Introduction to function NUMPY,Matplotlib and tensorflow

#numpy

#Array Creation Functions:

```
import numpy as np

# Creating an array from a list
arr1 = np.array([1, 2, 3, 4, 5])
print(arr1)

# Creating a 2D array (matrix)
arr2 = np.array([[1, 2, 3], [4, 5, 6]])
print(arr2)

print("......................................")
```

```
[1 2 3 4 5]
[[1 2 3]
 [4 5 6]]
......................................
```

#Zeros, Ones, and Empty Arrays

```
# Array of zeros
zeros_arr = np.zeros((3, 4))  # 3 rows, 4 columns
print(zeros_arr)


print("......................................")
```

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

Array of ones

```
ones_arr = np.ones((2, 3))  # 2 rows, 3 columns
print(ones_arr)
```

```
[[1. 1. 1.]
 [1. 1. 1.]]
```

```
# Empty array (random values)
empty_arr = np.empty((2, 2))  # 2x2 empty array
print(empty_arr)
```

```
[[6.23042070e-307 4.67296746e-307]
 [1.69121096e-306 1.69761995e-312]]
```

```
#Shape, Size, and Data Type

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

# Shape of the array
print(arr.shape)
# Size (total number of elements)
print(arr.size)
# Data type of elements in the array
print(arr.dtype)


print("......................................")
```

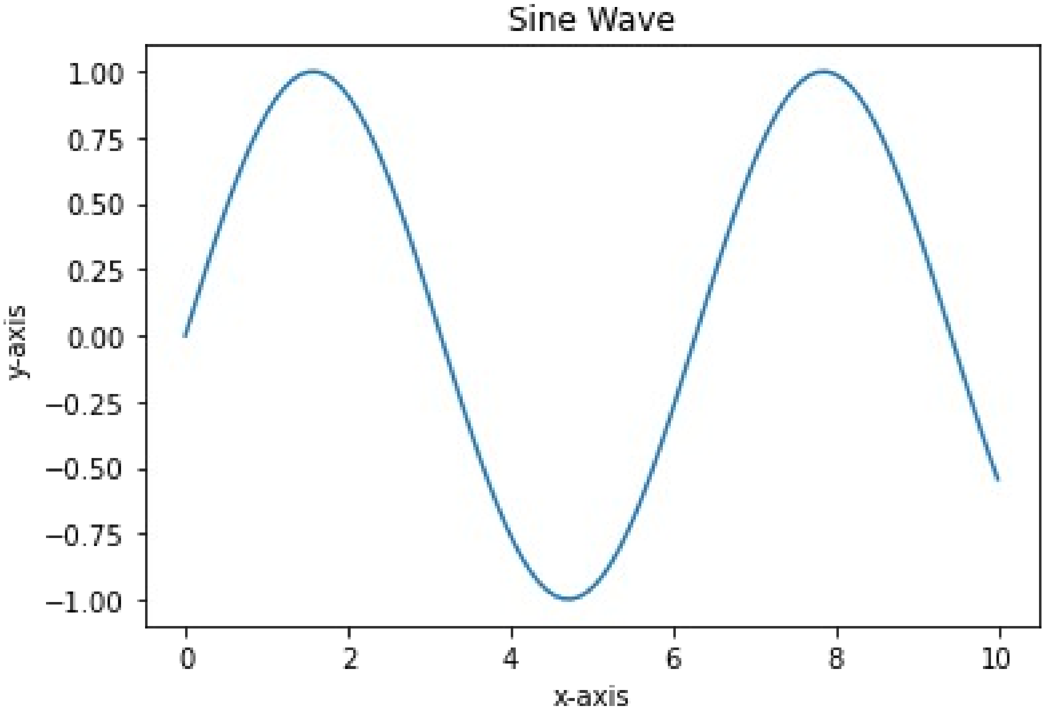
```
(2, 3)
6
int32
......................................
```

```
#Reshaping Arrays

arr = np.arange(12)  # 1D array with 12 elements
reshaped_arr = arr.reshape(3, 4)  # Reshape to a 3x4 array
print(reshaped_arr)

print("......................................")
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
......................................
```

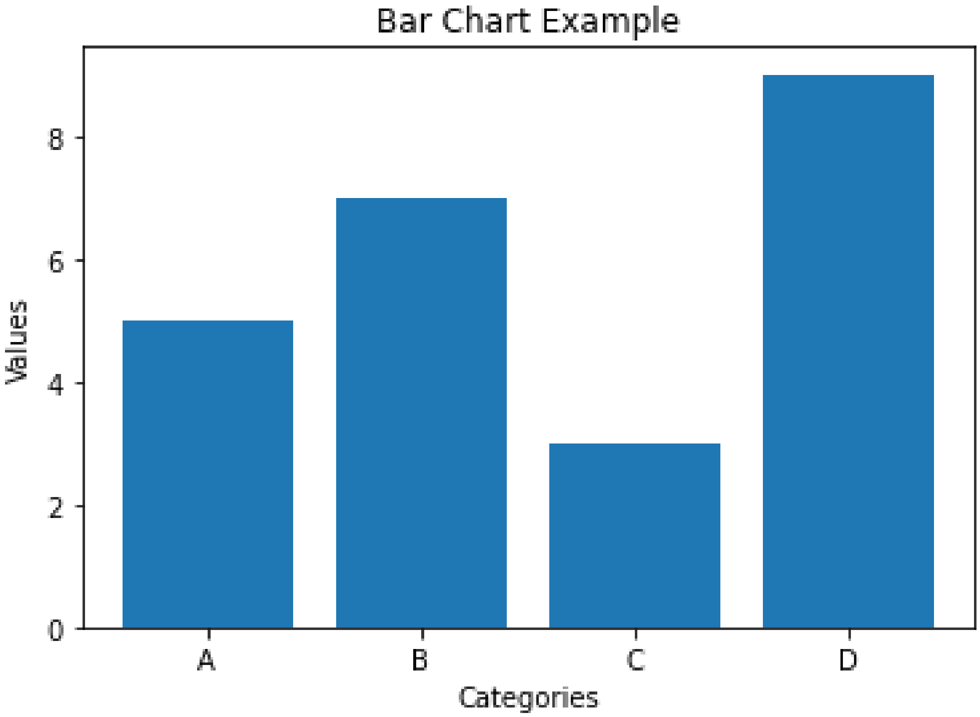
```
#Mathematical Operations

arr1 = np.array([[1, 2], [3, 4]])
arr2 = np.array([[5, 6], [7, 8]])

# Element-wise addition
sum_arr = arr1 + arr2
print(sum_arr)

# Matrix multiplication
mul_arr = np.dot(arr1, arr2)
print(mul_arr)
```

```
[[ 6  8]
 [10 12]]
[[19 22]
 [43 50]]

#Matplotlib Graph Plotting:
#Line Plot:

import matplotlib.pyplot as plt

x = np.linspace(0, 10, 100)
y = np.sin(x)

plt.plot(x, y)
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.title('Sine Wave')
plt.show()

print("......................................")
```



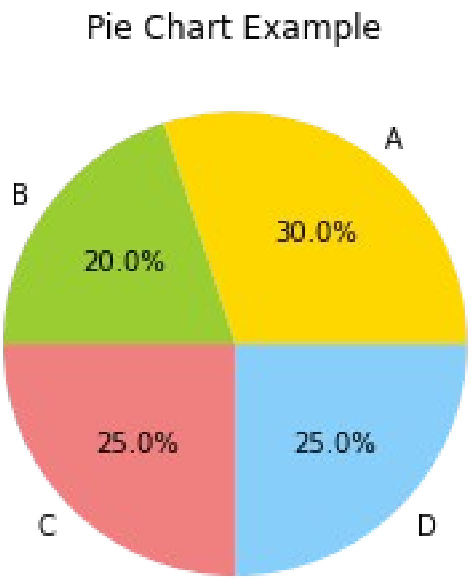
```
......................................

#Bar Chart
#Creating a simple bar chart to display different categories and their
values.
```

```python
categories = ['A', 'B', 'C', 'D']
values = [5, 7, 3, 9]

plt.bar(categories, values)
plt.xlabel('Categories')
plt.ylabel('Values')
plt.title('Bar Chart Example')
plt.show()

print("......................................")
```



```
......................................
```

```python
#Pie Chart

labels = ['A', 'B', 'C', 'D']
sizes = [30, 20, 25, 25]
colors = ['gold', 'yellowgreen', 'lightcoral', 'lightskyblue']

plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%')
plt.title('Pie Chart Example')
plt.show()

print("......................................")
```



```
......................................

#tensor


#Basic Arithmetic Operations:

import tensorflow as tf




#Matrix Operations:

# Matrix multiplication
matrix_a = tf.constant([[1, 2], [3, 4]])
matrix_b = tf.constant([[5, 6], [7, 8]])
result_matmul = tf.matmul(matrix_a, matrix_b)

# Transpose
result_transpose = tf.transpose(matrix_a)
print(result_transpose)
print(result_matmul)

tf.Tensor(
[[1 3]
 [2 4]], shape=(2, 2), dtype=int32)
tf.Tensor(
[[19 22]
 [43 50]], shape=(2, 2), dtype=int32)
```

```
#Activation Functions:

# Applying activation functions
tensor = tf.constant([-2.0, -1.0, 0.0, 1.0, 2.0])

# ReLU (Rectified Linear Unit)
result_relu = tf.nn.relu(tensor)

# Sigmoid
result_sigmoid = tf.nn.sigmoid(tensor)

# Softmax
result_softmax = tf.nn.softmax(tensor)
print(result_relu)
print(result_relu)
print(result_softmax)
print("......................................")
```

```
tf.Tensor([0. 0. 0. 1. 2.], shape=(5,), dtype=float32)
tf.Tensor([0. 0. 0. 1. 2.], shape=(5,), dtype=float32)
tf.Tensor([0.01165623 0.03168492 0.08612854 0.23412165 0.6364086 ],
shape=(5,), dtype=float32)
......................................
```

```
#Strided Convolution
import tensorflow as tf


model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu', strides=(2,
2), input_shape=(64, 64, 3))
])

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 31, 31, 64)        1792

=================================================================
Total params: 1,792
Trainable params: 1,792
Non-trainable params: 0
_________________________________________________________________
```