Numpy

Numpy is a fundamental library for scientific computation in python.it provides support for arrays and matrices,along with a collection of mathematical functions to operate on these data structures.

```
!pip install numpy
```

```
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (1.26.4)
```

```
import numpy as np
```

```
## create array using numpy
## create a 1d array
arr1=np.array([1,2,3,4,5])
print(arr1)
print(type(arr1))
print(arr1.shape)# shape of array (5,)comma blank means a single dimesnion
```

```
[1 2 3 4 5]
<class 'numpy.ndarray'>
(5,)
```

```
# 1 d array
arr2=np.array([1,2,3,4,5])
arr2.reshape(1,5)# 1 row 5 columns 5 elements so 5 columns
```

```
array([[1, 2, 3, 4, 5]])
```

```
arr2=np.array([[1,2,3,4,5]])
arr2.shape
```

```
(1, 5)
```

```
# 2 d array
arr2=np.array([[1,2,3,4,5],[2,3,4,5,6]])
print(arr2)
print(arr2.shape)# 2 rows 5 columns
```

```
[[1 2 3 4 5]
 [2 3 4 5 6]]
(2, 5)
```

```
# range in numpy is arange
arr3=np.arange(0,10,2)
arr3
```

```
array([0, 2, 4, 6, 8])
```

```
arr3=np.arange(0,10,2).reshape(5,1)
arr3
```

```
array([[0],
       [2],
       [4],
       [6],
       [8]])
```

```
# ones all elements will be ones
a=np.ones(3,4)
a
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-16-b7dfb3cb7d55> in <cell line: 0>()
      1 # ones all elements will be ones
----> 2 a=np.ones(3,4)
      3 a

/usr/local/lib/python3.11/dist-packages/numpy/core/numeric.py in ones(shape, dtype, order, like)
    189         return _ones_with_like(like, shape, dtype=dtype, order=order)
    190 
--> 191     a = empty(shape, dtype, order)
    192     multiarray.copyto(a, 1, casting='unsafe')
    193     return a

TypeError: Cannot interpret '4' as a data type
```

Next steps: Explain error

```
a=np.ones((3,4))# double parenthsis are important
a
```

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

```
a=np.zeros((3,4))# double parenthsis are important
a
```

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

```
## identity matrix eye()
## identity matrix all the diagonal elements are one (1)
a=np.eye(3,3)# 3 rows 3 columns
print(a)
print(a.ndim)
```

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
2
```

ndim- number of dimensions ,shape of array .shape,size of array .size,dtype -data type

```
# Attributes of numpy array
arr=np.array([[1,2,3],[4,5,6]])
print("Array:\n",arr)
print("Shape:",arr.shape)
print("Number of dimensions:",arr.ndim)
print("Size(number of elemnets):",arr.size)
print("Data type:",arr.dtype)# ouput may based on platform
print("Item size (in bytes):",arr.itemsize)#output may vary
```

```
Array:
 [[1 2 3]
 [4 5 6]]
Shape: (2, 3)
Number of dimensions: 2
Size(number of elemnets): 6
Data type: int64
Item size (in bytes): 8
```

```
# Numpy vectorized Operation
arr1=np.array([1,2,3,4,5])
arr2=np.array([10,20,30,40,50])
# Element wise addition
print("Addition:",arr1+arr2)
# Element wise subtraction
print("subtraction",arr1-arr2)
# ELement wise multiplication
print("Multiplication:",arr1*arr2)
```

```
# Element wise division
print("Division:",arr1/arr2)
```

```
Addition: [11 22 33 44 55]
subtraction [ -9 -18 -27 -36 -45]
Multiplication: [ 10  40  90 160 250]
Division: [0.1 0.1 0.1 0.1 0.1]
```

```
## Universal Function
arr=np.array([2,3,4,5,6])
## square root
print(np.sqrt(arr))
## exponential
print(np.exp(arr))# 2.718 - exp value **2,.....
## Sine
print(np.sin(arr))
## natural log
print(np.log)
```

```
[1.41421356 1.73205081 2.         2.23606798 2.44948974]
[  7.3890561   20.08553692  54.59815003 148.4131591  403.42879349]
[ 0.90929743  0.14112001 -0.7568025  -0.95892427 -0.2794155 ]
<ufunc 'log'>
```

```
# slicing and indexing
arr=np.array([[1,2,3,4],[5,6,7,8],[9,10,11,12]])
print("array: \n",arr)
print(arr.ndim)
```

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

```
# how do i retrieve specific elements
print(arr[0])
# i want 1 specific element
print(arr[0][0])
# i want 7,8
print(arr[1][2:])
# 3,4,7,8
print(arr[0:2,2:])
# 6,7,10,11
print(arr[1:,1:3])
```

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

```
# modify array elements
arr[0,0]=100
print(arr)
```

```
[[100   2   3   4]
 [  5   6   7   8]
 [  9  10  11  12]]
```

```
arr[1:]=100
print(arr)
```

```
[[100   2   3   4]
 [100 100 100 100]
 [100 100 100 100]]
```

```
## statistical concepts -- normalization
## to have a mean of 0 and standard devation of 1
## x-mean/std
```

```
data=np.array([1,2,3,4,5])
mean=np.mean(data)
```

```
std_dev=np.std(data)
normalized_data=(data-mean)/std_dev
print("Normalized Data: ",normalized_data)
```

```
Normalized Data:  [-1.41421356 -0.70710678  0.          0.70710678  1.41421356]
```

```
# lets see some statistical concepts
data=np.array([1,2,3,4,5,6,7,8,9,10])
print("Mean:",np.mean(data))
print("Median:",np.median(data))
print("Standard Deviation:",np.std(data))
print("Variance:",np.var(data))
```

```
Mean: 5.5
Median: 5.5
Standard Deviation: 2.8722813232690143
Variance: 8.25
```

```
# logical operations
data=np.array([1,2,3,4,5,6,7,8,9,10])
data>5
```

```
array([False, False, False, False, False,  True,  True,  True,  True,
        True])
```

```
data=np.array([1,2,3,4,5,6,7,8,9,10])
data[data>5]
```

```
array([ 6,  7,  8,  9, 10])
```

```
data=np.array([1,2,3,4,5,6,7,8,9,10])
data[(data>=5) & (data<=8)]
```

```
array([5, 6, 7, 8])
```

```
data=np.array([1,2,3,4,5,6,7,8,9,10])
data[(data>=5) | (data<=8)]
```

```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```