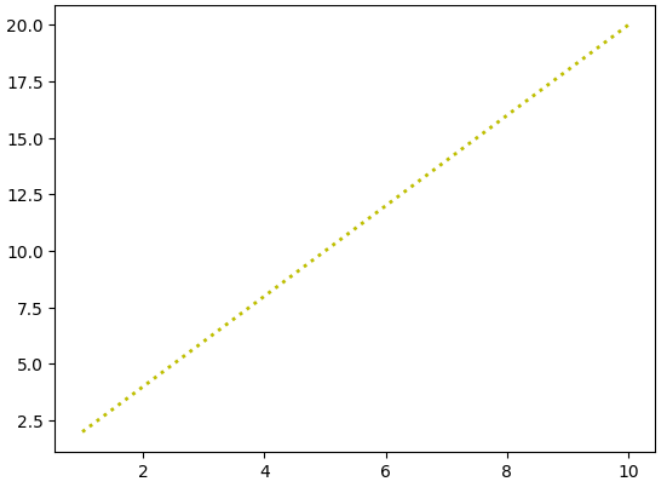


LINE PLOT: Changing line aesthetics

plt.plot(x,y,color='y',linestyle=':',linewidth=2)

plt.show()

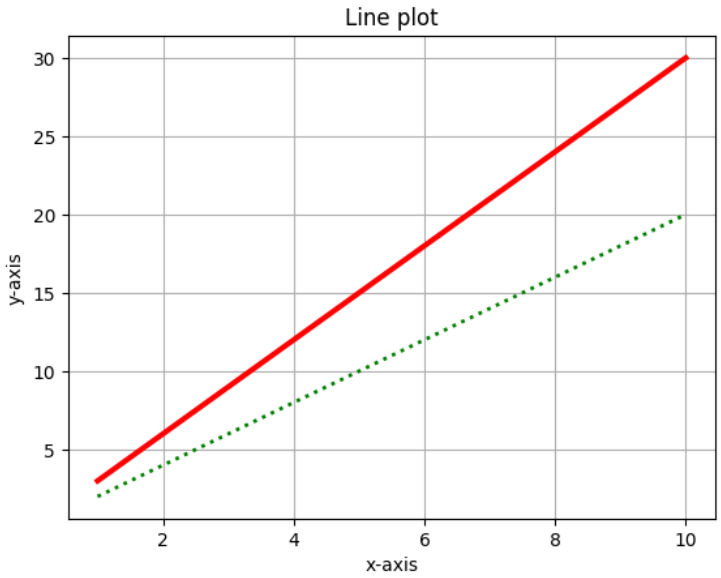
O/P:



LINE PLOT: Adding two lines in the same plot

```
x=np.arange(1,11)
y1=2*x
y2=3*x
plt.plot(x,y1,color='g',linestyle=':',linewidth=2)
plt.plot(x,y2,color='r',linestyle='-',linewidth=3)
plt.title("Line plot")
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.grid(True)
plt.show()
```

O/P:

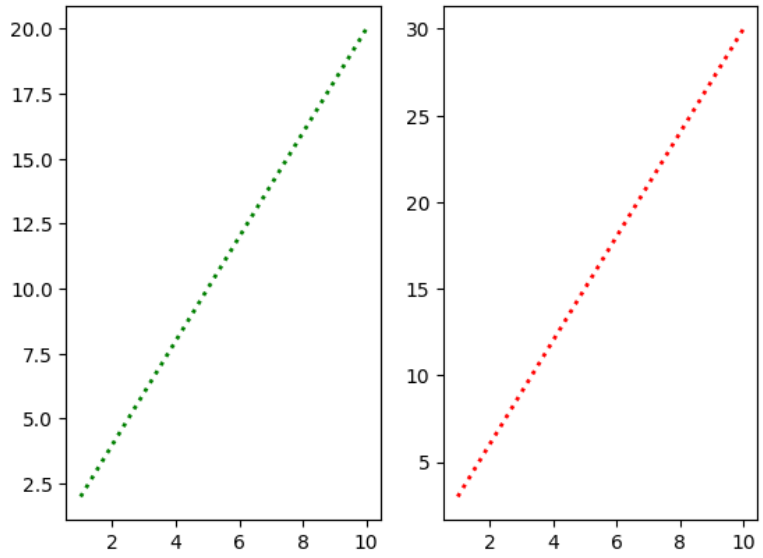


LINE PLOT: Adding sub-plots

```
x=np.arange(1,11)
```

```
y1=2*x
y2=3*x
plt.subplot(1,2,1)
plt.plot(x,y1,color='g',linestyle=':',linewidth=2)
plt.subplot(1,2,2)
plt.plot(x,y2,color='r',linestyle=':',linewidth=2)
plt.show()
```

O/P:

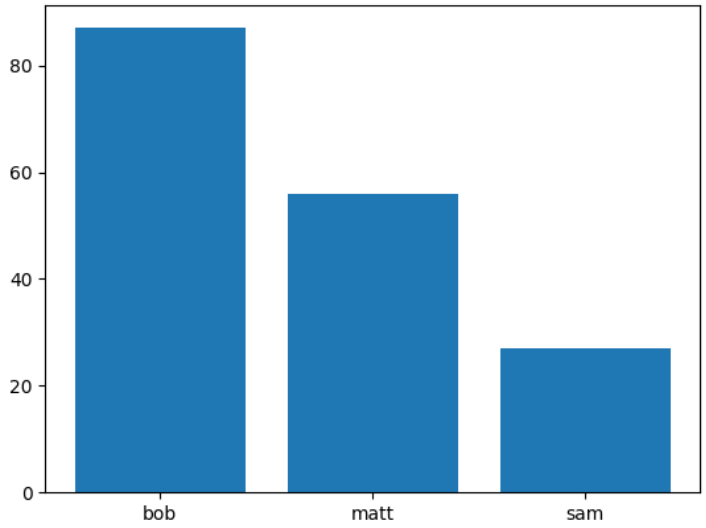


BAR PLOT:

```
student = {"bob":87,"matt":56,"sam":27}
names = list(student.keys())
values = list(student.values())
import matplotlib.pyplot as plt
plt.bar(names,values)
```

plt.show()

O/P:



ADDING TITLE AND LABLES:

plt.bar(names,values)

plt.title("Bar Plot")

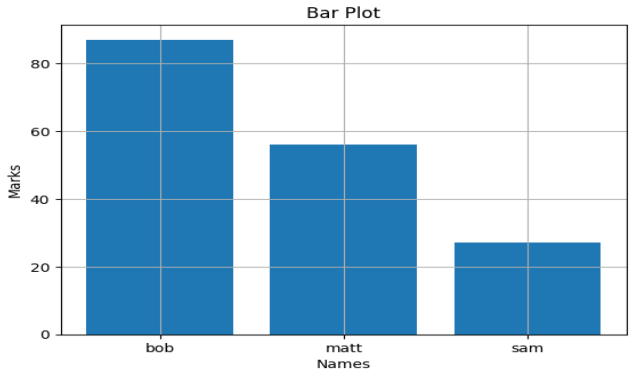
plt.xlabel("Names")

plt.ylabel("Marks")

plt.grid(True)

plt.show()

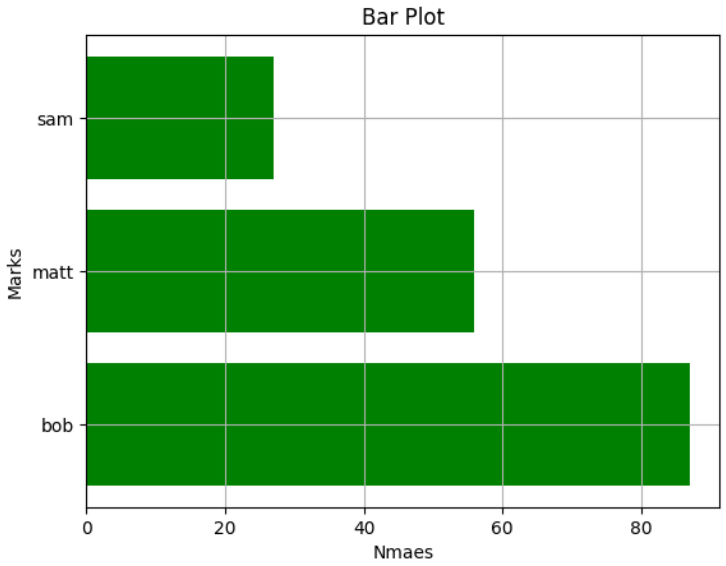
O/P:



HORIZONTAL BAR PLOT:

```
plt.barh(names,values,color='g')
plt.title("Bar Plot")
plt.xlabel("Nmaes")
plt.ylabel("Marks")
plt.grid(True)
plt.show()
```

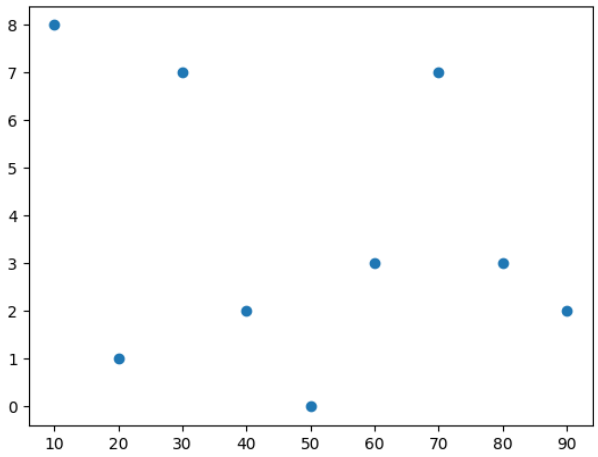
O/P:



SCATTER PLOT:

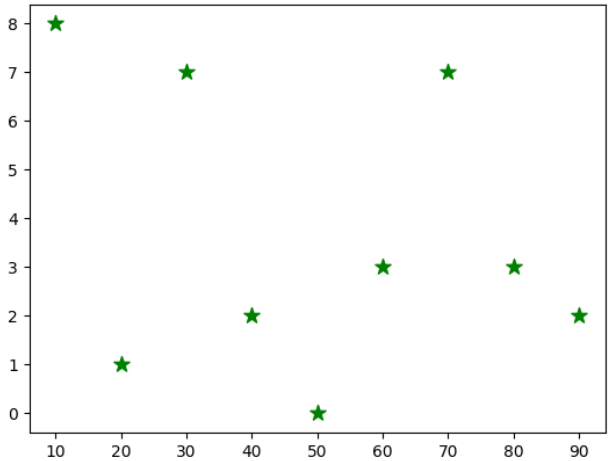
```
x=[10,20,30,40,50,60,70,80,90]
a=[8,1,7,2,0,3,7,3,2]
plt.scatter(x,a)
plt.show()
```

O/P:




```
x=[10,20,30,40,50,60,70,80,90]
a=[8,1,7,2,0,3,7,3,2]
plt.scatter(x,a,marker="*",c="g",s=100)
plt.show()
```

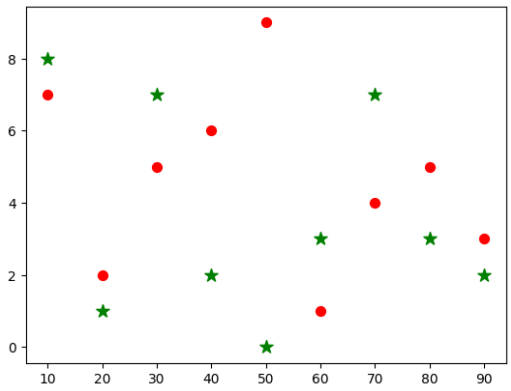
O/P:




```
x=[10,20,30,40,50,60,70,80,90]
a=[8,1,7,2,0,3,7,3,2]
b=[7,2,5,6,9,1,4,5,3]
plt.scatter(x,a,marker="*",c="g",s=100)
plt.scatter(x,b,marker=".",c="r",s=200)
```

plt.show()

O/P:



x=[10,20,30,40,50,60,70,80,90]

a=[8,1,7,2,0,3,7,3,2]

b=[7,2,5,6,9,1,4,5,3]

plt.subplot(1,2,1)

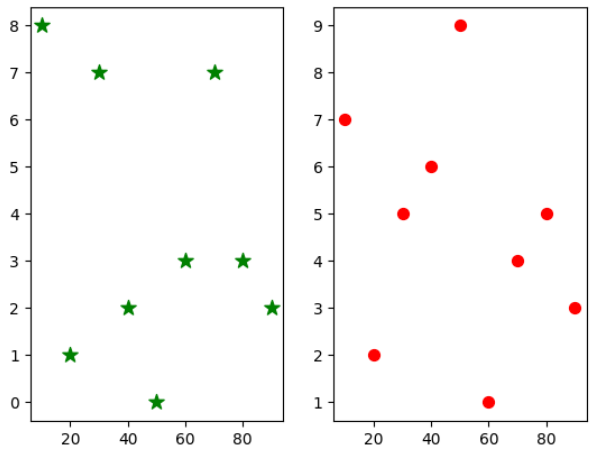
plt.scatter(x,a,marker="*",c="g",s=100)

plt.subplot(1,2,2)

plt.scatter(x,b,marker=".",c="r",s=200)

plt.show()

O/P:

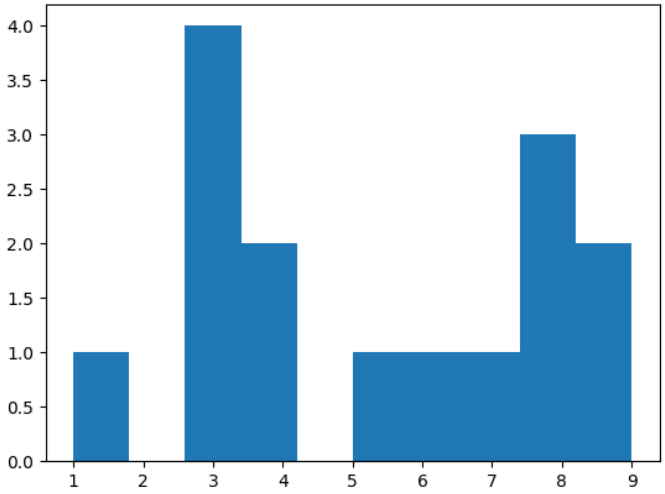


data = [1,3,3,3,3,9,9,5,4,4,8,8,8,6,7]

plt.hist(data)

plt.show()

O/P:

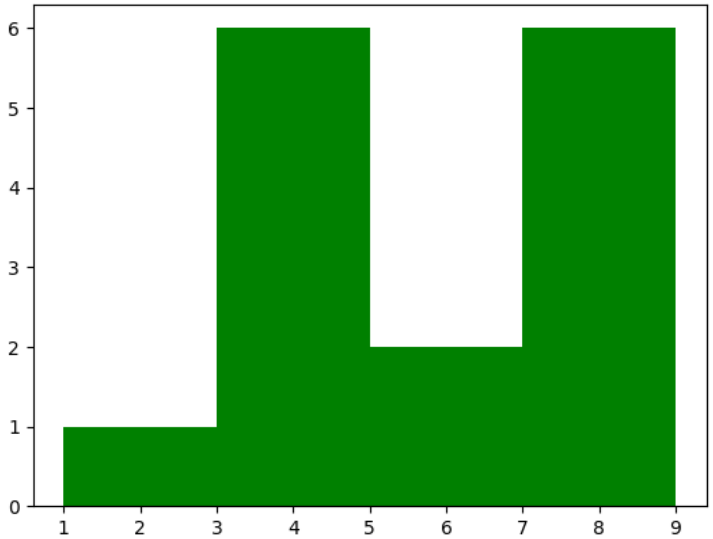


#CHANGING AESTHETICS

plt.hist(data,color="g",bins=4)

plt.show()

O/P:



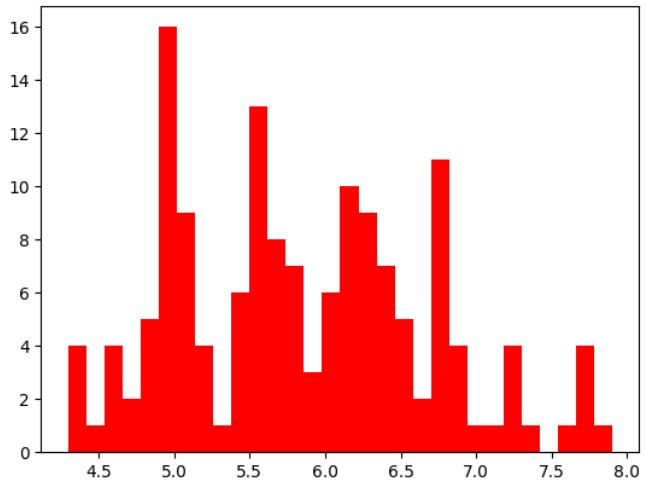
```
iris=pd.read_csv('Iris (1).csv')
iris.head()
```

O/P:

Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species	
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

```
plt.hist(iris['SepalLengthCm'],bins=30,color="r")
plt.show()
```

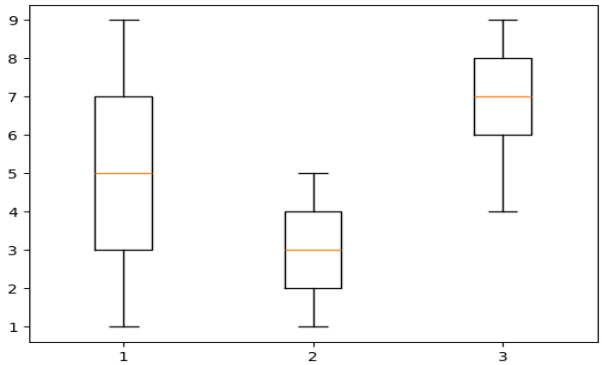
O/P:



#BOXPLOT

```
one = [1,2,3,4,5,6,7,8,9]
two = [1,2,3,4,5,4,3,2,1]
three = [6,7,8,9,8,7,6,5,4]
data = list([one,two,three])
plt.boxplot(data)
plt.show()
```

O/P:

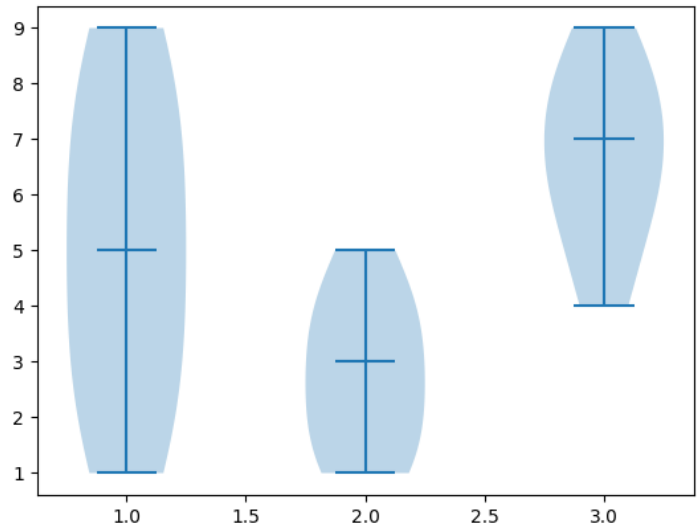


VIOLIN-PLOT:

#Creating data

```
import matplotlib.pyplot as plt
one = [1,2,3,4,5,6,7,8,9,]
two = [1,2,3,4,5,4,3,2,1]
three = [6,7,8,9,8,7,6,5,4]
data = list([one,two,three])
#Making plot
plt.violinplot(data,showmedians=True)
plt.show()
```

O/P:



PIE-CHART:

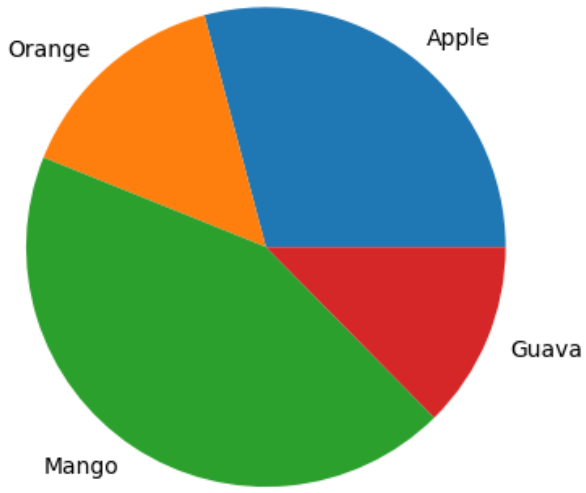
#Creating data

```
fruit = ['Apple','Orange','Mango','Guava']
quantity = [67,34,100,29]
```

#Making plot

```
plt.pie(quantity,labels=fruit)
plt.show()
```

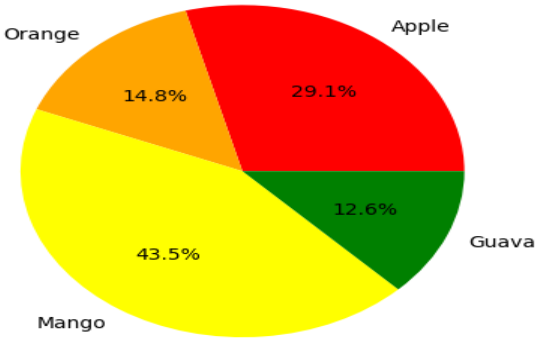
O/P:



CHANGING COLORS TO PIE-CHART:

```
plt.pie(quantity,labels=fruit,autopct='%0.1f%%',colors=['red','orange','yellow','green'])
plt.show()
```

O/P:



DOUGHNUT CHART:

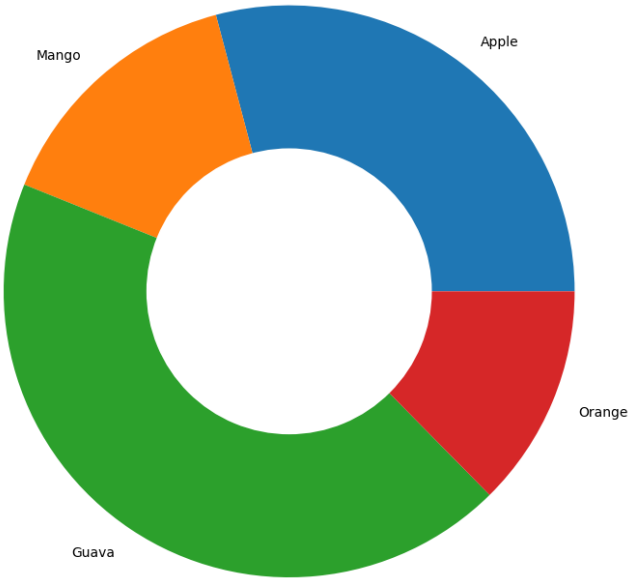
#Creating data

```
fruit =['Apple','Mango','Guava','Orange']
quantity =[67,34,100,29]
```

#Making plot

```
plt.pie(quantity,labels=fruit,radius=2)
plt.pie([1],colors=['w'],radius=1)
plt.show()
```

O/P:



SEABORN LINE PLOT:

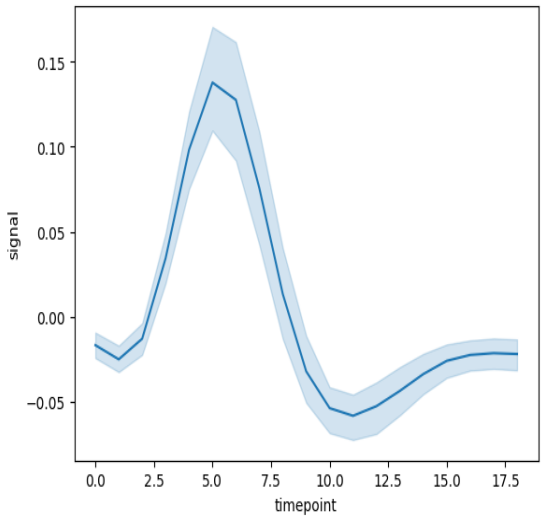
Seaborn is a python data visualization library based on matplotlib it provides a high-level interface for drawing attractive and informative statistical graphics.

```
import seaborn as sns
from matplotlib import pyplot as plt
fmri = sns.load_dataset("fmri")
fmri.head()
```

	subject	timepoint	event	region	signal
0	s13	18	stim	parietal	-0.017552
1	s5	14	stim	parietal	-0.080883
2	s12	18	stim	parietal	-0.081033
3	s11	18	stim	parietal	-0.046134
4	s10	18	stim	parietal	-0.037970

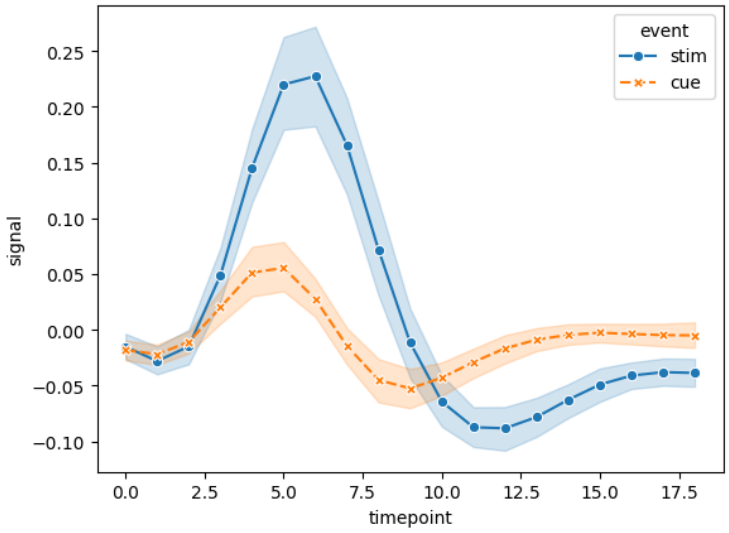
```
sns.lineplot(x="timepoint",y="signal",data=fmri)
plt.show()
```

O/P:




```
sns.lineplot(x="timepoint",y="signal",
             hue="event", style="event",
             markers=True, data=fmri)
plt.show()
```

O/P:



#seaborn

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

```
import matplotlib.pyplot as plt
import seaborn as sns
#visualising statistical relationships
ds = sns.load_dataset('tips')
ds.head()
```

O/P:

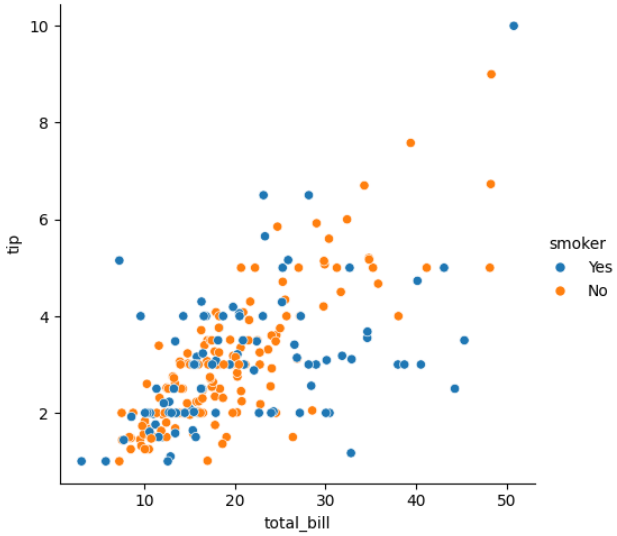
	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

```
ds.shape
```

O/P:

```
(244, 7)
#relating variables with scatter plots
sns.relplot(data=ds,x='total_bill',y='tip',hue='smoker')
```

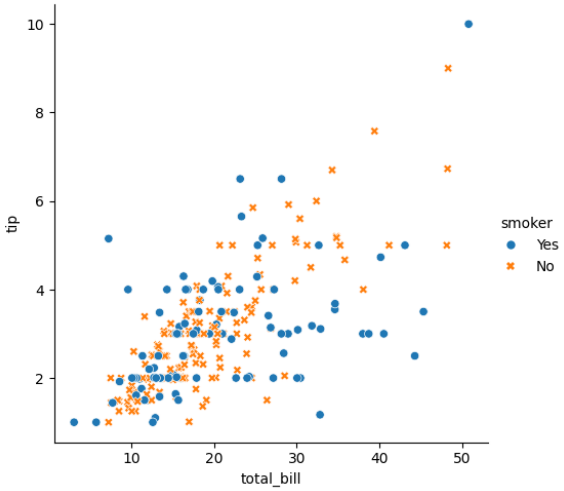
O/P:



#marker

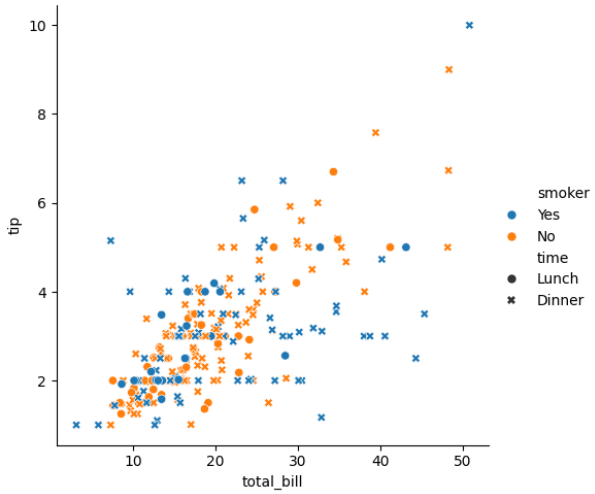
```
sns.relplot(data=ds,x='total_bill',y='tip',hue='smoker',style='smoker')
```

O/P:



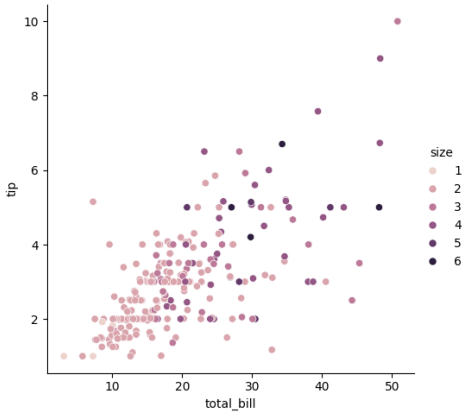

```
sns.relplot(
    data=ds,
    x="total_bill",y="tip",hue="smoker",style="time"
    )
```

O/P:



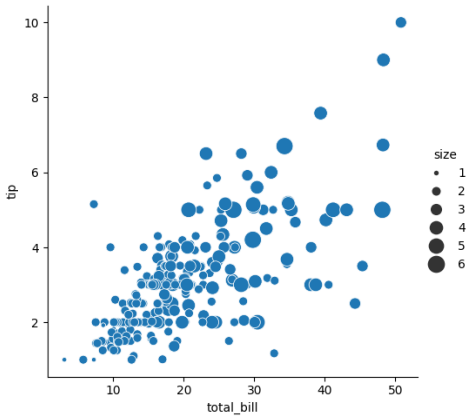

```
sns.relplot(
    data=ds, x="total_bill", y="tip", hue="size",
)
```

O/P:



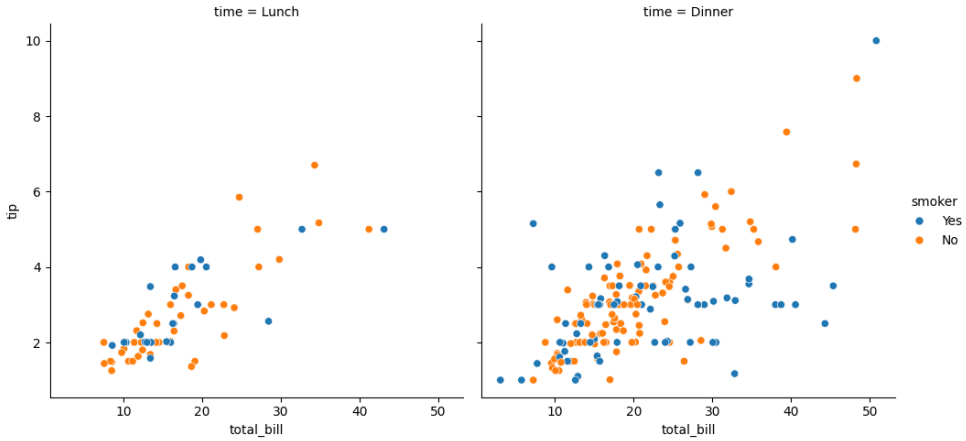

```
sns.relplot(
    data=ds, x="total_bill", y="tip",
    size="size", sizes=(15, 200)
)
```

O/P:




```
sns.relplot(
    data=ds,
    x="total_bill", y="tip", hue="smoker", col="time",
)
```

O/P:




```
import seaborn as sns
fmri = sns.load_dataset('fmri')
fmri.head()
```

O/P:

	subject	timepoint	event	region	signal
0	s13	18	stim	parietal	-0.017552
1	s5	14	stim	parietal	-0.080883
2	s12	18	stim	parietal	-0.081033
3	s11	18	stim	parietal	-0.046134
4	s10	18	stim	parietal	-0.037970

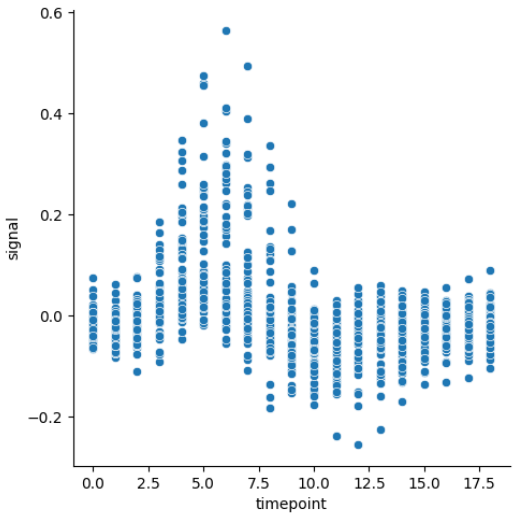
fmri.shape

O/P:

(1064, 5)

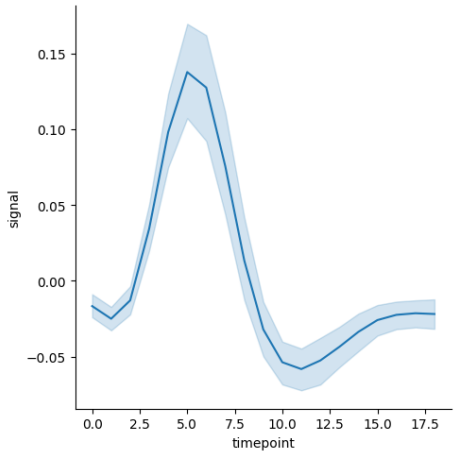
sns.relplot(data=fmri, x='timepoint',y='signal')

O/P:



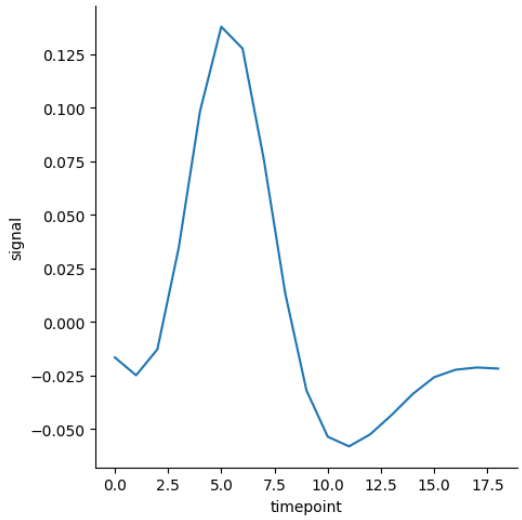
```
sns.relplot(data=fmri, x='timepoint',y='signal',kind='line')
```

O/P:




```
# remove error band
sns.relplot(
    data=fmri,kind='line',
    x='timepoint',y='signal',errorbar=None
)
```

O/P:

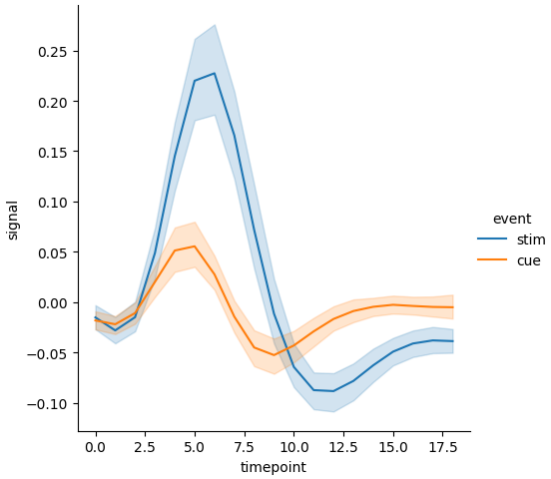


#adding a hue sematic with two level splits

#they plot into two lines and error bands

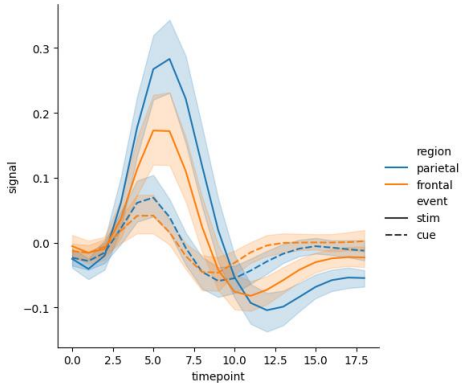
```
sns.relplot(
    data=fmri,kind='line',
    x='timepoint',y='signal',
    hue='event'
)
```

O/P:



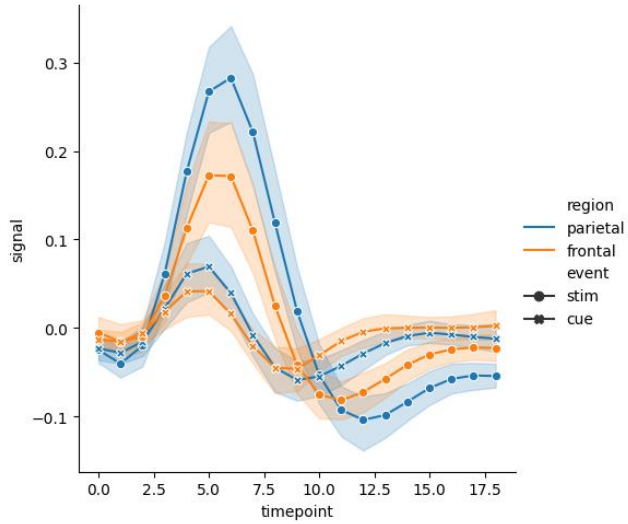

```
sns.relplot(
    data=fmri,kind='line',
    x='timepoint',y='signal',
    hue='region',style='event'
)
```

O/P:




```
sns.relplot(
    data=fmri,kind='line',
    x='timepoint',y='signal',
    hue='region',style='event',
    dashes=False,markers=True
)
```

O/P:




```
import seaborn as sns
tips = sns.load_dataset('tips')
tips.head()
```

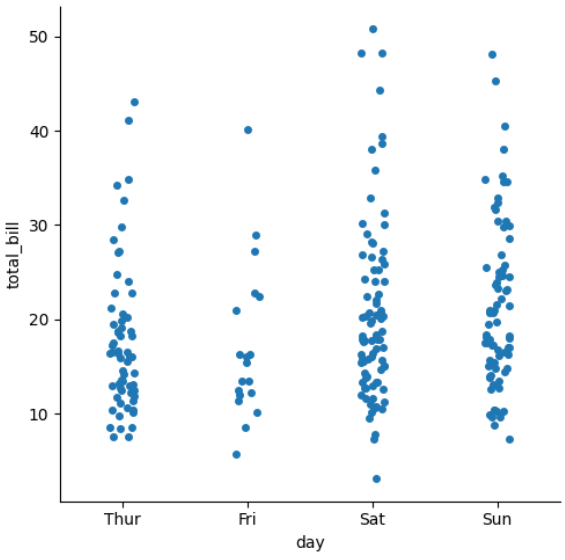
O/P:

	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

#Prevent from overlapping(swarm plot)

sns.catplot(data=tips,x='day',y='total_bill')

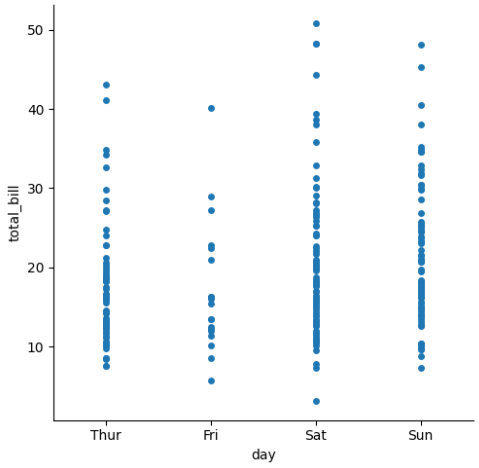
O/P:



#The jitter parameter controls the magnitude of jitter or disables it altogether:

sns.catplot(data=tips,x='day',y='total_bill',jitter=False)

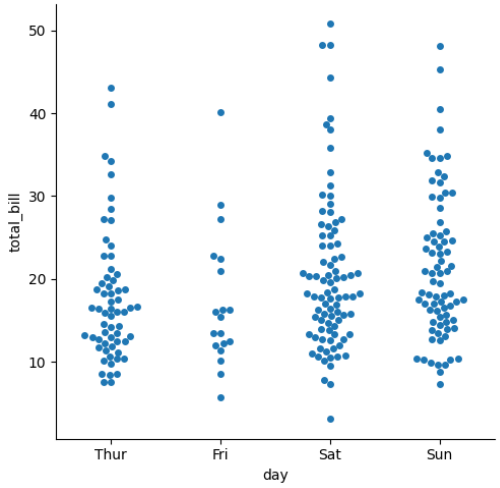
O/P:



#prevent from overlapping(swarm plot)

sns.catplot(data=tips,x='day',y='total_bill',kind='swarm')

O/P:



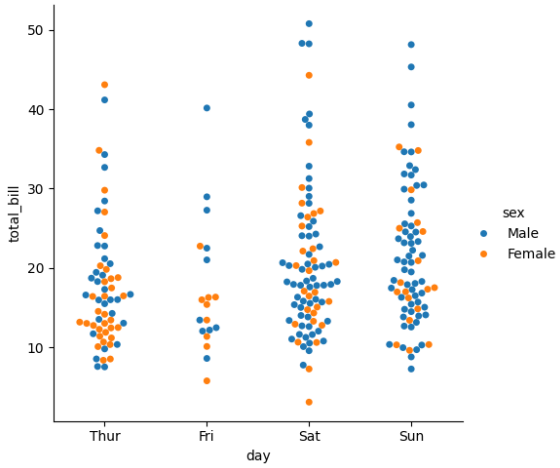
#add the hue semantic

import seaborn as sns

tips = sns.load_dataset('tips')

sns.catplot(data=tips,x='day',y='total_bill',hue='sex',kind='swarm')

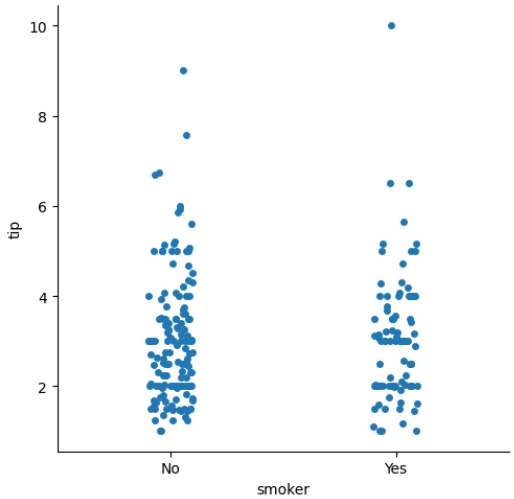
O/P:



#order parameter - to display multiple categorical plot in the same figure

sns.catplot(data=tips,x='smoker',y='tip',order=['No','Yes'])

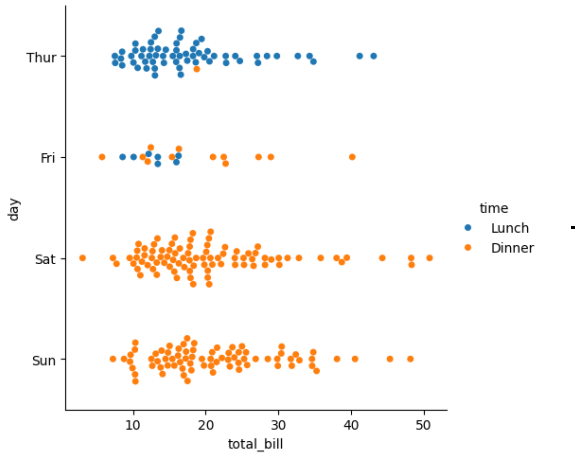
O/P:



#categorical plot on vertical axis

sns.catplot(data=tips,x='total_bill',y='day',hue='time',kind='swarm')

O/P:

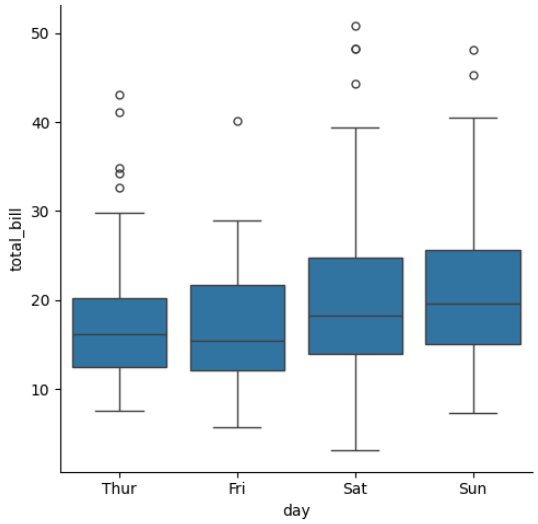


#comparing distribution

#box plots

sns.catplot(data=tips,x='day',y='total_bill',kind='box')

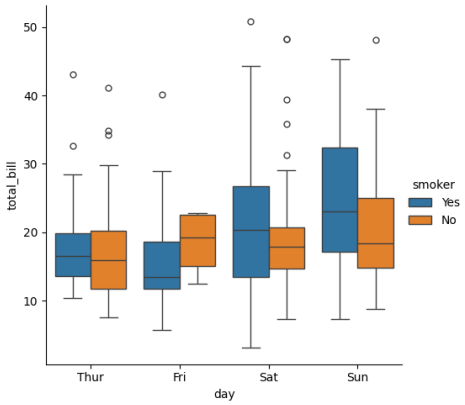
O/P:



#adding hue sematic

sns.catplot(data=tips,x='day',y='total_bill',hue='smoker',kind='box'
)

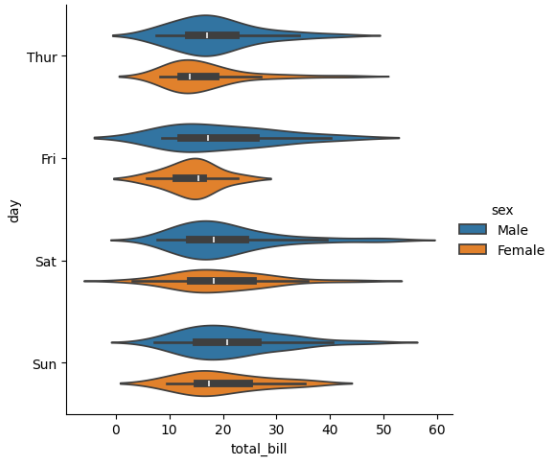
O/P:



#violin plot

sns.catplot(data=tips,x='total_bill',y='day',hue='sex',kind='violin')

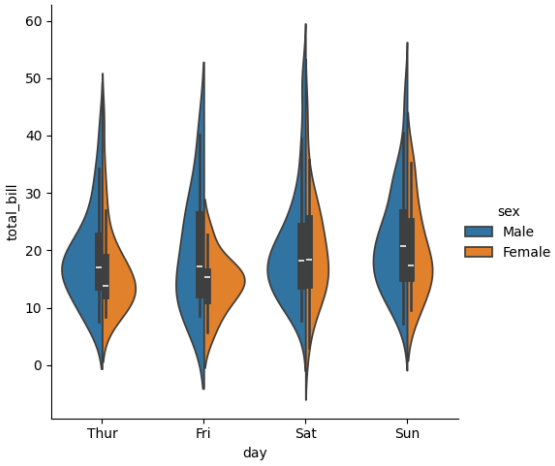
O/P:



#split in the violin plot

sns.catplot(data=tips,x='day',y='total_bill',hue='sex',kind='violin',split=True)

O/P:



BY

G.DIVYASREE.