In [ ]:

```
import pandas as pd
s=pd.Series()
print(s)
```

In [ ]:

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

In [ ]:

```
import pandas as pd
import numpy as np
data={"a":1,"b":2,"c":3}
s=pd.Series(data)
print(s)
```

In [ ]:

```
import pandas as pd
import numpy as np
data={"a":1,"b":2,"c":3}
s=pd.Series(data,index=["a","b","c","d"])
print(s)
```

In [ ]:

```
import pandas as pd
df=pd.DataFrame()
print(df)
```

In [ ]:

```
import pandas as pd
data=[1,2,3,4,5,6,7,8,9,10]
df=pd.DataFrame(data)
print(df)
```

In [ ]:

```
import pandas as pd
data=[["Sagnik",23],["Nirnoy",22],["Debadittya",22],["suman",24]]
df=pd.DataFrame(data,columns=["Name","Age"])
print(df)
```

In [ ]:

```
import pandas as pd
data=[["Sagnik",23],["Nirnoy",22],["Debadittya",22],["suman",24]]
df=pd.DataFrame(data,index=[1,2,3,4],columns=["Name","Age"],dtype=float)
print(df)
```

In [ ]:

```
# From Dictionary of Series
import pandas as pd
d={"one":pd.Series([1,2,3],index=["a","b","c"]),"Two":pd.Series([1,2,3,4],ir
df=pd.DataFrame(d)
print(df)
```

In [ ]:

```
# From Lists Create Series
import pandas as pd
d={"Name":["Sagnik","Nirnoy","Debargha","Debadittya"],"Age":[23,22,24,22]}
df=pd.DataFrame(d)
print(df)
```

In [ ]:

```
# From Dictionary of Lists
import pandas as pd
d=[{"a":1,"b":2},{"a":5,"b":10,"c":15}]
df=pd.DataFrame(d)
print(df)
```

In [ ]:

```
# From Dictionary of Lists
import pandas as pd
d=[{"a":1,"b":2},{"a":5,"b":10,"c":15}]
df=pd.DataFrame(d,index=["First","Second"])
print(df)
```

Basic Functionalities

In [ ]:

```
# From Lists Create Series
import pandas as pd
import numpy as np
d={"Name":pd.Series(["Sagnik","Nirnoy","Debargha","Debadittya","Ricky","Ram"
df=pd.DataFrame(d)
print("Our DataFrame is : ")
print(df)
```

In [ ]:

```
# Sum
print("The Sum is : ")
print(df.sum())
```

In [ ]:

```
# Sum
print("The Sum is : ")
print(df.sum(1))
```

In [ ]:

```
# Mean
print("The Mean is : ")
print(df.mean())
```

In [ ]:

```
# Median
print("The Median is : ")
print(df.median())
```

In [ ]:

```
# Standard Deviation
print("The Standard Deviation is : ")
print(df.std())
```

In [ ]:

```
# Cumulative Sum
print("The Cumulative Sum is : ")
print(df.cumsum())
```

In [ ]:

```
print(df.prod())
```

In [ ]:

```
print(df.cumprod())
```

Covariance

In [ ]:

```
# Covariance
# Cov Series
import pandas as pd
import numpy as np
s1=pd.Series(np.random.randn(10))
s2=pd.Series(np.random.randn(10))
print(s1)
print("\n")
print(s2)
print("\n")
print(s1.cov(s2))
```

In [ ]:

```
# Cov DataFrame
import pandas as pd
import numpy as np
frame=pd.DataFrame(np.random.randn(10,5),columns=["a","b","c","d","e"])
print(frame)
print("\n")
print(frame["a"].cov(frame["b"]))
print("\n")
print(frame.cov())
```

Correlation

In [ ]:

```
# Correlation
import pandas as pd
import numpy as np
frame=pd.DataFrame(np.random.randn(10,5),columns=["a","b","c","d","e"])
print(frame)
print("\n")
print("The Correlation between frame a &  frame b is : ")
print(frame["a"].corr(frame["b"]))
print("\n")
print("The Correlation of the whole frame is : ")
print(frame.corr())
```

Missing values in the Dataset

In [ ]:

```
import numpy as np
import pandas as pd
df=pd.DataFrame(np.random.randn(5,3),index=["a","c","e","f","h"],columns=["C
print(df)
```

In [ ]:

```
df=df.reindex(["a","b","c","d","e","f","g","h"])
print(df)
```

In [ ]:

```
import numpy as np
import pandas as pd
df=pd.DataFrame(np.random.randn(3,3),index=["f","g","h"],columns=["One","Two
df=df.reindex(["a","b","c"])
print(df)
print("\n")
print("NaN is replaced with 0")
print(df.fillna(0))
```

In [ ]:

```
# Fill Na Forward
print(df.fillna(method="pad"))
```

In [ ]:

```
# Fill Na Backward
print(df.fillna(method="backfill"))
```

In [ ]:

```
print(df.dropna())
```

In [ ]:

```
print(df.dropna(axis=1))
```

Matplot Library

In [ ]:

```
# Matplotlib
import matplotlib.pyplot as plt
import numpy as np
import math
x=np.arange(0,math.pi*2,0.05)
y=np.sin(x)
plt.plot(x,y)
plt.xlabel("Angle")
plt.ylabel("Sine Wave")
plt.show()
```

In [ ]:

```
from numpy import *
from pylab import *
x=linspace(-3,3,30)
y=x**2
plot(x,y)
show()
```

In [ ]:

```
from pylab import *
x=linspace(-3,3,30)
y=x**2
plot(x,y,"y.")
show()
```

In [ ]:

```
from pylab import *
x=linspace(-3,3,30)
y=x**2
plot(x,y,"g.")
show()
```

Barplot

In [ ]:

```python
# ax.bar(x,height,width,bottom)
import matplotlib.pyplot as plt
fig=plt.figure()
ax=fig.add_axes([0,0,1,1])
language=["c","c++","Python","R","SQL","Java","PHP","SAS","SPSS"]
students=[23,17,35,40,32,28,20,30,32]
ax.bar(language,students)
plt.show()
```

In [ ]:

```python
import matplotlib.pyplot as plt
import numpy as np
data=[[30,25,27,29],[40,26,38,43],[35,22,15,28]]
x=np.arange(4)
fig=plt.figure()
ax=fig.add_axes([0,0,1,1])
ax.bar(x+0.00,data[0],color="b",width=0.25)
ax.bar(x+0.25,data[1],color="r",width=0.25)
ax.bar(x+0.50,data[2],color="g",width=0.25)
plt.show()
```

In [ ]:

```python
import matplotlib.pyplot as plt
import numpy as np
N=5
men=(20,35,29,27,30)
women=(25,28,27,29,31)
i=np.arange(N)
width=0.35
fig=plt.figure()
ax=fig.add_axes([0,0,1,1])
ax.bar(i,men,width,color="r")
ax.bar(i,women,width,bottom=men,color="m")
ax.legend(labels=["Men","Women"])
plt.show()
```

Scatter Plot

In [ ]:

```python
import matplotlib.pyplot as plt
boys_grade=[68,35,91,75,39,99,98,88,80,100]
girls_grade=[50,82,71,29,31,78,82,94,99,97]
grade_range=[52,56,71,81,92,55,65,87,98,99]
fig=plt.figure()
ax=fig.add_axes([0,0,1,1])
ax.scatter(range,women,color="r")
ax.scatter(range,men,color="b")
ax.set_title("Scatter Plot")
ax.set_xlabel("Grade_Range")
ax.set_ylabel("Score")
plt.show()
```

Logistic Regression

In [ ]:

```
# Supervised Learning used for predicting the probability of a targeted vari
# Dichotomous - Only Two Possible Outcomes.
# P(Y=1) as a function of x
# Simplest ML algo
# Assumptions :
# in case of Binomial LR, the target variables must be binary & the desired
# There should no be multi-colinearity in data
# We should choose a large sample size of data for Logistic Regression
```

In [ ]:

```
import sklearn
from sklearn import datasets
from sklearn import linear_model
from sklearn import metrics
from sklearn.model_selection import train_test_split
```

In [ ]:

```
digits=datasets.load_digits()
print(digits)
```

In [ ]:

```
x=digits.data
y=digits.target
```

In [ ]:

```
print(x)
```

In [ ]:

```
print(y)
```

In [ ]:

```
x_train,x_test,y_train,x_test=train_test_split(x,y,test_size=0.4,random_stat
```

```
digreg=linear_model.LogisticRegression()
```

In [ ]:

```
digerg.fit()
```

In [ ]:

```
```

In [ ]:

```
y_preds=digerg.predict(x_test)
```

In [ ]:

```
print(y_preds)
```

In [ ]:

```
# Accuracy
print("Accuracy of the Logistic Regression is :",metrics.accuracy_Score(y_te
```

Linear Regression

In [ ]:

```
# Simple Linear Regression
```

In [ ]:

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

In [ ]:

```
def coef_estimation(x,y):
    n=np.size(x)
    m_x,m_y=np.mean(x),np.mean(y)
    ss_xy=np.sum(y*x)-n*m_y*m_x
    ss_xx=np.sum(y*x)-n*m_x*m_x
    b_1=ss_xy/ss_xx
    b_0=m_y-b_1*m_x
    return(b_0,b_1)
```

In [ ]:

```
def plot_regression_line(x,y,b):
    plt.scatter(x,y,color="b",marker="0",s=30)
    y_pred=b[0]-b[1]*x
    plt.plot(x,y_pred,color="g")
    plt.xlabel("x")
    plt.ylabel("y")
    plt.show()
```

In [ ]:

```
def main():
    x=np.array([0,1,2,3,4,5,6,7,8,9,10])
    y=np.array(100,300,200,500,600,400,800,700,900,401,5002)
```

In [ ]:

In [1]:

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets,linear_model
from sklearn.metrics import mean_squared_error,r2_score
```

In [2]:

```
diabetes=datasets.load_diabetes()
```

In [5]:

```
x=diabetes.data[:,np.newaxis,2]
```

In [6]:

```
x_train=x[:-30]
x_test=x[-30:]
```

In [7]:

```
y_train=diabetes.target[:-30]
y_test=diabetes.target[-30:]
```

In [8]:

```
regr=linear_model.LinearRegression()
```

```
regr.fit(x_train,y_train)
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: