#Introduction to NumPy

NumPy (Numerical Python) is a powerful, open-source library in Python used for numerical computations. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently. NumPy is fundamental for scientific computing in Python and serves as the foundation for many other libraries like SciPy, Pandas, and Matplotlib.

Key Features of NumPy N-Dimensional Arrays:

The core feature of NumPy is its ndarray object, which is a multi-dimensional array of elements, typically of the same type. These arrays allow for efficient storage and manipulation of large datasets. Mathematical Functions:

NumPy provides a wide array of mathematical functions for operations such as trigonometry, statistics, linear algebra, and more, all optimized for performance. Broadcasting:

Broadcasting allows NumPy to perform operations on arrays of different shapes without needing to copy data. It simplifies the implementation of mathematical operations. Integration with C/C++ and Fortran:

NumPy can interface with code written in C, C++, or Fortran, allowing for high-performance numerical computation. Linear Algebra:

NumPy includes functions for linear algebra operations such as matrix multiplication, eigenvalue decomposition, and singular value decomposition. Random Number Generation:

The library provides tools for generating random numbers, including random sampling from different probability distributions.

NumPy Exercises

Import NumPy as np

```
import numpy as np
```

Create an array of 10 zeros

```
np.zeros(10)
```

```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

Create an array of 10 ones

```
np.ones(10)
```

```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])


array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

Create an array of 10 fives

```
np.ones(10)*5
```

```
array([5., 5., 5., 5., 5., 5., 5., 5., 5., 5.])


array([5., 5., 5., 5., 5., 5., 5., 5., 5., 5.])
```

Create an array of the integers from 10 to 50

```
np.arange(10,51)
```

```
array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25,
26,
       27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42,
43,
       44, 45, 46, 47, 48, 49, 50])


array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25,
26,
       27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42,
43,
       44, 45, 46, 47, 48, 49, 50])
```

Create an array of all the even integers from 10 to 50

```
num=np.arange(10,51)
li=[]
for i in num:
  if i%2==0:
    li.append(i)
arr1=np.array(li)
arr1
```

```
array([10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40,
42,
       44, 46, 48, 50])


array([10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40,
42,
       44, 46, 48, 50])
```

Create a 3x3 matrix with values ranging from 0 to 8

```
np.arange(0,9).reshape(3,3)
```

```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

Create a 3x3 identity matrix

```
# prompt: Create a 3x3 identity matrix

import numpy as np
np.eye(3,3)
```

```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

Use NumPy to generate a random number between 0 and 1

```
list1=[]
num=np.random.uniform(0,1)
list1.append(num)
np.array(list1)
```

```
array([0.71498617])
```

```
# prompt: Use NumPy to generate a random number between 0 and 1

import numpy as np
random_number = np.random.uniform(0, 1)
random_number
```

```
0.43839366123390955
```

Create the following matrix:

```
import numpy as np

# Create a 1D array with values from 0.01 to 1.0 with a step of 0.01
array_1d = np.arange(0.01, 1.01, 0.01)
```

```
# Reshape the 1D array into a 2D array with 10 rows and 10 columns
array_2d = array_1d.reshape(10, 10)

print(array_2d)
```

```
[[0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.1 ]
 [0.11 0.12 0.13 0.14 0.15 0.16 0.17 0.18 0.19 0.2 ]
 [0.21 0.22 0.23 0.24 0.25 0.26 0.27 0.28 0.29 0.3 ]
 [0.31 0.32 0.33 0.34 0.35 0.36 0.37 0.38 0.39 0.4 ]
 [0.41 0.42 0.43 0.44 0.45 0.46 0.47 0.48 0.49 0.5 ]
 [0.51 0.52 0.53 0.54 0.55 0.56 0.57 0.58 0.59 0.6 ]
 [0.61 0.62 0.63 0.64 0.65 0.66 0.67 0.68 0.69 0.7 ]
 [0.71 0.72 0.73 0.74 0.75 0.76 0.77 0.78 0.79 0.8 ]
 [0.81 0.82 0.83 0.84 0.85 0.86 0.87 0.88 0.89 0.9 ]
 [0.91 0.92 0.93 0.94 0.95 0.96 0.97 0.98 0.99 1.  ]]
```

Create an array of 20 linearly spaced points between 0 and 1:

(Hint: Use linspace function)

```
np.linspace(0,1,20)
```

```
array([0.        , 0.05263158, 0.10526316, 0.15789474, 0.21052632,
       0.26315789, 0.31578947, 0.36842105, 0.42105263, 0.47368421,
       0.52631579, 0.57894737, 0.63157895, 0.68421053, 0.73684211,
       0.78947368, 0.84210526, 0.89473684, 0.94736842, 1.        ])
```

```
array([ 0.        ,  0.05263158,  0.10526316,  0.15789474,
0.21052632,
        0.26315789,  0.31578947,  0.36842105,  0.42105263,
0.47368421,
        0.52631579,  0.57894737,  0.63157895,  0.68421053,
0.73684211,
        0.78947368,  0.84210526,  0.89473684,  0.94736842,
1.        ])
```

Numpy Indexing and Selection

```
mat = np.arange(1,26).reshape(5,5)
mat
```

```
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10],
       [11, 12, 13, 14, 15],
       [16, 17, 18, 19, 20],
       [21, 22, 23, 24, 25]])
```

You are given this matrix named mat. Write some code to get the outputs accordingly in the cells given below

```
#Enter your code here
```

```
#Enter your code here
mat[3,4]
```

```
20
```

```
20
```

```
#Enter your code here
import numpy as np

# Original array
array = np.array([[ 1,  2,  3,  4,  5],
                  [ 6,  7,  8,  9, 10],
                  [11, 12, 13, 14, 15],
                  [16, 17, 18, 19, 20],
                  [21, 22, 23, 24, 25]])

# Extracting the subarray
array[0:3, 1:2]  # Rows 0 to 2, Column 1
```

```
array([[ 2],
       [ 7],
       [12]])
```

```
array([[ 2],
       [ 7],
       [12]])
```

```
#Enter your code here
import numpy as np

# Original array
array = np.array([[ 1,  2,  3,  4,  5],
                  [ 6,  7,  8,  9, 10],
                  [11, 12, 13, 14, 15],
                  [16, 17, 18, 19, 20],
                  [21, 22, 23, 24, 25]])

# Extracting the subarray
array[4]
```

```
array([21, 22, 23, 24, 25])
```

```
#Enter your code here

import numpy as np

# Original array
array = np.array([[ 1,  2,  3,  4,  5],
                  [ 6,  7,  8,  9, 10],
                  [11, 12, 13, 14, 15],
                  [16, 17, 18, 19, 20],
                  [21, 22, 23, 24, 25]])

# Extracting the subarray
array[3:5]
```

```
array([[16, 17, 18, 19, 20],
       [21, 22, 23, 24, 25]])
```

Get the sum of all the values in mat

```
import numpy as np

# Original array
array = np.array([[ 1,  2,  3,  4,  5],
                  [ 6,  7,  8,  9, 10],
                  [11, 12, 13, 14, 15],
                  [16, 17, 18, 19, 20],
                  [21, 22, 23, 24, 25]])

# Extracting the subarray
array.sum()
```

```
325
```

```
325
```

Get the standard deviation of the values in mat

```
import numpy as np

# Original array
array = np.array([[ 1,  2,  3,  4,  5],
                  [ 6,  7,  8,  9, 10],
                  [11, 12, 13, 14, 15],
                  [16, 17, 18, 19, 20],
                  [21, 22, 23, 24, 25]])

# Step 1: Calculate the mean
total_sum = np.sum(array)
num_elements = array.size
```

```
mean = total_sum / num_elements
print("Mean of the entire array:", mean)

# Step 2: Calculate the variance
squared_diff = (array - mean) ** 2
variance = np.sum(squared_diff) / num_elements
print("Variance of the entire array:", variance)

# Step 3: Calculate the standard deviation
std_dev = np.sqrt(variance)
print("Standard Deviation of the entire array:", std_dev)
```

```
Mean of the entire array: 13.0
Variance of the entire array: 52.0
Standard Deviation of the entire array: 7.211102550927978
```

```
7.211102550927978
```

Get the sum of all the columns in mat

```
import numpy as np

# Original array
array = np.array([[ 1,  2,  3,  4,  5],
                  [ 6,  7,  8,  9, 10],
                  [11, 12, 13, 14, 15],
                  [16, 17, 18, 19, 20],
                  [21, 22, 23, 24, 25]])

# Calculate the sum of all columns
column_sums = np.sum(array, axis=0)
print("Sum of all columns:", column_sums)
```

```
Sum of all columns: [55 60 65 70 75]
```

```
array([55, 60, 65, 70, 75])
```