USES OF NUMPY

The numpy can be uses the below operations: 1.Arithmetic operations 2.Statistical operations 3.Bitwise operations 4.Copying & Viewing Arrays 5.Stacking operations 6.Matrix operations 7.Linear algebra 8.Boardcasting 9.Mathematical operations 10.Searching,Sorting & Counting

1.Array Creation Function

In [9]:

```
import numpy as np
```

In [10]:

```
a=np.array([1,2,3,4,5,6,7])    # here we crating the array from the list
print("Array a:",a)
```

```
Array a: [1 2 3 4 5 6 7]
```

In [11]:

```
b=np.arange(0,10,2)
print("Array b:",b)
```

```
Array b: [0 2 4 6 8]
```

In [12]:

```
c=np.zeros((2,3))      # 2x3 here (2- is the row  and 3- means col)
print("Array c is:\n",c)  # here the zeros()- print 2x3 zeros Matrixs
```

```
Array c is:
 [[0. 0. 0.]
 [0. 0. 0.]]
```

In [13]:

```
d=np.ones((2,4))    # 2x4  here (the ones()-prints 2x4 ones matrix
print("Array d:\n",d)
```

```
Array d:
 [[1. 1. 1. 1.]
 [1. 1. 1. 1.]]
```

In [14]:

```
f=np.eye(4)     # here we  creating identity matrix
print("Identity Matrix f:\n",f)    # here 4x4 identity matrix
```

```
Identity Matrix f:
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
```

Array Manipulation Function

In [35]:

```
# Reshape an array
a1=np.array([1,2,3])
reshaped=np.reshape(a1,(1,3))  # Reshape to 1x3
print("Reshaped array:",reshaped)
```

```
Reshaped array: [[1 2 3]]
```

In [36]:

```
# Flatten an array
f1=np.array([[1,2],[3,4]])    #Flatten to 1d array
```

```
flattened = np.ravel(f1)
print("Flattened array:",flattened)
```

```
Flattened array: [1 2 3 4]
```

In [37]:
```
#Transposed an array
e1=np.array([[1,2], [3,4]])     #Transpose the array
transposed =np.transpose(e1)
print("Transposed array:\n",transposed)
```

```
Transposed array:
 [[1 3]
 [2 4]]
```

In [38]:
```
# stack arrays vertically
a2=np.array([1,2])
b2=np.array([3,4])
stacked=np.vstack([a2,b2])   # stack a and b vertically
print("Stacked arrays:\n",stacked)
```

```
Stacked arrays:
 [[1 2]
 [3 4]]
```

Mathematical Functions

In [39]:
```
g=np.array([1,2,3,4])
added=np.add(g,2)      # add 2 to each element(1+2,2+2,3+2,4+2)
print("Added 2 to g:",added)
```

```
Added 2 to g: [3 4 5 6]
```

In [40]:
```
squared=np.power(g,2)      #squre each element
print("Squared g:",squared)
```

```
Squared g: [ 1  4  9 16]
```

In [41]:
```
sqrt_val=np.sqrt(g)  #sqrt-root of each element
print("Square root of g:",sqrt_val)
```

```
Square root of g: [1.         1.41421356 1.73205081 2.        ]
```

In [42]:
```
print(a1)
print(a)
```

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

In [50]:
```
a2 = np.array([1, 2, 3])
dot_product = np.dot(a2, g)  # Dot product of a and g
print("Dot product of a and g:", dot_product)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[50], line 2
      1 a2 = np.array([1, 2, 3])
----> 2 dot_product = np.dot(a2, g)  # Dot product of a and g
      3 print("Dot product of a and g:", dot_product)

ValueError: shapes (3,) and (4,) not aligned: 3 (dim 0) != 4 (dim 0)
```

In [51]:
```
a3 = np.array([1, 2, 3])
dot_product = np.dot(a1, a)  # Dot product of a and g
```

```
print("Dot product of a1 and a:", dot_product)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[51], line 2
      1 a3 = np.array([1, 2, 3])
----> 2 dot_product = np.dot(a1, a)  # Dot product of a and g
      3 print("Dot product of a1 and a:", dot_product)

ValueError: shapes (3,) and (7,) not aligned: 3 (dim 0) != 7 (dim 0)
```

Statistical functions

In [ ]:

```
#here in This we Learn the mean, mean minimum value and mean maximum value
```

In [52]:

```
s = np.array([1, 2, 3, 4])
mean = np.mean(s)
print("Mean of s:", mean)
```

```
Mean of s: 2.5
```

In [53]:

```
std_dev = np.std(s)     #here we are finding the std()- standerd deviation
print("Standard deviation of s:", std_dev)
```

```
Standard deviation of s: 1.118033988749895
```

In [ ]:

```
minimum = np.min(s)       #here we find the min()- means finding the minimum val
print("Min of s:", minimum)
```

In [54]:

```
maximum = np.max(s)    #here we find the max()- means finding the maximum value
print("Max of s:", maximum)
```

```
Max of s: 4
```

liner Algebra functions

In [55]:

```
matrix=np.array([[1,2],[3,4]])
```

In [56]:

```
matrix
```

Out[56]:

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

Random Sampling function

In [57]:

```
random_vals = np.random.rand(3)  # Array of 3 random values btw 0 and 1
print("Random values:", random_vals)
```

```
Random values: [0.34179566 0.11963383 0.30313243]
```

In [58]:

```
# Set seed for reproducibility
np.random.seed(0)

# Generate random values btw 0 and 1
random_vals = np.random.rand(3)  # Array of 3 random values btw 0 and 1
print("Random values:", random_vals)
```

```
Random values: [0.5488135  0.71518937 0.60276338]
```

In [59]:

```
# Generate random integers
rand_ints = np.random.randint(0, 10, size=5)  # here Random integers btw 0 and 1
print("Random integers:", rand_ints)
```

```
Random integers: [3 7 9 3 5]
```

In [60]:

```
# Set seed for reproducibility
np.random.seed(0)

# Generate random integers
rand_ints = np.random.randint(0, 10, size=5)  # Random integers bwt 0 and 10
print("Random integers:", rand_ints)
```

```
Random integers: [5 0 3 3 7]
```

Boolean & logical function

In [62]:

```
logical_test = np.array([True, False, True])
all_true = np.all(logical_test)  # Check if all are True
print("All elements True:", all_true)
```

```
All elements True: False
```

In [63]:

```
logical_test = np.array([True, False, True])
all_true = np.all(logical_test)  # Check if all are True
print("All elements True:", all_true)
```

```
All elements True: False
```

In [64]:

```
logical_test = np.array([False, False, False])
all_true = np.all(logical_test)  # Check if all are True
print("All elements True:", all_true)
```

```
All elements True: False
```

In [65]:

```
any_true = np.any(logical_test)  # Check if any are True
print("Any elements True:", any_true)
```

```
Any elements True: False
```

Set Operation

In [67]:

```
set_a = np.array([1, 2, 3, 4])     #Intersection of two arrays
set_b = np.array([3, 4, 5, 6])
intersection = np.intersect1d(set_a, set_b)
print("Intersection of a and b:", intersection)
```

```
Intersection of a and b: [3 4]
```

In [69]:

```
union = np.union1d(set_a, set_b)   #here we are combimg the array by using union
print("Union of a and b:", union)
```

```
Union of a and b: [1 2 3 4 5 6]
```

Array Attribute functions

In [71]:

```
a = np.array([1, 2, 3,4,5])
shape = a.shape        # Shape of the array
size = a.size          # Number of elements
dimensions = a.ndim    # Number of dimensions
dtype = a.dtype        # Data type of the array
```

```
print("Shape of a:", shape)
print("Size of a:", size)
print("Number of dimensions of a:", dimensions)
print("Data type of a:", dtype)
```

```
Shape of a: (5,)
Size of a: 5
Number of dimensions of a: 1
Data type of a: int32
```

In [ ]: