Working with Numpy library

```
import numpy as np
```

Numpy Array Creation

```
list1 = [2,4,6,8,10]
arr1 = np.array(list1)

print(arr1)
```

```
[ 2  4  6  8 10]
```

Create an Array Using np.zeros()

The np.zeros() function allows us to create an array filled with all zeros

```
array1 = np.zeros(5)
print(array1)
```

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

Create an Array With np.arange()

The np.arange() function returns an array with values within a specified interval.

```
# Create a array from with the values from 0 to 4
array2 = np.arange(5)
print(array2)

# Create an array with values from 1 to 8 with a step of 2
array3 = np.arange(1,9,2)
print(array3)
```

```
[0 1 2 3 4]
[1 3 5 7]
```

NumPy N-D Array Creation

NumPy is not restricted to 1-D arrays, it can have arrays of multiple dimensions, also known as N-dimensional arrays or ndarrays.

```
# Create a 2-D NumPy Array

array1 = np.array([[1,2,3,4],
                  [5,6,7,8]])
print(array1)
```

```
[[1 2 3 4]
 [5 6 7 8]]
```

```
# Create a 3-D NumPy Array

array2 = 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]]])
print(array2)
```

```
[[[ 1  2  3  4]
  [ 5  6  7  8]
  [ 9 10 11 12]]

 [[13 14 15 16]
  [17 18 19 20]
  [21 22 23 24]]]
```

Array Indexing

Array indexing is the same as accessing an array element.

You can access an array element by referring to its index number.

The indexes in NumPy arrays start with 0, meaning that the first element has index 0, and the second has index 1 etc.

```
ar =  np.array(
    [
        [1,3],
        [5,6]
    ]
)
print(ar[1][1])
```

```
6
```

Array Slicing

Slicing arrays Slicing in python means taking elements from one given index to another given index.

We pass slice instead of index like this: [start:end].

We can also define the step, like this: [start:end:step].

If we don't pass start its considered 0

If we don't pass end its considered length of array in that dimension

If we don't pass step its considered 1

```
ar1 = np.array([10,2,5,4,8,6,7,9,5,22,66,7])
print(ar1[0:8])
```

```
[10  2  5  4  8  6  7  9]
```

```
ar1 = np.array([10,2,5,4,8,6,7,9,5,22,66,7])
print(ar1[:5]) # Starting to 5 element
print(ar1[2:]) # Starting from 2 to last element
print(ar1[0:8:2]) # Starting from 0 to 8 elemement with 2 gaps
```

```
[10  2  5  4  8]
[ 5  4  8  6  7  9  5 22 66  7]
[10  5  8  7]
```

N-D Array

```
ar = np.array([[2,4,5,6,],[3,6,9,1]])
print(ar.shape)
print(ar.size)
print(ar.itemsize)
print(ar.sum())
print(ar.mean())
```

```
(2, 4)
8
4
36
4.5
```

Data Types in NumPy

strings - used to represent text data, the text is given under quote marks. e.g. "ABCD"

integer - used to represent integer numbers. e.g. -1, -2, -3

float - used to represent real numbers. e.g. 1.2, 42.42

boolean - used to represent True or False.

complex - used to represent complex numbers. e.g. 1.0 + 2.0j, 1.5 + 2.5j

Note:Below is a list of all data types in NumPy and the characters used to represent them.

i - integer

b - boolean

u - unsigned integer

f - float

c - complex float

m - timedelta

M - datetime

O - object

S - string

U - unicode string

V - fixed chunk of memory for other type (void)

```
arr = np.array([1,2,3,4])
print(arr.dtype)
```

```
int32
```

```
arr = np.array(['apple', 'banana', 'mango'],dtype=np.str_)

print(arr.dtype)
```

```
<U6
```

```
arr = np.array([1,2,3,4],dtype='int64')
print(arr.dtype)
```

```
int64
```

```
arr = np.array([1, 2, 3, 4], dtype='S')

print(arr)
print(arr.dtype)
```

```
[b'1' b'2' b'3' b'4']
|S1
```

```
# timedelta
'''
timedelta is a data type in NumPy that represents the difference
between two dates or times.
'''
date1 = np.datetime64('2025-01-24')
date2 = np.datetime64('2025-01-12')
time_diff = date1 - date2
print(time_diff)
```

```
12 days

# datetime
'''
datetime is a data type in NumPy that represents a specofic date and
time.
'''
dt = np.datetime64('2024-01-24 12:26:01')
print(dt)
print(type(dt))

2024-01-24T12:26:01
<class 'numpy.datetime64'>
```

Data Type Conversion

The astype() function creates a copy of the array, and allows you to specify the data type as a parameter.

```
arr = np.array([1.1,2.1,3.1,4.1])
new_arr = arr.astype(int)
print(new_arr)
print(arr.dtype)
print(new_arr.dtype)

[1 2 3 4]
float64
int32
```

Array Attributes

In NumPy, attributes are properties of NumPy arrays that provide information about the array's shape, size, data type, dimension, and so on.

ndim returns number of dimension of the array

size returns number of elements in the array

dtype returns data type of elements in the array

shape returns the size of the array in each dimension.

itemsize returns the size (in bytes) of each elements in the array

data returns the buffer containing actual elements of the array in memory

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

1
```

```
array1 = np.array([[2,4,6],[1,3,5]])
print(array1.ndim)
```

```
2
```

```
array1 = np.array([[2,4,6],[1,3,5]])
print(array1.size)
```

```
6
```

```
array1 = np.array([[2,4,6],[1,3,5]])
print(array1.shape)
```

```
(2, 3)
```

```
arr = np.array([1,2,3,4,5,6])
array1 = np.array([2.1,4.2,6.4])
print(arr.dtype)
print(array1.dtype)
```

```
int32
float64
```

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

```
4
```

```
arr = np.array([1,2,3,4,5,6])
print(arr.data)
array1 = np.array([2.1,4.2,6.4])
print(array1.data)
```

```
<memory at 0x000002258E5D0580>
<memory at 0x000002258E5D0580>
```

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

```
(12, 4)
```

Array Methods in NumPy

In NumPy, methods are functions applied to NumPy arrays that allow for manipulation, computation, and analysis of array data.

reshape(): Reshapes the array without changing its data.

flatten(): Converts a multi-dimensional array into a one-dimensional array.

transpose(): Returns a transposed version of the array (rows become columns and vice versa).

sum(): Calculates the sum of all elements in the array or along a specified axis.

mean(): Computes the mean (average) of the array elements.

argmax(): Returns the index of the maximum element in the array.

argmin(): Returns the index of the minimum element in the array.

sort(): Sorts the elements of the array.

unique(): Finds unique elements of the array.

concatenate(): Joins two or more arrays along a specified axis.

```
ar = np.array([10,20,30,40,50,60,70,80])
```

```
print(ar.reshape(2,4))
```

```
[[10 20 30 40]
 [50 60 70 80]]
```

```
print(ar.sum())
```

```
360
```

```
array = np.array([[1, 2, 3],
                  [4, 5, 6],
                  [7, 8, 9]])
print(array.sum(axis=1))
```

```
[ 6 15 24]
```

```
array = np.array([[1, 2, 3],
                  [4, 5, 6],
                  [7, 8, 9]])
print(array.min(axis=0))
```

```
[1 2 3]
```

```
array = np.array([[1, 2, 3],
                  [4, 55, 6],
                  [7, 8, 9]])
print(array.argmax())
```

```
4
```