

1

Perform elementary mathematical operations in Octave/MATLAB/R/Python like addition, multiplication, division and exponentiation.

```
In [1]: # Addition  
5 + 7
```

Out[1]: 12

```
In [2]: # Subtraction  
5 - 7
```

Out[2]: -2

```
In [3]: # Multiplication  
5 * 7
```

Out[3]: 35

```
In [4]: # Division  
5 / 7
```

Out[4]: 0.7142857142857143

```
In [5]: # Exponentiation  
5 ^ 7
```

Out[5]: 2

2

Perform elementary logical operations in Octave/MATLAB/R/Python (like OR, AND, Checking for Equality, NOT, XOR).

```
In [6]: # OR  
True | False
```

Out[6]: True

```
In [7]: # AND  
True & False
```

Out[7]: False

```
In [8]: # Check for inequality  
True == False
```

Out[8]: False

```
In [9]: # NOT
~True
```

Out[9]: -2

```
In [10]: # XOR
print(True ^ True)
print(True ^ False)

False
True
```

3

Create, initialize and display simple variables and simple strings and use simple formatting for variable.

```
In [11]: # Creating and initializing a variable
a = 5
x1 = 10
x2 = 20
name = "Anshul"
```

```
In [12]: # Displaying variables
print(a)
print(x1)
print(x2)
print(name)
```

5
10
20
Anshul

4

Create/Define single dimension / multi-dimension arrays, and arrays with specific values like array of all ones, all zeros, array with random values within a range, or a diagonal matrix.

```
In [13]: import numpy as np
```

```
In [14]: # Creating single-dimension arrays
x = np.array([1, 2, 3, 4, 5])
print('x = ', x)
```

```
y = np.array([[1], [2], [3]])
print('y = \n', y)
```

```
x = [1 2 3 4 5]
y =
[[1]
 [2]
 [3]]
```

```
In [15]: # Creating multi-dimension arrays
z = np.array([[1, 2, 3], [6, 7, 8]])
print('z = \n', z)

z1 = np.matrix('1 2 3; 6 7 8')
print('z1 = \n', z1)
```

```
z =
[[1 2 3]
 [6 7 8]]
z1 =
[[1 2 3]
 [6 7 8]]
```

```
In [16]: # Matrix with all ones
A = np.ones((4, 4))
A
```

```
Out[16]: array([[1., 1., 1., 1.],
               [1., 1., 1., 1.],
               [1., 1., 1., 1.],
               [1., 1., 1., 1.]])
```

```
In [17]: # Matrix with all zeros
B = np.zeros((4, 4))
B
```

```
Out[17]: array([[0., 0., 0., 0.],
               [0., 0., 0., 0.],
               [0., 0., 0., 0.],
               [0., 0., 0., 0.]])
```

```
In [18]: # Matrix with random values within a range
C = np.random.randint(20, 50, (4,5))
print("C = \n", C)

# Matrix with range
C1 = np.arange(12).reshape((3, 4))
print("C1 = \n", C1)
```

```
C =
[[30 39 40 29 32]
 [23 47 36 32 22]
 [41 27 26 20 49]
 [35 39 46 30 44]]
C1 =
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
```

```
In [19]: # Diagonal matrix
D = np.diag([1, 2, 3, 4, 5])
print('D = \n', D)
```

```
D =
```

```
[[1 0 0 0 0]
 [0 2 0 0 0]
 [0 0 3 0 0]
 [0 0 0 4 0]
 [0 0 0 0 5]]
```

5

Use command to compute the size of a matrix, size/length of a particular row/column, load data from a text file, store matrix data to a text file, finding out variables and their features in the current scope.

```
In [20]: import numpy

A = np.arange(12).reshape((3, 4))
A
```

```
Out[20]: array([[ 0,  1,  2,  3],
                [ 4,  5,  6,  7],
                [ 8,  9, 10, 11]])
```

```
In [21]: # Size of matrix
# A.size
np.size(A)
```

```
Out[21]: 12
```

```
In [22]: # Shape of matrix
A.shape
```

```
Out[22]: (3, 4)
```

```
In [23]: # Length of 1st row
len(A[0])
```

```
Out[23]: 4
```

```
In [24]: # Length of first column
np.size(A, 0)
```

```
Out[24]: 3
```

```
In [25]: # Loading data from text file
#np.loadtxt("input5.txt", delimiter=',', dtype=int)
import pandas as pd

df = pd.read_csv('input5.txt')
df
```

```
Out[25]:
```

	Sell	List	Living	Rooms	Beds	Baths	Age	Acres	Taxes
0	142	160	28	10	5	3	60	0.28	3167

	Sell	"List"	"Living"	"Rooms"	"Beds"	"Baths"	"Age"	"Acres"	"Taxes"
1	175	180	18	8	4	1	12	0.43	4033
2	129	132	13	6	3	1	41	0.33	1471
3	138	140	17	7	3	1	22	0.46	3204
4	232	240	25	8	4	3	5	2.05	3613
5	135	140	18	7	4	3	9	0.57	3028
6	150	160	20	8	4	3	18	4.00	3131
7	207	225	22	8	4	2	16	2.22	5158
8	271	285	30	10	5	2	30	0.53	5702

```
In [26]: df.drop(df.tail(5).index,inplace=True)
df
```

```
Out[26]:
```

	Sell	"List"	"Living"	"Rooms"	"Beds"	"Baths"	"Age"	"Acres"	"Taxes"
0	142	160	28	10	5	3	60	0.28	3167
1	175	180	18	8	4	1	12	0.43	4033
2	129	132	13	6	3	1	41	0.33	1471
3	138	140	17	7	3	1	22	0.46	3204

```
In [27]: # Saving data to txt file
df.to_csv('output5.txt', sep=' ')
```

```
In [28]: # List of features
df.columns.values
```

```
Out[28]: array(['Sell', ' "List"', ' "Living"', ' "Rooms"', ' "Beds"', ' "Baths"',
               ' "Age"', ' "Acres"', ' "Taxes"'], dtype=object)
```

6

Perform basic operations on matrices (like addition, subtraction, multiplication) and display specific rows or columns of the matrix.

```
In [29]: import numpy as np
A = np.array([[3, 6, 9], [5, -10, 15], [-7, 14, 21]])
B = np.array([[9, -18, 27], [11, 22, 33], [13, -26, 39]])
print("A = \n", A, "\nB = \n", B)
```

```
A =
[[ 3  6  9]
 [ 5 -10 15]
 [-7 14 21]]
B =
[[ 9 -18 27]
 [11 22 33]
 [13 -26 39]]
```

```
In [30]: # Matrix Addition
C = A + B
print('C = A + B =\n', C)
```

```
C = A + B =
[[ 12 -12  36]
 [ 16  12  48]
 [  6 -12  60]]
```

```
In [31]: # Matrix Subtraction
C = A - B
print('C = A - B =\n', C)
```

```
C = A - B =
[[ -6  24 -18]
 [ -6 -32 -18]
 [-20  40 -18]]
```

```
In [32]: # Matrix Multiplication
C = A.dot(B)
print('C = A * B =\n', C)
```

```
C = A * B =
[[ 210 -156  630]
 [ 130 -700  390]
 [ 364 -112 1092]]
```

```
In [33]: # Print 2nd row of Matrix A
print(A[1:2])
```

```
[[  5 -10  15]]
```

```
In [34]: # Print 1st row of Matrix B
print(B[:1])
```

```
[[  9 -18  27]]
```

```
In [35]: # Print 2nd column of Matrix A
print(A[:,1:2])
```

```
[[  6]
 [-10]
 [ 14]]
```

```
In [36]: # Print 3rd column of Matrix B
print(B[:,2:3])
```

```
[[27]
 [33]
 [39]]
```

7

Perform other matrix operations like converting matrix data to absolute values, taking the negative of matrix values, adding/removing rows/columns from a matrix, finding the maximum or minimum values in a matrix or in a

row/column, and finding the sum of some/all elements in a matrix.

```
In [37]: import numpy as np
A = np.array([[3, 6, 9], [5, -10, 15], [-7, 14, 21]])
B = np.array([[9, -18, 27], [11, 22, 33], [13, -26, 39]])
print("A = \n", A, "\nB = \n", B)
```

```
A =
[[ 3  6  9]
 [ 5 -10 15]
 [-7 14 21]]
B =
[[ 9 -18 27]
 [11 22 33]
 [13 -26 39]]
```

```
In [38]: # Converting matrix A data to its absolute values
np.absolute(A)
```

```
Out[38]: array([[ 3,  6,  9],
               [ 5, 10, 15],
               [ 7, 14, 21]])
```

```
In [39]: # Converting matrix B data to its negative values
np.negative(B)
```

```
Out[39]: array([[ -9, 18, -27],
               [-11, -22, -33],
               [-13, 26, -39]])
```

```
In [40]: # Deleting a row from Matrix A
np.delete(A, 1, 0)
```

```
Out[40]: array([[ 3,  6,  9],
               [-7, 14, 21]])
```

```
In [41]: # Deleting a column from Matrix B
np.delete(B, 0, 1)
```

```
Out[41]: array([[ -18, 27],
               [ 22, 33],
               [-26, 39]])
```

```
In [42]: # Adding a row to Matrix A
np.append(A, np.array([[23, -45, 56]]), axis=0)
```

```
Out[42]: array([[ 3,  6,  9],
               [ 5, -10, 15],
               [-7, 14, 21],
               [23, -45, 56]])
```

```
In [43]: # Adding a column to Matrix B
np.append(B, [[23], [-45], [56]], axis=1)
```

```
Out[43]: array([[ 9, -18, 27, 23],
               [11, 22, 33, -45],
               [13, -26, 39, 56]])
```

```
In [44]: # Maximum of 2nd row of Matrix A
np.max(A, 0)[1]
```

Out[44]: 14

```
In [45]: # Minimum of 2nd row of Matrix A  
np.min(A, 0)[1]
```

Out[45]: -10

```
In [46]: # Maximum of 3rd column of Matrix B  
np.max(B, 1)[2]
```

Out[46]: 39

```
In [47]: # Minimum of 3rd column of Matrix B  
np.min(B, 1)[2]
```

Out[47]: -26

```
In [48]: # Sum of some elements of array  
np.sum(A[1:, 1:])
```

Out[48]: 40

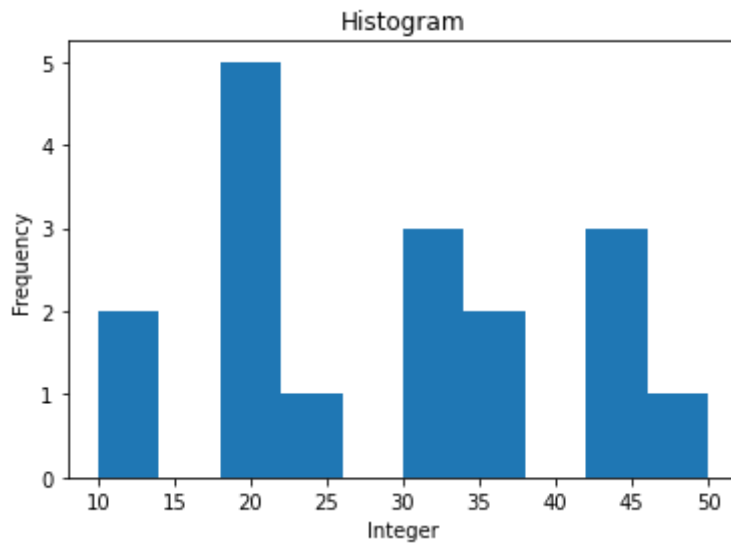
```
In [49]: # Sum of all elements of array  
sumA = np.sum(A)  
sumB = np.sum(B)  
print('sumA = ', sumA, ', sumB = ', sumB)
```

sumA = 56 , sumB = 110

8

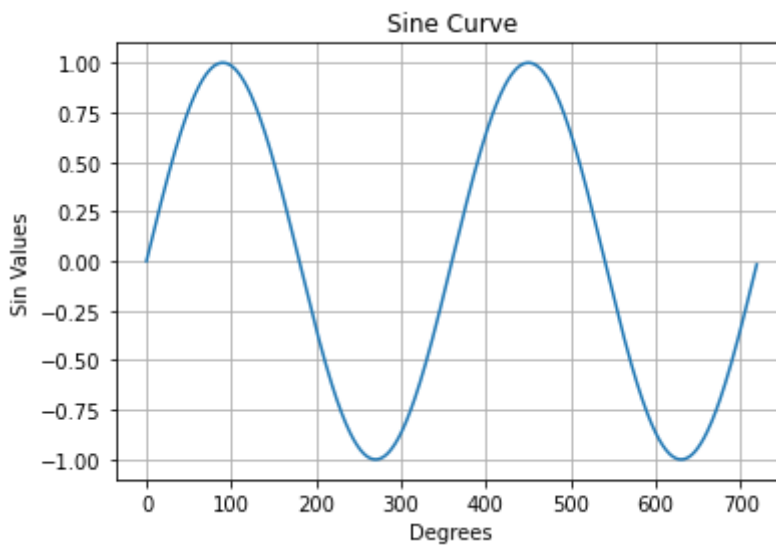
Create various type of plots/charts like histograms, plot based on sine/cosine function based on data from a matrix. Further label different axes in a plot and data in a plot.

```
In [50]: import numpy as np  
import matplotlib.pyplot as plt  
  
# Histogram  
list = np.array([20, 45, 45, 35, 30, 10, 30, 20, 20, 50, 30, 20, 20, 10, 45])  
plt.hist(list)  
plt.xlabel('Integer')  
plt.ylabel('Frequency')  
plt.title('Histogram')  
plt.show()
```

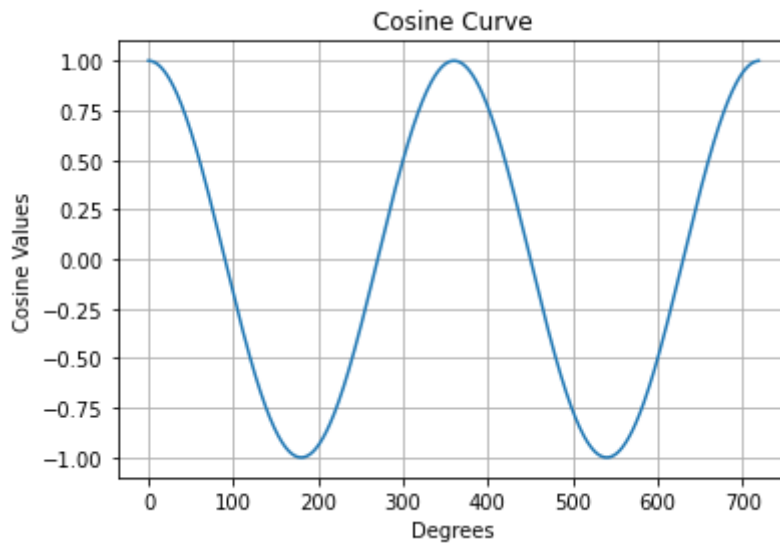



```
In [51]: import math

# Sine curve
degrees = range(0 , 720)
sinValues = [math.sin(math.radians(i)) for i in degrees]
plt.plot(sinValues)
plt.xlabel('Degrees')
plt.ylabel('Sin Values')
plt.title('Sine Curve')
plt.grid()
plt.show()
```



```
In [52]: # Cosine curve
degrees = range(0 , 720)
sinValues = [math.cos(math.radians(i)) for i in degrees]
plt.plot(sinValues)
plt.xlabel('Degrees')
plt.ylabel('Cosine Values')
plt.title('Cosine Curve')
plt.grid()
plt.show()
```



9

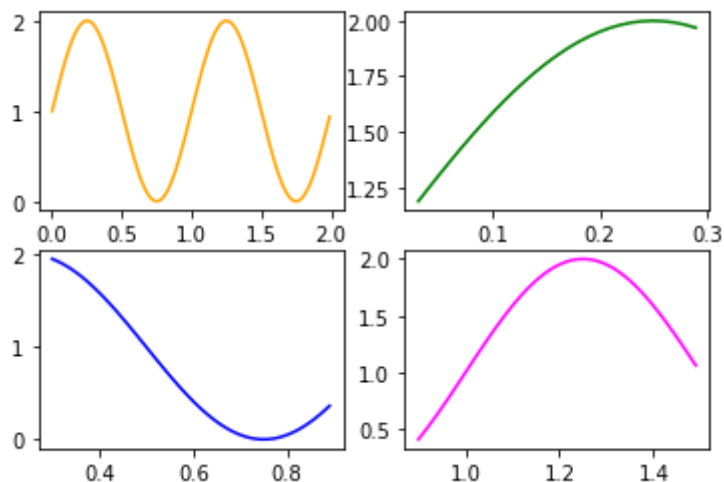
Generate different subplots from a given plot and color plot data.

```
In [53]: import matplotlib.pyplot as plt
import numpy as np

# Data for plotting
x = np.arange(0.0, 2.0, 0.01)
y = 1 + np.sin(2 * np.pi * x)

# Creating 6 subplots and unpacking the output array immediately
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2)
ax1.plot(x, y, color="orange")
ax2.plot(x[3:30], y[3:30], color="green")
ax3.plot(x[30:90], y[30:90], color="blue")
ax4.plot(x[90:150], y[90:150], color="magenta")
```

```
Out[53]: [<matplotlib.lines.Line2D at 0x1d7944a4ca0>]
```



10

Use conditional statements and different type of loops based on simple example/s.

```
In [54]: #if - elif - else
grade = None
marks = 90
if marks >= 95:
    grade = 'A+'
elif marks >= 90:
    grade = 'A'
elif marks >= 80:
    grade = 'B'
elif marks >= 75:
    grade = 'C'
elif marks >= 65:
    grade = 'D'
else:
    grade = 'F'
grade
```

Out[54]: 'A'

```
In [55]: # while Loop
i = 1
while i < 6:
    print(i)
    if i == 3:
        break
    i += 1
```

1
2
3

```
In [56]: # for loop
fruits = ["apple", "cherry", "banana"]
for x in fruits:
    print(x)
    if x == "banana":
        break
```

apple
cherry
banana

11

Perform vectorized implementation of simple matrix operation like finding the transpose of a matrix, adding, subtracting or multiplying two matrices.

```
In [57]: import numpy as np
A = np.array([[3, 6, 9], [5, -10, 15], [-7, 14, 21]])
B = np.array([[9, -18, 27], [11, 22, 33], [13, -26, 39]])
print("A = \n", A, "\nB = \n", B)
```

```
A =  
[[ 3  6  9]  
 [ 5 -10 15]  
 [-7 14 21]]  
B =  
[[ 9 -18 27]  
 [11 22 33]  
 [13 -26 39]]
```

```
In [58]: # Addition  
A + B
```

```
Out[58]: array([[ 12, -12,  36],  
                [ 16,  12,  48],  
                [  6, -12,  60]])
```

```
In [59]: # Subtraction  
A - B
```

```
Out[59]: array([[ -6,  24, -18],  
                [ -6, -32, -18],  
                [-20,  40, -18]])
```

```
In [60]: # Multiplication  
A @ B
```

```
Out[60]: array([[ 210, -156,  630],  
                [ 130, -700,  390],  
                [ 364, -112, 1092]])
```

```
In [61]: # Transpose  
print("A' = \n", np.transpose(A),  
      "\nB' = \n", np.transpose(B))
```

```
A' =  
[[ 3  5 -7]  
 [ 6 -10 14]  
 [ 9 15 21]]  
B' =  
[[ 9 11 13]  
 [-18 22 -26]  
 [ 27 33 39]]
```