TASK - 1

Open Jupyter Notebook on your system. Create a notebook and write some simple Python codes to ensure everything works properly. If you face any problem, ask the tutor to help you.

```
import numpy as np

x = 10
y = 3

if x > y:
    print("x is greater than y")
else:
    print("x is not greater than y")

a = 5
while a < 20:
    print(a)
    a = a * 2 - 1

print(f"Addition of X and Y is {x + y}")
print(f"Subtraction of X and Y is {x - y}")
print(f"Multiplication of X and Y is {x * y}")
print(f"Division of X and Y is {x / y}")

arr = np.array([[[13, 20, 53], [21, 65, 97], [12, 98, 25]], [[31, 43, 79], [40, 15, 36], [17, 52, 90]]])
print(f"Array is {arr}")
print(f"Type of array is {type(arr)}")
print(f"Shape of array is {arr.shape}")
print(f"Size of array is {arr.size}")
print(f"Dimension of array is {arr.ndim}")

print(arr[0, 1, 1])

print(arr[:,2,:])

x = np.ones((3,3))
print("Checkerboard pattern:")
x = np.zeros((8,8),dtype=int)
x[1::2,::2] = 1
x[::2,1::2] = 1
print(x)

x = arr.copy()
arr[0] = [13, 57, 53], [21, 92, 97], [12, 67, 25]
print(f"Copy Array: {arr}")
print(f"Copy x: {x}")

print(f"Copy: {x.base}")
```

```
y= arr.view()
arr[0] = [1, 57, 53], [2, 92, 97], [7, 67, 25]
print(f"View Array: {arr}")
print(f"View y: {y}")


print(f"View: {y.base}")
```

```
  [21 65 97]
  [12 98 25]]

 [[31 43 79]
  [40 15 36]
  [17 52 90]]]
Type of array is <class 'numpy.ndarray'>
Shape of array is (2, 3, 3)
Size of array is 18
Dimension of array is 3
65
[[12 98 25]
 [17 52 90]]
Checkerboard pattern:
[[0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]
 [0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]
 [0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]
 [0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]]
Copy Array: [[[13 57 53]
  [21 92 97]
  [12 67 25]]

 [[31 43 79]
  [40 15 36]
  [17 52 90]]]
Copy x: [[[13 20 53]
  [21 65 97]
  [12 98 25]]

 [[31 43 79]
  [40 15 36]
  [17 52 90]]]
Copy: None
View Array: [[[ 1 57 53]
  [ 2 92 97]






 [[31 43 79]
```

```
      [[31 43 79]
       [40 15 36]
       [17 52 90]]]
     View: [[[ 1 57 53]
       [ 2 92 97]
       [ 7 67 25]]

      [[31 43 79]
       [40 15 36]
       [17 52 90]]]
```

TASK - 2

Download and open the Workshop#1 Python file on Canvas and complete the tasks

A.

(a) Plot y against x in a two-dimensional graph.

(b) Run a simple regression model and find the coeffiicent m and b according to the lecture file.

(c) Precidct y values using the regression model you built in (b).

(d) Plot two graphs on the same plane. The first graph should display the predicted y-values against the x-values. The second graph should display the original y-values against the x-values.

(e) Are the predicted values accurate? How can we measure the accuracy of the model?

NOTE: You need to research and find relevant Python modules to complete the task

Hint: You can use linregress from the scipy package for this task

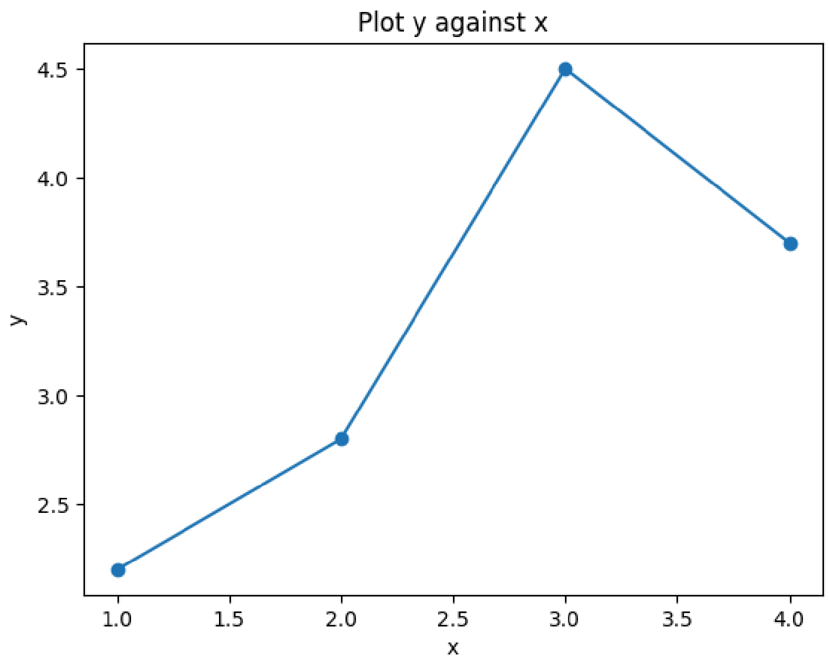
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import linregress
from sklearn.metrics import mean_squared_error, r2_score
```

Plot of Y against X

```
x = [1, 2, 3, 4]
y = [2.2, 2.8, 4.5, 3.7]

plt.plot(x,y, marker='o')
plt.title('Plot y against x')
plt.xlabel('x')
plt.ylabel('y')

plt.show()
```



Finding Coefficient of m and b

```
slope, intercept = linregress(x,y)[:2]
print(f"m: {slope:.3f}")
print(f"b: {intercept:.3f}")
```

```
m: 0.620
b: 1.750
```

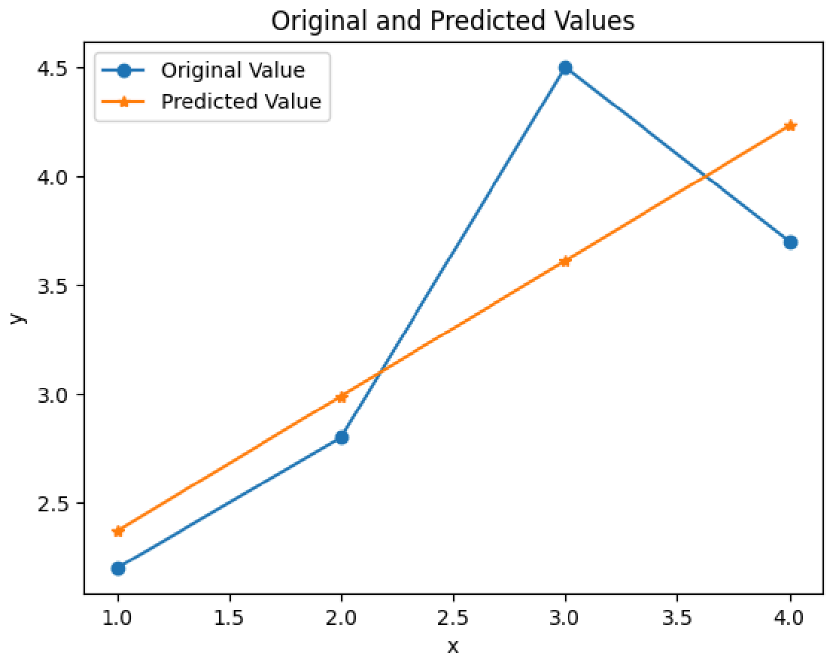
Predicting the Value of Y

```
y_pred = slope * np.array(x) + intercept
print(f"Predicted y: {y_pred}")
```

```
Predicted y: [2.37 2.99 3.61 4.23]
```

Plotting Original Values of Y and Predicted Values of Y against X on the same plane.

```
plt.plot(x,y, label = "Original Value", marker = 'o')
plt.plot(x,y_pred, label = "Predicted Value", marker = '*')
plt.title('Original and Predicted Values')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.show()
```



Evaluating Model Accuracy

```
mse = mean_squared_error(y, y_pred)
r2 = r2_score(y, y_pred)

print(f"nModel Accuracy Metrics:")
print(f"Mean Squared Error: {mse:.3f}")
print(f"R-squared Score: {r2:.3f}")

if r2 > 0.7:
    print("\nThe model is a good fit (R² > 0.7)")
elif r2 > 0.5:
    print("\nThe model is a moderate fit (R² > 0.5)")
else:
    print("\nThe model is a poor fit (R² ≤ 0.5)")
```

```
nModel Accuracy Metrics:
Mean Squared Error: 0.285
R-squared Score: 0.628
```

```
The model is a moderate fit (R² > 0.5)
```

B.

Repeat task 1 for the following set of points X and y.

NOTE 1: This is an example of multiple linear regression when the input (predictor) has more than one dimension.

NOTE 2: You need to plot the graphs in a 3 dimensional space

Hint: You can use LinearRegression class from the sklearn package for this task

```
from sklearn.linear_model import LinearRegression
from mpl_toolkits.mplot3d import Axes3D
```

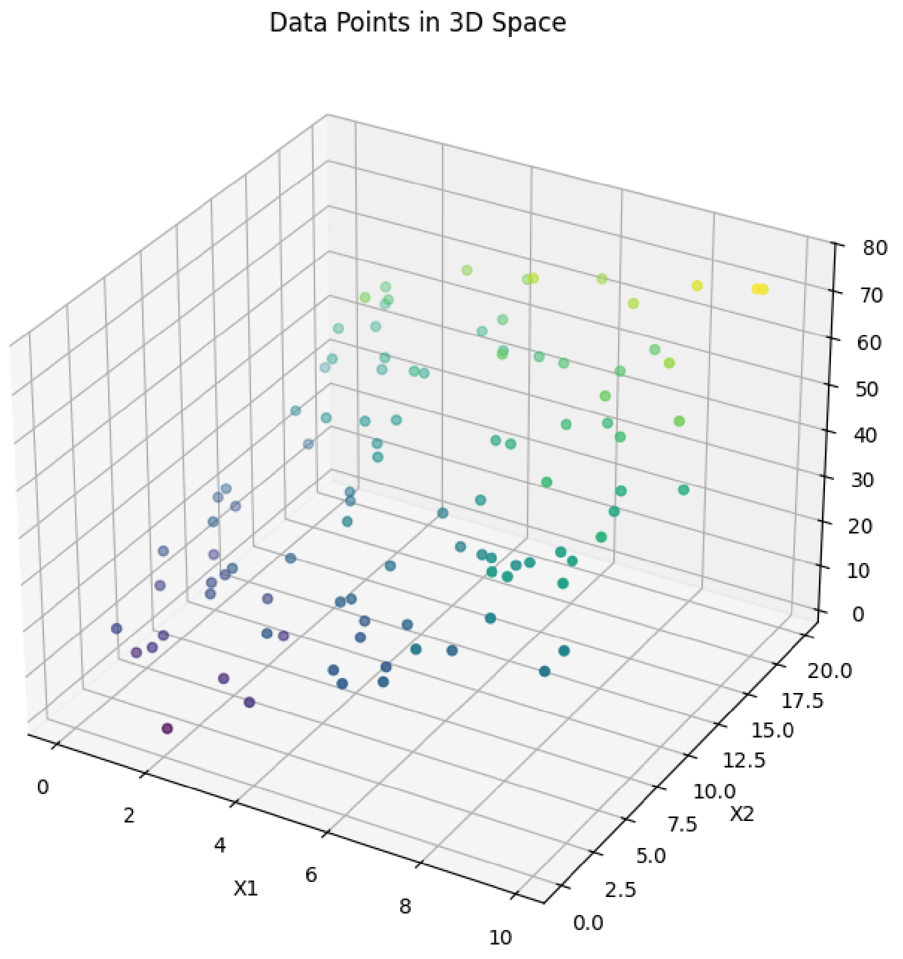
Plotting data points of x1, x2 and y in 3d space

```
# Generate example data for multiple regression
np.random.seed(42)
X1 = np.random.rand(100) * 10  # Predictor 1
X2 = np.random.rand(100) * 20  # Predictor 2


# Combine predictors into a single matrix
X = np.column_stack((X1, X2))

# Response variable with noise
y = 4 + 3 * X1 + 2 * X2 + np.random.randn(100) * 5

fig = plt.figure(figsize=(12, 8))
ax = plt.axes(projection='3d')
ax.scatter3D(X1, X2, y, c=y, cmap='viridis', )
ax.set_xlabel('X1')
ax.set_ylabel('X2')
ax.set_zlabel('y')
ax.set_title('Data Points in 3D Space')
plt.show()
```



Finding Coefficient of m and b using model (LinearRegression)

```
model = LinearRegression()
model.fit(X, y)

x1_slope = model.coef_[0]
x2_slope = model.coef_[1]
intercept = model.intercept_

print(f"m1: {x1_slope:.3f}")
print(f"m2: {x2_slope:.3f}")
print(f"b: {intercept:.3f}")
```

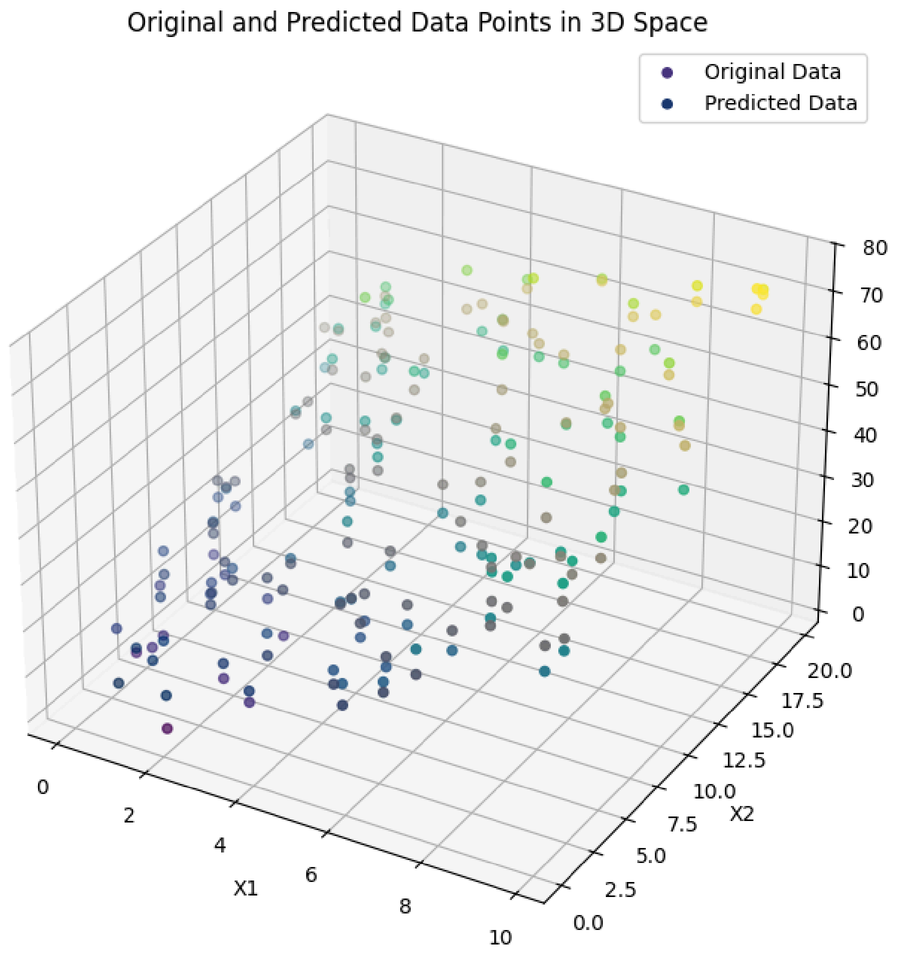
```
m1: 2.829
m2: 2.180
b: 3.553
```

Predicting Y

```
y_pred = model.predict(X)
```

Plotting Original and Predicted Data Points in 3D Space

```
fig = plt.figure(figsize=(12, 8))
ax = plt.axes(projection='3d')
ax.scatter3D(X1, X2, y, c=y, cmap='viridis', label='Original Data')
ax.scatter3D(X1, X2, y_pred, c=y_pred, cmap='cividis', label='Predicted Data')
ax.set_xlabel('X1')
ax.set_ylabel('X2')
ax.set_zlabel('y')
ax.set_title('Original and Predicted Data Points in 3D Space')
ax.legend()
plt.show()
```



Evaluating Model Accuracy

```
mse = mean_squared_error(y, y_pred)
r2 = r2_score(y, y_pred)

print(f"nModel Accuracy Metrics:")
print(f"Mean Squared Error: {mse:.3f}")
print(f"R-squared Score: {r2:.3f}")

if r2 > 0.7:
    print("\nThe model is a good fit (R² > 0.7)")
elif r2 > 0.5:
    print("\nThe model is a moderate fit (R² > 0.5)")
else:
    print("\nThe model is a poor fit (R² ≤ 0.5)")
```

```
nModel Accuracy Metrics:
Mean Squared Error: 23.642
R-squared Score: 0.905

The model is a good fit (R² > 0.7)
```

[illegible]