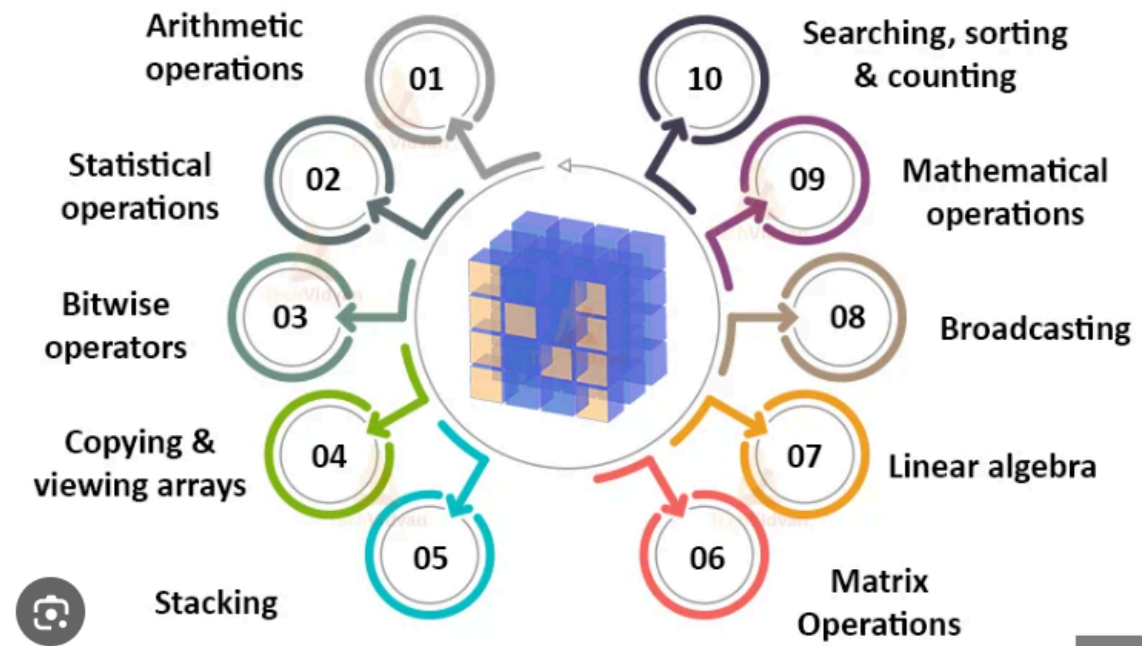# Uses of NumPy

1.Array Creation Functions

```
In [3]: import numpy as np
```

```
In [4]: arr = np.array([10,20,30+40j,"python"])
        print(f"Array arr :{arr} ")

        Array arr :['10' '20' '(30+40j)' 'python']
```

```
In [8]: arr1 = np.arange(30,0,-5)
        print(f"Array arr1 :{arr1} ")

        Array arr1 :[30 25 20 15 10  5]
```

```
In [11]: arr2 = np.linspace(20,25,5)          #evenly spaced 5 values
         print(f"Array arr2 :{arr2} ")

         Array arr2 :[20.   21.25 22.5  23.75 25.  ]
```

```
In [13]: arr3 = np.linspace(-20,5,7)
         print(f"Array arr3 :{arr3} ")

         Array arr3 :[-20.         -15.83333333 -11.66666667  -7.5         -3.33333
         333
             0.83333333   5.        ]
```

```
In [17]:  arr4 = np.eye(6)
          print(f"Identity matrix f : \n {arr4}")
```

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

2. Array Manipulation Functions

```
In [23]:  a = np.array([10,20,50.2,True,"hello",2+1j])
          reshaped = a.reshape(3,2)
          print(f"Reshaped array :\n {reshaped}")
```

```
Reshaped array :
 [['10' '20']
 ['50.2' 'True']
 ['hello' '(2+1j)']]
```

```
In [26]:  reshaped_fortran = a.reshape(3,2,order="F")
          print(f"Fortran array : \n {reshaped_fortran}")
```

```
Fortran array :
 [['10' 'True']
 ['20' 'hello']
 ['50.2' '(2+1j)']]
```

```
In [ ]:  a1 = np.array([[10,20.5,36],[25,"python"],[2.5,3,-6.5]])        #homogenity
         flattened = np.ravel(a1)
         print(f"Flattened array : \n {flattened}")
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[37], line 1
----> 1 a1 = np.array([[10,20.5,36],[25,"python"],[2.5,3,-6.5]])
      2 flattened = np.ravel(a1)
      3 print(f"Flattened array : \n {flattened}")

ValueError: setting an array element with a sequence. The requested array
has an inhomogeneous shape after 1 dimensions. The detected shape was (3,)
+ inhomogeneous part.
```

```
In [43]:  a2 = np.array([[10,20.5,36],[25,"python",True],[2.5,3,-6.5]])
          flattened = np.ravel(a2)
          print(f"Flattened array : \n {flattened}")
```

```
Flattened array :
 ['10' '20.5' '36' '25' 'python' 'True' '2.5' '3' '-6.5']
```

In [40]:
```python
a2
```

Out[40]:
```
array([['10', '20.5', '36'],
       ['25', 'python', 'True'],
       ['2.5', '3', '-6.5']], dtype='<U32')
```

In [ ]:
```python
transpose_arr = a2.transpose()        #rows converted into columns and co
print(transpose_arr)
```

```
[['10' '25' '2.5']
 ['20.5' 'python' '3']
 ['36' 'True' '-6.5']]
```

In [46]:
```python
a3 = np.array([1,2.5,3])
a4 = np.array([2.5,3,-5.5])
stacked = np.vstack([a3,a4])
print(f"Stacked array : \n {stacked}")
```

```
Stacked array :
 [[ 1.   2.5  3. ]
 [ 2.5  3.  -5.5]]
```

3. Mathematical Functions

In [54]:
```python
b = np.array([10,25.5,20,30])
add = np.add(b,-5)
print(add)
```

```
[ 5.  20.5 15.  25. ]
```

In [56]:
```python
power_value = np.power(b,2)
print(power_value)
```

```
[100.   650.25 400.   900.  ]
```

In [57]:
```python
squr_root = np.sqrt(b)
print(squr_root)
```

```
[3.16227766 5.04975247 4.47213595 5.47722558]
```

```python
In [63]: dot1 = np.array([1,2,3,4,5])
         dot2 = np.array([5,6,7,8])
         dot_product = np.dot(dot1,dot2)
         print(f"Dot product of dot1 and dot2 :\n {dot_product}")
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[63], line 3
      1 dot1 = np.array([1,2,3,4,5])
      2 dot2 = np.array([5,6,7,8])
----> 3 dot_product = np.dot(dot1,dot2)
      4 print(f"Dot product of dot1 and dot2 :\n {dot_product}")

ValueError: shapes (5,) and (4,) not aligned: 5 (dim 0) != 4 (dim 0)
```

```python
In [64]: dot1 = np.array([1,2,3,4])
         dot2 = np.array([5,6,7,8])
         dot_product = np.dot(dot1,dot2)
         print(f"Dot product of dot1 and dot2 :\n {dot_product}")
```

```
Dot product of dot1 and dot2 :
 70
```

4. Statistical Functions

```python
In [65]: s = np.array([1,2,3,4])
         mean_value = np.mean(s)
         print(f"mean of s : {mean_value}")
```

```
mean of s : 2.5
```

```python
In [66]: stad_dev = np.std(s)
         print(stad_dev)
```

```
1.118033988749895
```

```python
In [67]: mini_value = np.min(s)
         print(f"minimum value : {mini_value}")
```

```
minimum value : 1
```

```python
In [69]: max_value = np.max(s)
         print(f"maximum value : {max_value}")
```

```
maximum value : 4
```

5. Liner Algebra Functions

```python
In [70]: matrix = np.array([[1,2],[3,4]])
         determinat = np.linalg.det(matrix)
         print(f"Determinat of matrix : {determinat}")
```

```
Determinat of matrix : -2.0000000000000004
```

```python
In [71]: inverse_matrix = np.linalg.inv(matrix)
         print(inverse_matrix)
```

```
[[-2.   1. ]
 [ 1.5 -0.5]]
```

6. Boolean & Logical Functions

```python
In [72]: logical_test = np.array([True,False,True])
         all_true = np.all(logical_test)
         print("All elements are True :",all_true)
```

```
All elements are True : False
```

```python
In [73]: any_true = np.any(logical_test)
         print("Any elements True :", any_true)
```

```
Any elements True : True
```

7. Set Operations

```python
In [74]: set_a = np.array([1,2,3,4])
         set_b = np.array([3,4,5,6])
         np.intersect1d(set_a,set_b)
```

```
Out[74]: array([3, 4])
```

```python
In [ ]: np.union1d(set_a,set_b)
```

8.Array Attribute Functions

```python
In [76]: arr5 = np.array([1,2,3,4])
         print(arr5.shape)
         print(arr5.size)
         print(arr5.ndim)
         print(arr5.dtype)
```

```
(4,)
4
1
int64
```

9. Other Functions

In [77]:
```python
arr5
```

Out[77]: array([1, 2, 3, 4])

In [78]:
```python
copied_array = np.copy(arr5)
print("copied array :",copied_array)
```

copied array : [1 2 3 4]

In [79]:
```python
array_size_in_bytes = a.nbytes
print("Size in bytes:", array_size_in_bytes)
```

Size in bytes: 1536

In [80]:
```python
shared = np.shares_memory(arr5,copied_array)
print("Do arr5 and copied_array share memory?",shared)
```

Do arr5 and copied_array share memory? False