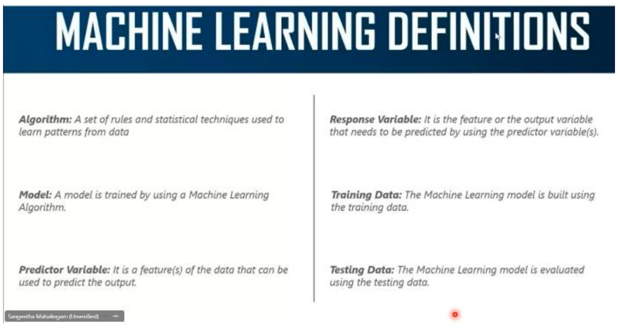
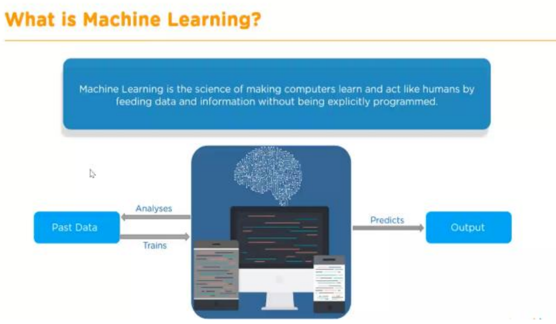
Report on SkillMatch Resume Matcher and Skill Recommender

1.What is Machine Learning (ML)?

Machine Learning is a crucial component of Artificial Intelligence.

- **Core Definition:** Machine learning is a **subset of Artificial Intelligence (AI)** which provides machines the ability to **learn automatically & improve from experience** without being explicitly programmed.
- **Alternative Definition:** Machine Learning is the science of making computers **learn and act like humans** by feeding data and information without being explicitly programmed.



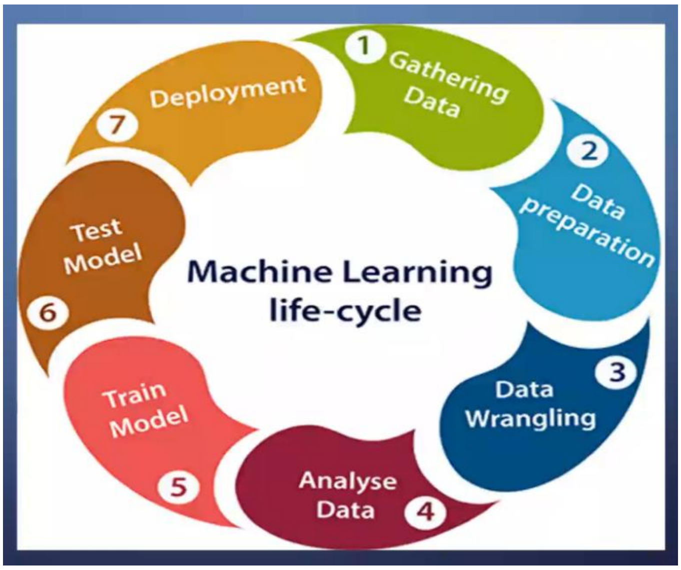


- **The Learning Rule:** "A computer program is said to learn from **experience E** with respect to some class of **tasks T** and **performance measure P** if its performance at tasks in T, as measured by P, **improves with experience E**."
- **History: Arthur Samuel** first coined the term Machine Learning in the year **1959**.
- **Basic Flow:** Data $\rightarrow$ Training the Machine $\rightarrow$ Building a Model $\rightarrow$ Predicting Outcome.
    - The machine takes **Past Data**, **Analyses** it, and **Trains** on it to **Predict** an **Output**.

2. The Machine Learning Process and Life-Cycle

The Machine Learning process involves building a **Predictive model** that can be used to find a **solution for a Problem Statement**.

The 7-Step Machine Learning Life-Cycle



The process is cyclical and involves:

1. **Gathering Data**
2. **Data preparation**
3. **Data Wrangling**
4. **Analyse Data**
5. **Train Model**
6. **Test Model**
7. **Deployment**

Alternative ML Process Steps

Another view of the process emphasizes model development:

- Define Objective
- Data Gathering
- Preparing Data
- Data Exploration
- Building a Model
- Model Evaluation
- Predictions

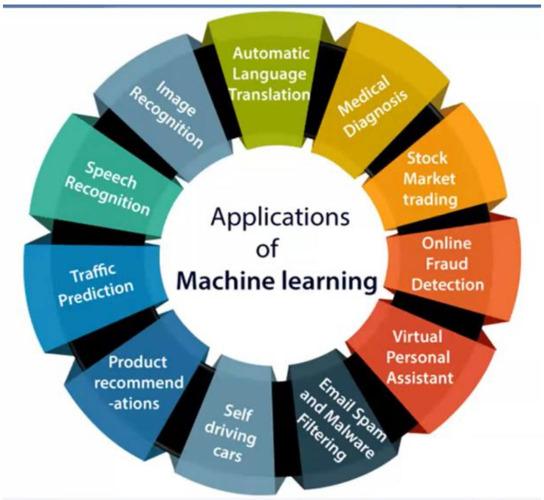
---

3. Real-World Applications of Machine Learning

ML is widely applied across various domains, including:

- Automatic Language Translation
- Medical Diagnosis
- Stock Market trading
- Online Fraud Detection
- Virtual Personal Assistant
- Email Spam and Malware Filtering
- Self driving cars

- Product recommendations
- Traffic Prediction
- Speech Recognition
- Image Recognition
- Face Recognition
- Siri and Cortana
- Healthcare Industry
- Weather Forecasting
- Produce a Web Series (e.g., Netflix recommendations)



Certainly! I can expand section 4 to include **Matplotlib** and **Seaborn**, which are essential libraries for data visualization in the ML process, often used in the **Analyze Data** and **Test Model** phases.

Here is the updated and detailed report.

🤓 Detailed Report: Machine Learning and Data Science Fundamentals

This report is compiled from the provided instructional slides and expanded with essential information about core Python data visualization libraries.

---

1. What is Machine Learning (ML)?

Machine Learning is a crucial component of Artificial Intelligence.

- **Core Definition:** Machine learning is a **subset of Artificial Intelligence (AI)** which provides machines the ability to **learn automatically & improve from experience** without being explicitly programmed.
- **Alternative Definition:** Machine Learning is the science of making computers **learn and act like humans** by feeding data and information without being explicitly programmed.
- **The Learning Rule:** "A computer program is said to learn from **experience E** with respect to some class of **tasks T** and **performance measure P** if its performance at tasks in T, as measured by P, **improves with experience E**."

- **History: Arthur Samuel** first coined the term Machine Learning in the year **1959**.
- **Basic Flow:** Data $\rightarrow$ Training the Machine $\rightarrow$ Building a Model $\rightarrow$ Predicting Outcome.
    - The machine takes **Past Data**, **Analyses** it, and **Trains** on it to **Predict** an **Output**.

---

2. The Machine Learning Process and Life-Cycle

The Machine Learning process involves building a **Predictive model** that can be used to find a **solution for a Problem Statement**.

The 7-Step Machine Learning Life-Cycle

The process is cyclical and involves:

1. **Gathering Data**
2. **Data preparation**
3. **Data Wrangling**
4. **Analyse Data**
5. **Train Model**
6. **Test Model**
7. **Deployment**

Alternative ML Process Steps

Another view of the process emphasizes model development:

- Define Objective
- Data Gathering
- Preparing Data
- Data Exploration
- Building a Model
- Model Evaluation
- Predictions

---

3. Real-World Applications of Machine Learning

ML is widely applied across various domains, including:

- Automatic Language Translation
- Medical Diagnosis
- Stock Market trading
- Online Fraud Detection
- Virtual Personal Assistant
- Email Spam and Malware Filtering
- Self driving cars
- Product recommendations
- Traffic Prediction
- Speech Recognition
- Image Recognition
- Face Recognition
- Siri and Cortana
- Healthcare Industry
- Weather Forecasting
- Produce a Web Series (e.g., Netflix recommendations)

---

4. Key Python Libraries for Data Science

The Python ecosystem relies on several libraries for data handling and analysis in ML. **NumPy** and **Pandas** handle the data itself, while **Matplotlib** and **Seaborn** handle its visual representation.

Library	Full Meaning	Core Functionality	Data Structures / Focus
NumPy	**Numerical Python**	The core library for **numeric and scientific computing**.	**Multi-dimensional array objects** and routines for processing them.
Pandas	**Panel Data**	The core library for **data manipulation and data analysis**.	**Single and multi-dimensional data-structures** (like DataFrames).

Library	Full Meaning	Core Functionality	Data Structures / Focus
Matplotlib	Matplotlib	A **low-level** library for creating **static, animated, and interactive visualizations**.	Used for **basic plotting** (line charts, scatter plots, bar charts) and offers **high customizability**.
Seaborn	Seaborn	A **high-level** library **built on top of Matplotlib** for drawing **attractive and informative statistical graphics**.	Simplifies creating complex plots (like heatmaps and violin plots) with **beautiful default themes** and statistical functions.

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

```
array([10, 20, 30, 40])
```

```
import numpy as np
n1=np.full((3,3),10)
n1
```

```
array([[10, 10, 10],
       [10, 10, 10],
       [10, 10, 10]])
```

```
Start coding or generate with AI.
```

```
import numpy as np
# Corrected the function name from 'np.zeroes' to 'np.zeros'
n1 = np.zeros((1, 2))
print(n1)
```

```
[[0. 0.]]
```

```
import numpy as np
# Corrected the function name from 'np.zeroes' to 'np.zeros'
n1 = np.arange(1,10)
n1
```

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

```
import numpy as np
# Corrected the function name from 'np.zeroes' to 'np.zeros'
n1 = np.arange(1,100,3)
n1
```

```
array([ 1,  4,  7, 10, 13, 16, 19, 22, 25, 28, 31, 34, 37, 40, 43, 46, 49,
       52, 55, 58, 61, 64, 67, 70, 73, 76, 79, 82, 85, 88, 91, 94, 97])
```

```
import numpy as np
# Corrected the function name from 'np.zeroes' to 'np.zeros'
n1 = np.random.randint(1,100,5)
n1
```

```
array([73, 48, 21, 44, 91])
```

Double-click (or enter) to edit

```
import numpy as np
n1= np.array([[10,200,4,30],[50,6,20,70]])
n1.shape
```

```
(2, 4)
```

```
import numpy as np
# Corrected the function name from 'np.zeroes' to 'np.zeros'
n1 = np.arange(1,10)
n1
```

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

```
import numpy as np
n1= np.array([10,20,30,40])
n2= np.array([50,60,70,90])
np.vstack((n1,n2))
```

```
array([[10, 20, 30, 40],
       [50, 60, 70, 90]])
```

```
import numpy as np
n1= np.array([10,20,30,40])
n2= np.array([50,60,70,90])
np.hstack((n1,n2))
```

```
array([10, 20, 30, 40, 50, 60, 70, 90])
```

```
import numpy as np
n1= np.array([10,20,30,40])
n2= np.array([50,60,70,90])
np.column_stack((n1,n2))
```

```
array([[10, 50],
       [20, 60],
       [30, 70],
       [40, 90]])
```

```
import numpy as np
n1= np.array([10,20,30,40,50,60])
n2= np.array([50,60,70,90])
```

```
np.intersect1d(n1,n2)
```

```
array([50, 60])
```

```
np.setdiff1d(n1,n2)
```

```
array([10, 20, 30, 40])
```

```
np.setdiff1d(n2,n1)
```

```
array([70, 90])
```

```
import numpy as np
n1= np.array([10,20,30,40])
n2= np.array([50,60,70,90])
np.sum([n1,n2])
```

```
np.int64(370)
```

```
import numpy as np
n1= np.array([10,20,30,40])
n2= np.array([50,60,70,90])
np.sum([n1,n2],axis=0)
```

```
array([ 60,  80, 100, 130])
```

```
import numpy as np
n1= np.array([10,20,30,40])
n2= np.array([50,60,70,90])
np.sum([n1,n2],axis=1)
```

```
array([100, 270])
```

```
import numpy as np
n1= np.array([10,20,30,40])
n1=n1*2
n1
```

```
array([20, 40, 60, 80])
```

```
import numpy as np
n1= np.array([10,20,30,40])
n1=n1+2
n1
```

```
array([12, 22, 32, 42])
```

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

```
n1=n1/2
n1
```

```
array([ 5., 10., 15., 20.])
```

```
import numpy as np
n1= np.array([10,20,30,40])
n1=n1-2
n1
```

```
array([ 8, 18, 28, 38])
```

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

```
np.float64(25.0)
```

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

```
np.float64(11.180339887498949)
```

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

```
np.float64(25.0)
```

```
import numpy as np
# Added a comma after the second list
n1 = np.array([[10, 200, 4, 30], [50, 6, 20, 70], [20, 33, 1, 3]])
n1
```

```
array([[ 10, 200,   4,  30],
       [ 50,   6,  20,  70],
       [ 20,  33,   1,   3]])
```

```
import numpy as np
n1= np.array([[10,200,4,30], [50,6,20,70],[20,33,1,3]])
n1.transpose()
```

```
array([[ 10,  50,  20],
       [200,   6,  33],
       [  4,  20,   1],
       [ 30,  70,   3]])
```

```
import numpy as np
n1= np.array([[10,20,4,30], [50,6,20,70],[20,33,1,3]])
n1
```

```
array([[10, 20,  4, 30],
       [50,  6, 20, 70],
       [20, 33,  1,  3]])
```

```
import numpy as np
n2= np.array([[1,2,3,4], [5,6,7,8],[9,10,11,12]])
n2
```

```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

```
import pandas as pd
s1=pd.Series([1,2,3,4,5])
s1
```

	0
0	1
1	2
2	3
3	4
4	5

dtype: int64

```
import pandas as pd
s1=pd.Series([1,2,3,4,5], index = ['a','b','c','d','e'])
s1
```

	0
a	1
b	2
c	3
d	4
e	5

dtype: int64

```
import pandas as pd
s1=pd.Series({'a':1,'b':2,'c':3,'d':4,'e':5})
s1
```

	0
a	1
b	2
c	3
d	4
e	5

dtype: int64

```
import pandas as pd
s1=pd.Series({'a':1,'b':2,'c':3,'d':4,'e':5},index=['b','c','d','a','e'])
s1
```

	0
b	2
c	3
d	4
a	1
e	5

dtype: int64

```
import pandas as pd
s1=pd.Series({'a':1,'b':2,'c':3,'d':4,'e':5},index=['f','b','c','d','a','m'])
s1
```

	0
f	NaN
b	2.0
c	3.0
d	4.0
a	1.0
m	NaN

dtype: float64

```
import pandas as pd
s1=pd.Series([1,2,3,4,5])
s1[2]
```

```
np.int64(3)
```

```
import pandas as pd
s1=pd.Series([1,2,3,4,5])
s1[:2]
```

	0
0	1
1	2

dtype: int64

```
import pandas as pd
s1=pd.Series([1,2,3,4,5])
s1[-2:]
```

	0
3	4
4	5

dtype: int64

```
import pandas as pd
s1=pd.Series([1,2,3,4,5])
s1+5
```

	0
0	6
1	7
2	8
3	9
4	10

dtype: int64

```
import pandas as pd
s1=pd.Series([1,2,3,4,5])
s2=pd.Series([10,20,30,40,50])
s1+s2
```

	0
0	11
1	22
2	33
3	44
4	55

dtype: int64

```
import pandas as pd
pd.DataFrame({"Name":["hemanth","Siri","Ammu"],"Marks":[80,99,100]})
```

	Name	Marks
0	hemanth	80
1	Siri	99
2	Ammu	100

```
import pandas as pd
ds = pd.read_csv('/Iris.csv')
```

```
ds.head()
```

	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

Next steps: Generate code with ds New interactive sheet

```
ds.head(20)
```

	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

Next steps: Generate code with ds New interactive sheet

```
ds.tail()
```

	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.shape
```

```
(150, 6)
```

```
ds.describe()
```

	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

```
ds.iloc[100:150,0:6]
```

```
ds.loc[10:20,('SepalLengthCm','PetalWidthCm')]
```

	SepalLengthCm	PetalWidthCm
10	5.4	0.2
11	4.8	0.2
12	4.8	0.1
13	4.3	0.1
14	5.8	0.2
15	5.7	0.4
16	5.4	0.4
17	5.1	0.3
18	5.7	0.3
19	5.1	0.3
20	5.4	0.2

100	101	6.3	3.3	6.0	2.5	Iris-virginica
101	102	5.8		5.1	1.9	Iris-virginica
				5.9	2.1	Iris-virginica
103	104	6.3	2.9	5.6	1.8	Iris-virginica
			3.0	5.8	2.2	Iris-virginica
105	106	7.6	3.0	6.6	2.1	Iris-virginica
			2.5	4.5	1.7	Iris-virginica
107	108	7.3	2.9	6.3	1.8	Iris-virginica
			2.5	5.8	1.8	Iris-virginica
109	110	7.2	3.6	6.1	2.5	Iris-virginica
			3.2	5.1	2.0	Iris-virginica
111	112	6.4	2.7	5.3	1.9	Iris-virginica
			3.0	5.5	2.1	Iris-virginica

```
Start coding or generate with AI.
```

```
ds.drop('SepalLengthCm',axis=1)
```

	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

116	117	6.5	3.0	5.5	1.8	Iris-virginica
					2.2	Iris-virginica
					2.3	Iris-virginica
					1.5	Iris-virginica
					2.3	Iris-virginica
					2.0	Iris-virginica
					2.0	Iris-virginica
					1.8	Iris-virginica
					2.1	Iris-virginica
					1.8	Iris-virginica
					1.8	Iris-virginica
					1.8	Iris-virginica
					2.1	Iris-virginica
129	130	7.2	3.0	5.8	1.6	Iris-virginica

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

	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

131	132	7.9	3.8	6.4	2.0	Iris-virginica
132	133	6.4	2.8	5.6	2.2	Iris-virginica
134	135	6.1	2.6	5.6	1.4	Iris-virginica

```
import numpy as np
from matplotlib import pyplot as plt
```

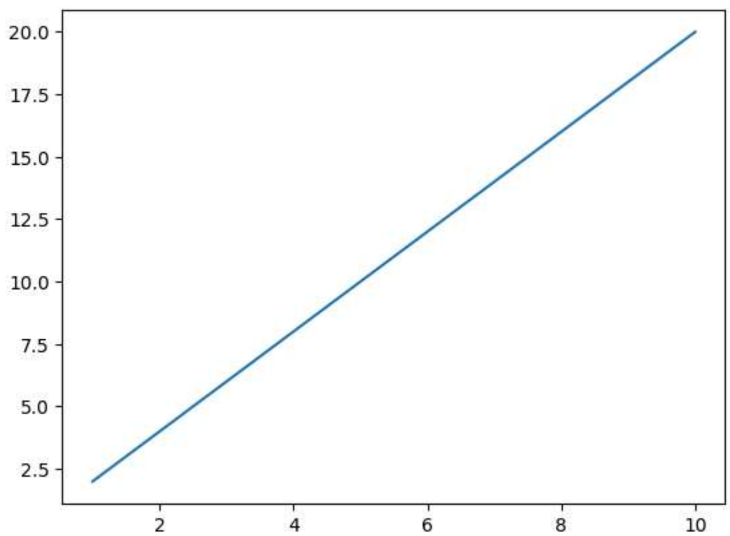
```
x=np.arange(1,11)
x
```

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

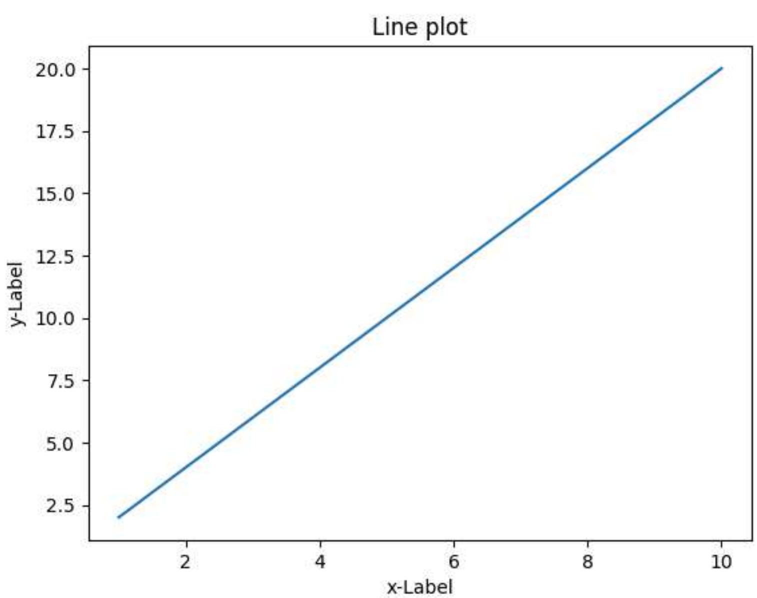
```
y=2*x
y
```

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

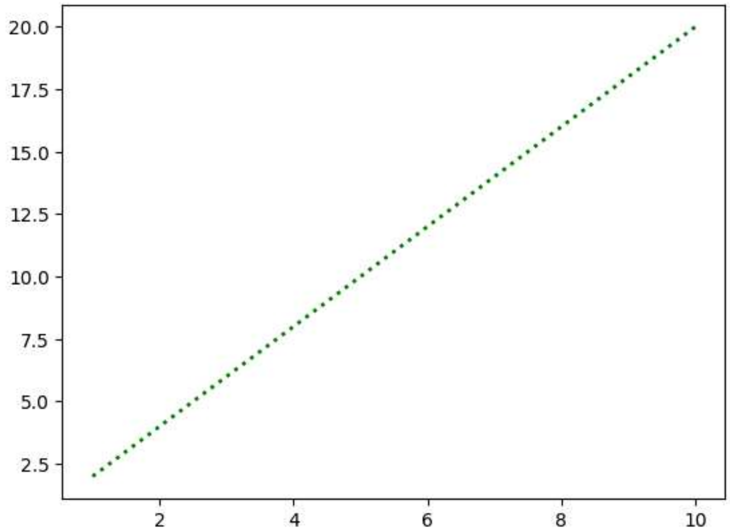
```
plt.plot(x,y)
plt.show()
```




```
plt.plot(x,y)
plt.title("Line plot")
plt.xlabel("x-Label")
plt.ylabel("y-Label")
plt.show()
```



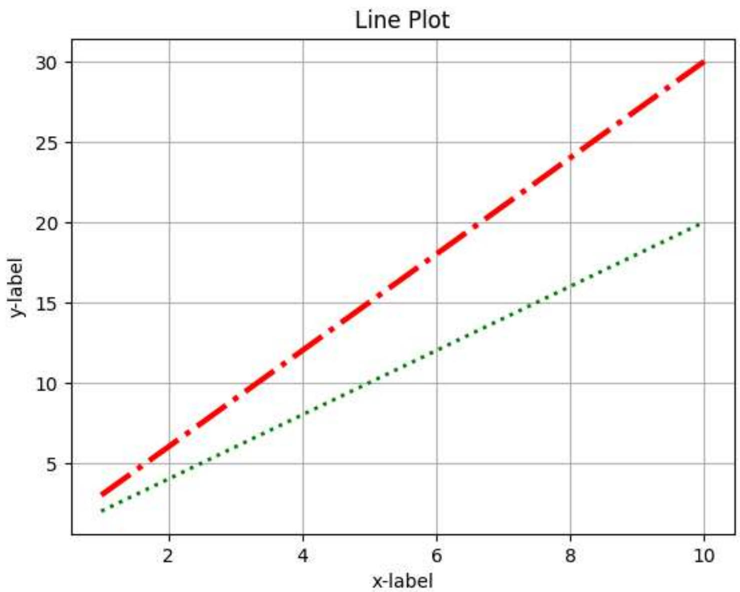

```
plt.plot(x,y,color="g",linestyle=':',linewidth=2)
plt.show()
```



```
x=np.arange(1,11)
y1=2*x
y2=3*x
```

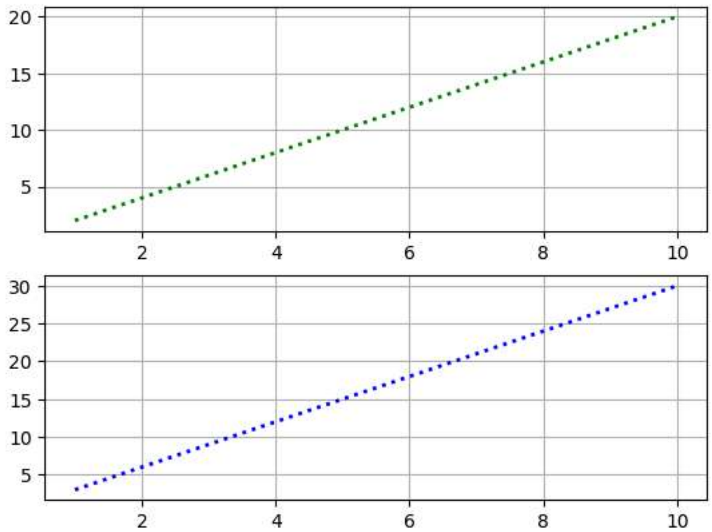
```
plt.plot(x,y1,color='g',linestyle=':',linewidth=2)
plt.plot(x,y2,color='r',linestyle='-.',linewidth=3)
plt.title("Line Plot")
plt.xlabel("x-label")
plt.ylabel("y-label")
plt.grid(True)
plt.show
```

```
matplotlib.pyplot.show
def show(*args, **kwargs) -> None

Display all open figures.

Parameters
----------
block : bool, optional
    Whether to wait for all figures to be closed before returning.
```




```
x=np.arange(1,11)
y1=2*x
y2=3*x

plt.subplot(2,1,1)
plt.plot(x,y1,color='g',linestyle=':',linewidth=2)
```

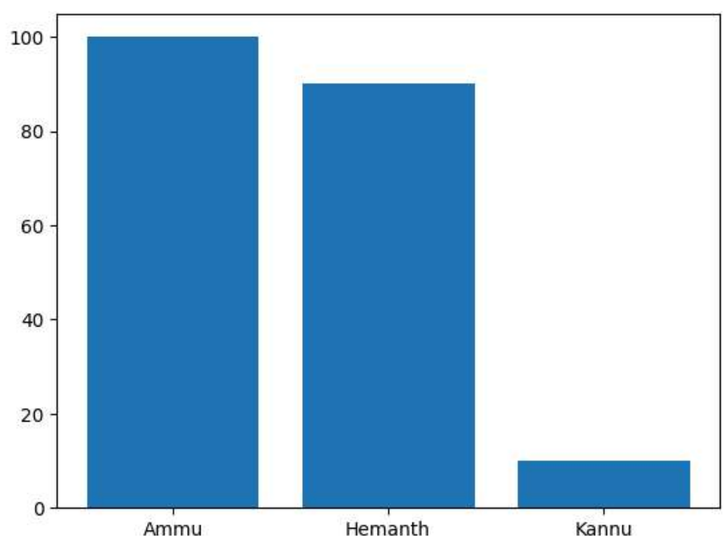
```
plt.grid(True)
plt.subplot(2,1,2)
plt.plot(x,y2,color='b',linestyle=':',linewidth=2)
plt.grid(True)
plt.show()
```



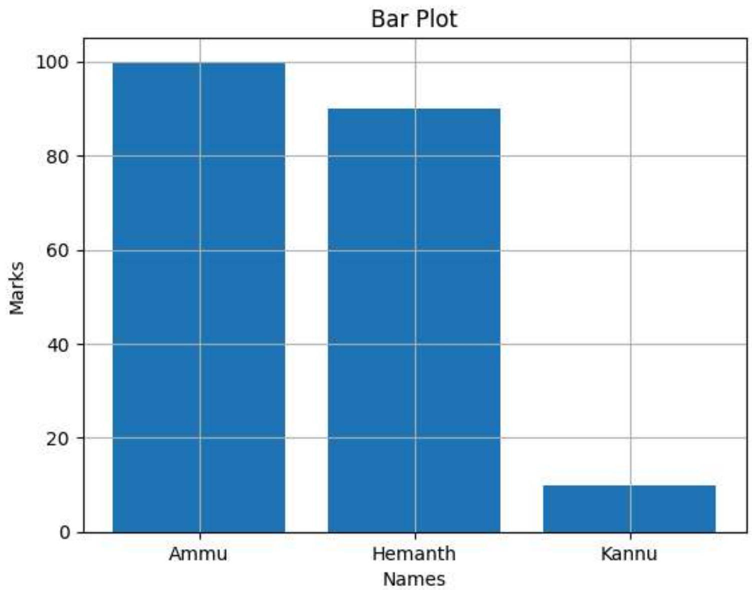

```
student = {"Ammu":100, "Hemanth":90,"Kannu":10}
```

```
names = list(student.keys())
values = list(student.values())
```

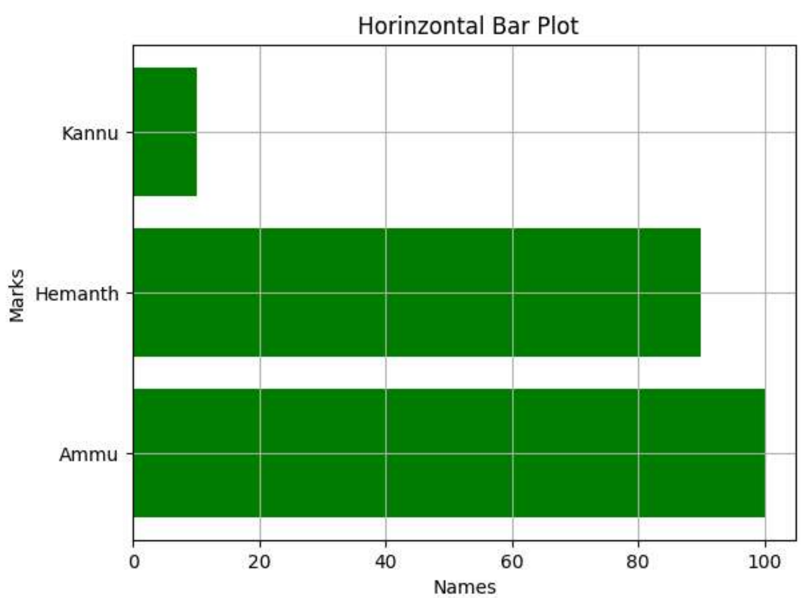
```
plt.bar(names,values)
plt.show()
```



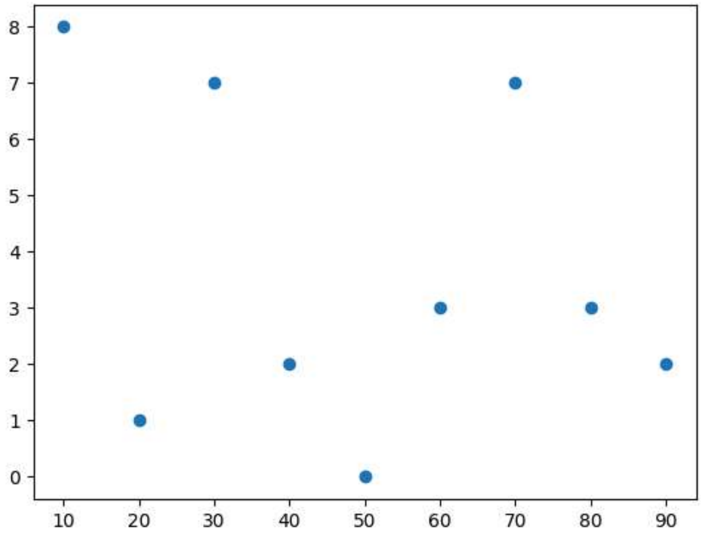

```
plt.bar(names,values)
plt.title("Bar Plot")
plt.xlabel("Names")
plt.ylabel("Marks")
plt.grid(True)
plt.show()
```



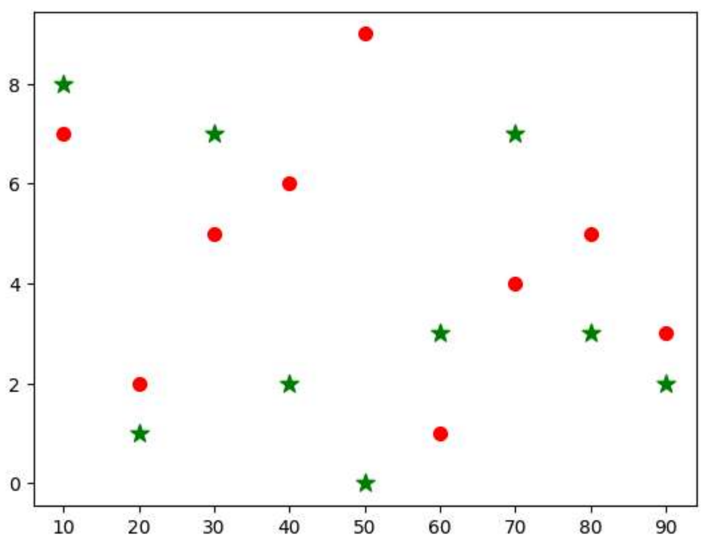

```
plt.barh(names,values,color="g")
plt.title(" Horinzontal Bar Plot")
plt.xlabel("Names")
plt.ylabel("Marks")
plt.grid(True)
plt.show()
```



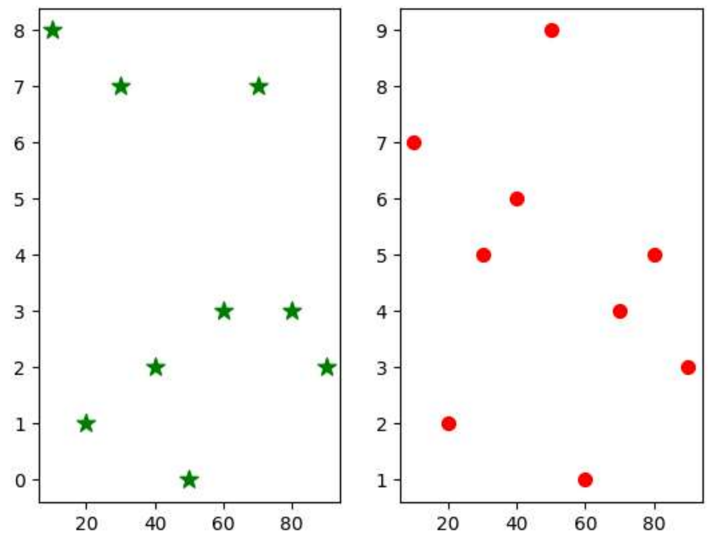

```
from matplotlib import pyplot as plt
x = [10, 20, 30, 40, 50, 60, 70, 80, 90]
a = [8,1,7,2,0,3,7,3,2]
plt.scatter(x,a)
plt.show()
```



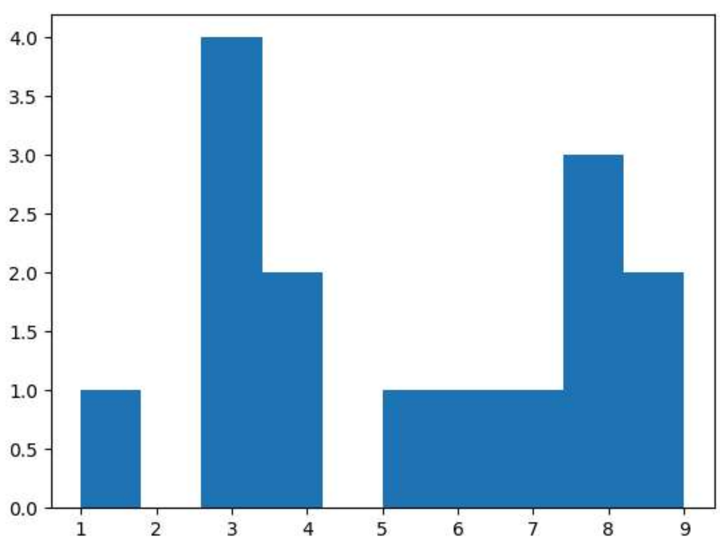

```
x = [10, 20, 30, 40, 50, 60, 70, 80, 90]
a = [8,1,7,2,0,3,7,3,2]
b = [7,2,5,6,9,1,4,5,3]
plt.scatter(x,a,marker="*",c="g",s=100)
plt.scatter(x,b,marker=".",c="r",s=200)
plt.show()
```




```
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()
```

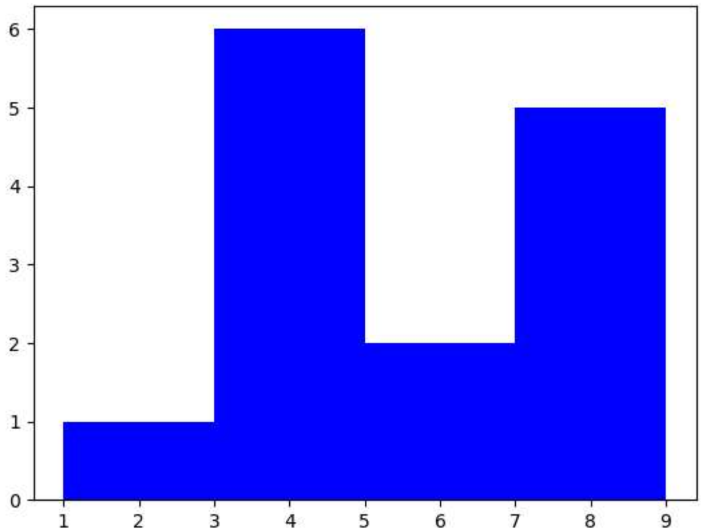



```
from matplotlib import pyplot as plt
data = [1,3,3,3,3,9,9,5,4,4,8,8,8,6,7]
plt.hist(data)
plt.show()
```

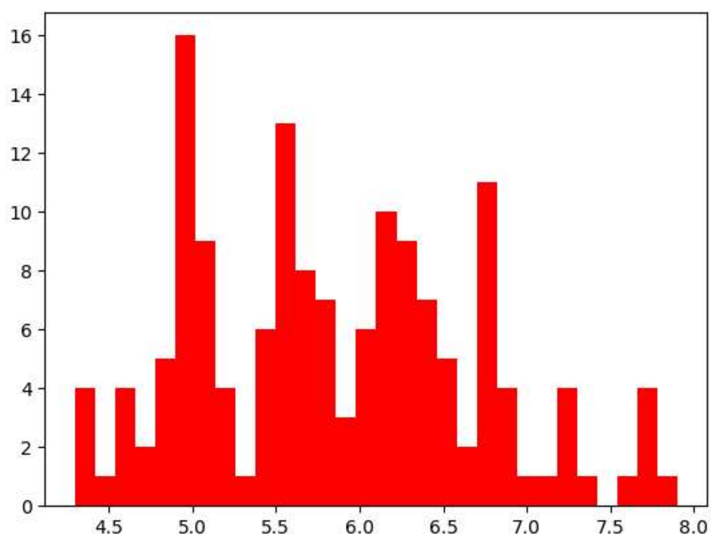


Double-click (or enter) to edit

```
from matplotlib import pyplot as plt
data = [1,3,3,3,3,9,9,5,4,4,8,8,8,6,7]
plt.hist(data, color='b',bins =4)
plt.show()
```

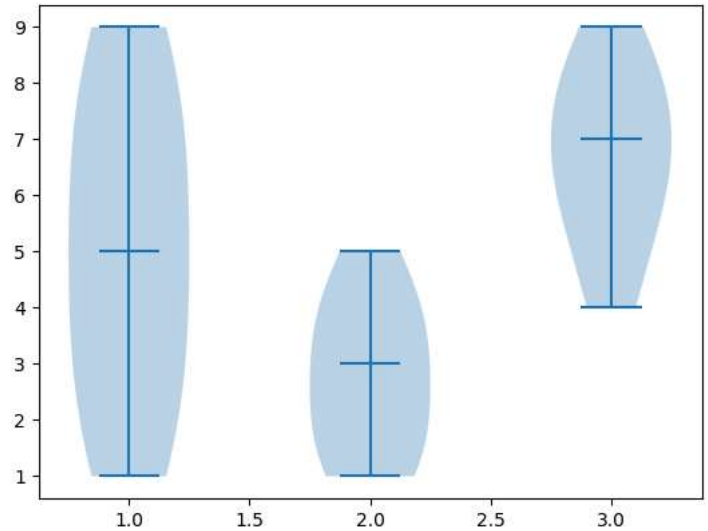



```
import pandas as pd
Iris = pd.read_csv('Iris.csv')
Iris.head()
plt.hist(Iris['SepalLengthCm'],bins=30,color="r")
plt.show()
```

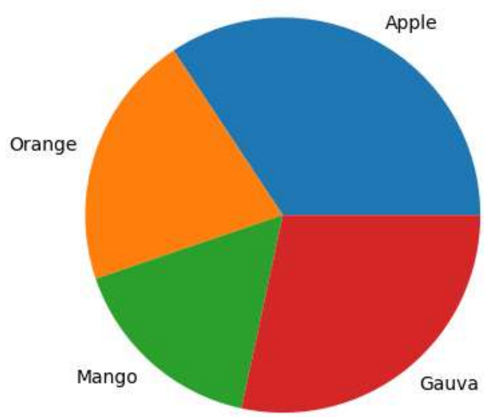



```
one = [1,2,3,4,5,6,7,8,9]
two = [1,2,3,4,5,4,3,2,1]
three = [6,7,8,9,8,7,6,5,4]

data = list([one,two,three])
plt.boxplot(data)
plt.show()
```


```
plt.violinplot(data,showmedians=True)
plt.show()
```




```
from matplotlib import pyplot as plt
fruit = ['Apple','Orange','Mango','Gauva']
quantity =[69,42,33,57]
plt.pie(quantity,labels=fruit)
plt.show()
```

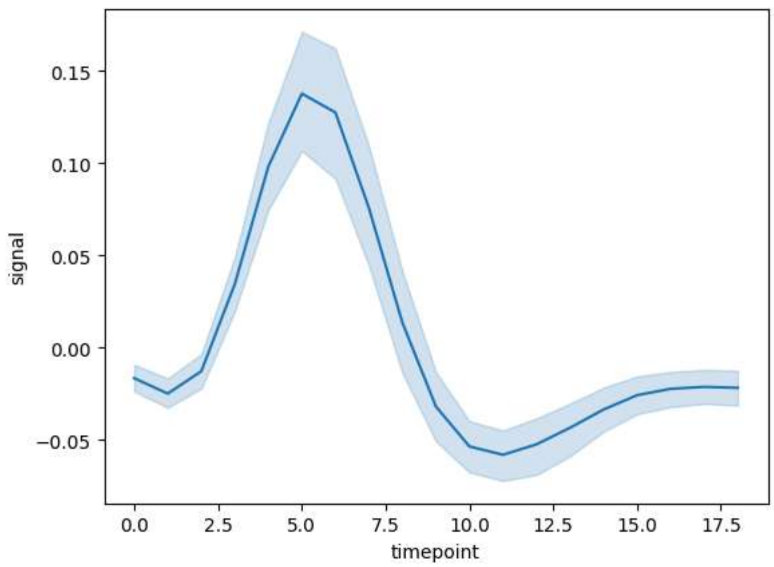



```
from matplotlib import pyplot as plt
fruit = ['Apple','Orange','Mango','Gauva']
quantity =[69,42,33,57]
plt.pie(quantity,labels=fruit,radius=2)
plt.pie([1],colors=['w'],radius=1)
plt.show()
```

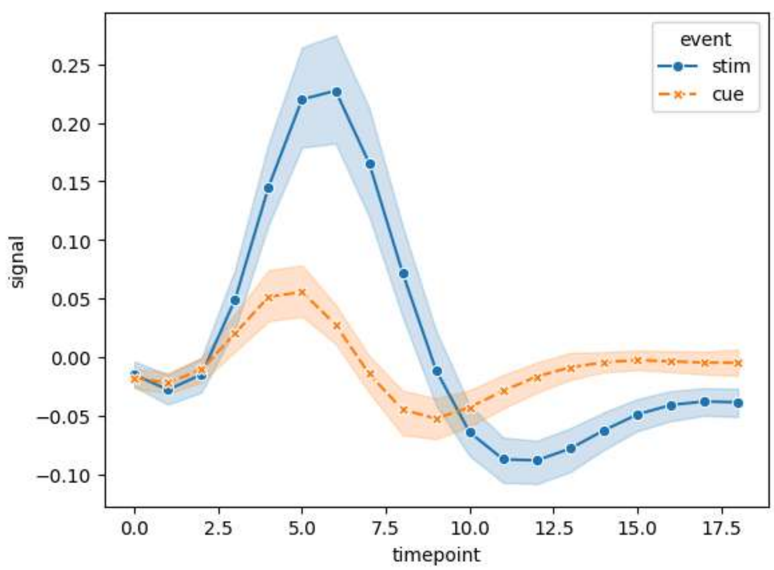
```
import seaborn as sns
from matplotlib import pyplot as plt
fmri = sns.load_dataset("fmri")
fmri.head()
```

	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()
```




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




```
import pandas as pd
import seaborn as sns
sns.set(style="whitegrid")
```

```
pokemon=pd.read_csv('/content/sample_data/pokemon.csv')
pokemon.head()
```

	#	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
0	1	Bulbasaur	Grass	Poison	45	49	49	65	65	45	1	False
1	2	Ivysaur	Grass	Poison	60	62	63	80	80	60	1	False
2	3	Venusaur	Grass	Poison	80	82	83	100	100	80	1	False
3	4	Mega Venusaur	Grass	Poison	80	100	123	122	120	80	1	False
4	5	Charmander	Fire	NaN	39	52	43	60	50	65	1	False

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
ds = sns.load_dataset('tips')
```

```
ds.head()
```

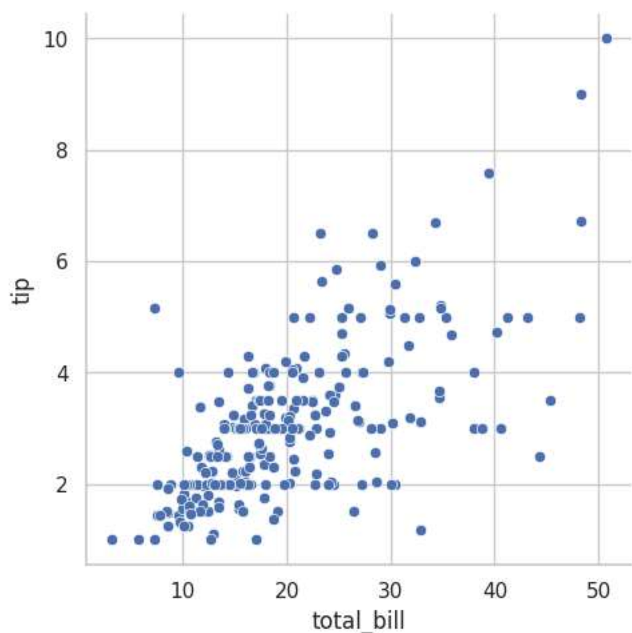
	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

```
ds.shape
```

```
(244, 7)
```

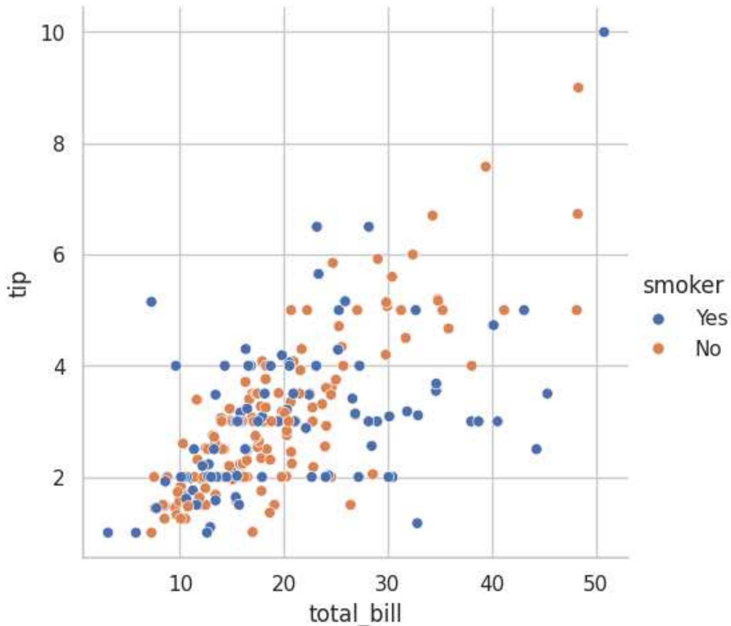
```
sns.relplot(data=ds,x='total_bill',y='tip')
```

```
<seaborn.axisgrid.FacetGrid at 0x7af01be89790>
```



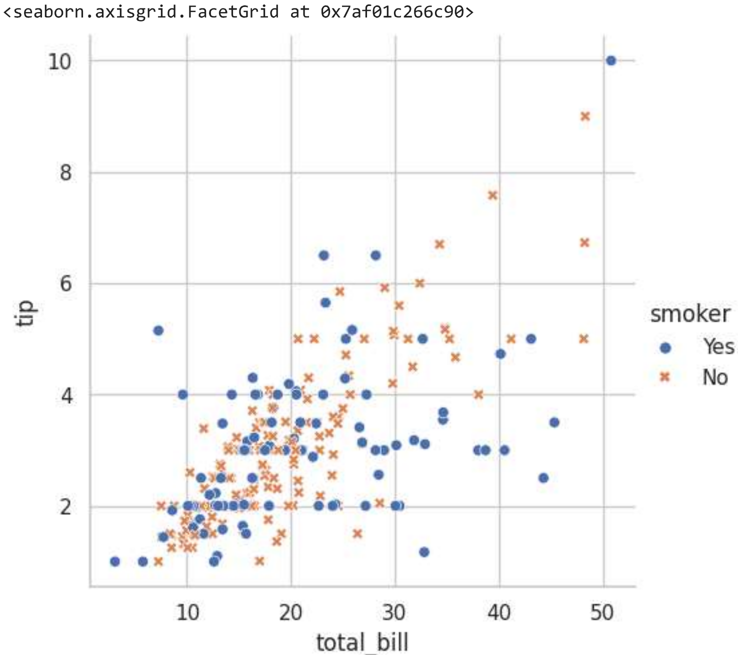

```
sns.relplot(data=ds,x='total_bill',y='tip',hue='smoker')
```

```
<seaborn.axisgrid.FacetGrid at 0x7af0182bcf80>
```



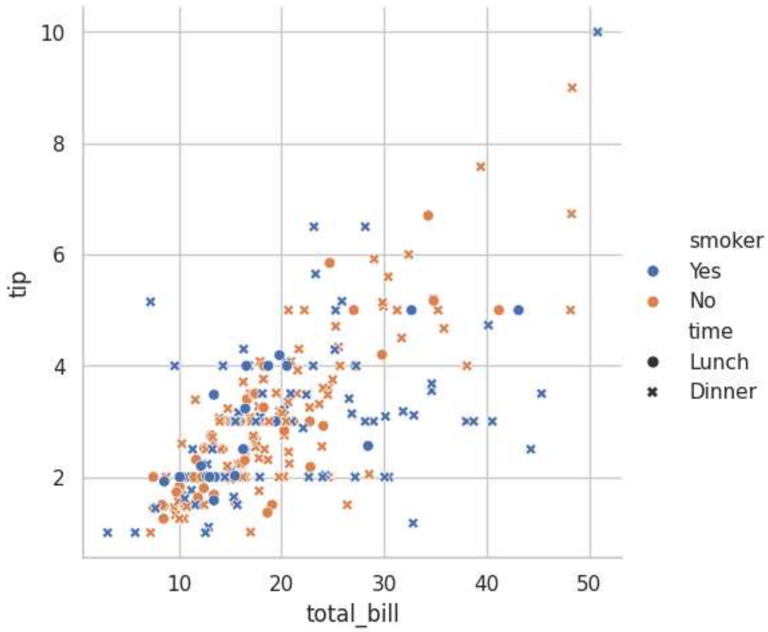

```
sns.relplot(data=ds,x='total_bill',y='tip',hue='smoker', style='smoker')
```

```
<seaborn.axisgrid.FacetGrid at 0x7af01c266c90>
```



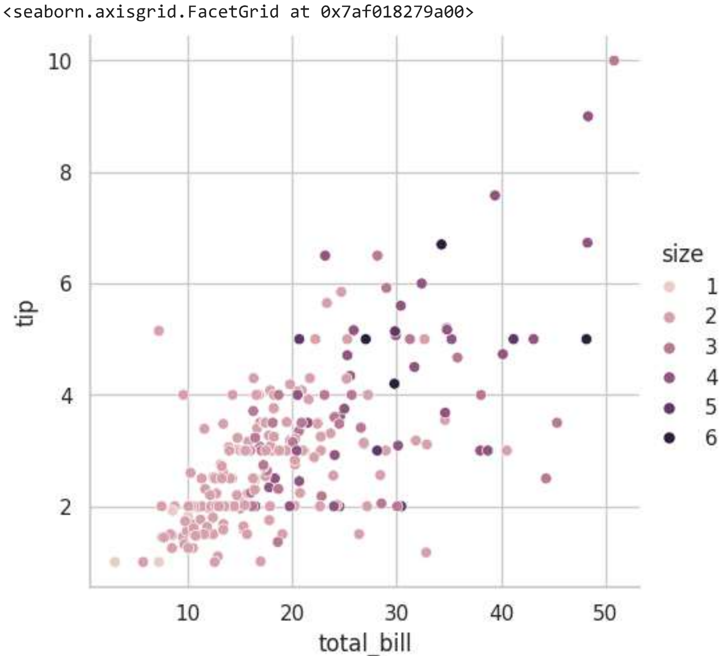

```
sns.relplot(data=ds,x='total_bill',y='tip',hue='smoker',style='time')
```

```
<seaborn.axisgrid.FacetGrid at 0x7af0181d0cb0>
```



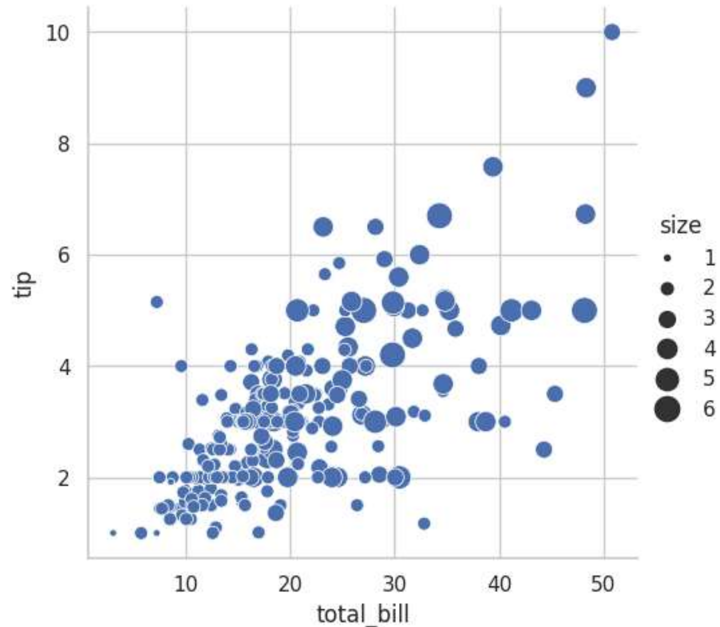

```
sns.relplot(data=ds,x='total_bill',y='tip',hue='size')
```

```
<seaborn.axisgrid.FacetGrid at 0x7af018279a00>
```



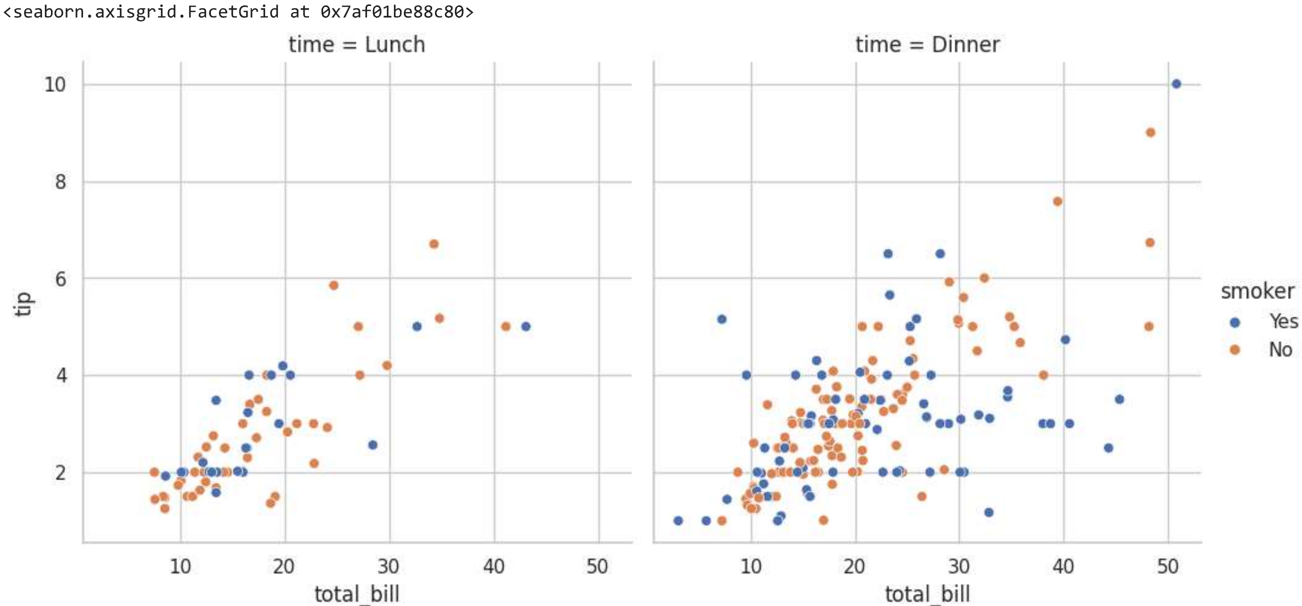

```
sns.relplot(data=ds,x='total_bill',y='tip',size ='size',sizes=(15, 200))
```

```
<seaborn.axisgrid.FacetGrid at 0x7af018278d70>
```




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

```
<seaborn.axisgrid.FacetGrid at 0x7af01be88c80>
```




```
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

```
fmri.shape
```

```
(1064, 5)
```

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

<seaborn.axisgrid.FacetGrid at 0x7af01be94f80>