Complete NUMPY DOCUMENTATION

In [22]:
```
image=(r"C:\Users\santo\OneDrive\Desktop\pppp.jpg")
```

In [23]:
```
image
```

Out[23]:
```
'C:\\Users\\santo\\OneDrive\\Desktop\\pppp.jpg'
```

In [24]:
```
import numpy as np
```

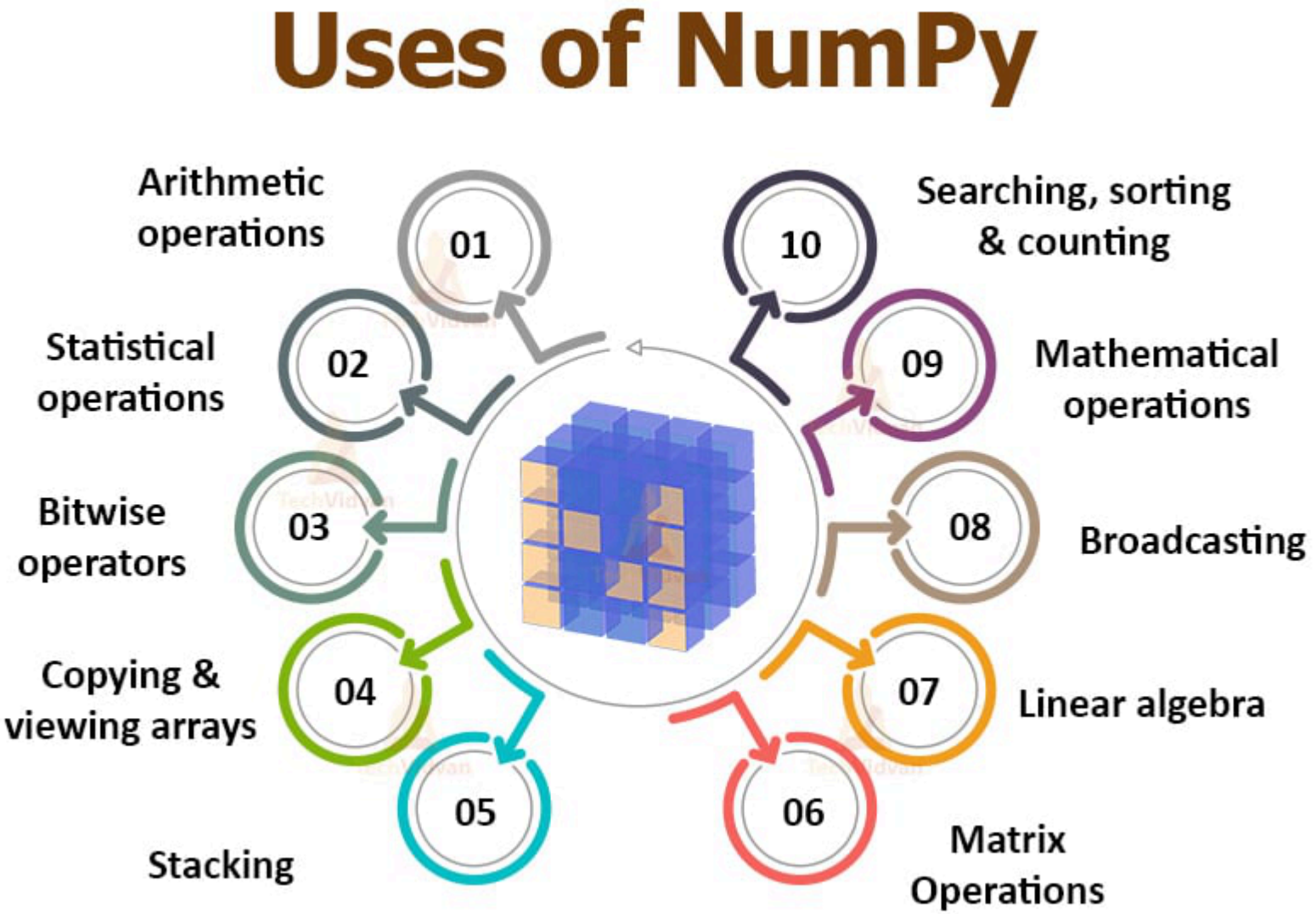
In [25]:
```
import matplotlib.pyplot as plt
```

In [26]:
```
from PIL import Image
```

In [27]:
```
feature_image= Image.open(r"C:\Users\santo\OneDrive\Desktop\pppp.jpg")
```

In [28]:
```
feature_image
```

Out[28]:



1. Array Creation Function

In [29]:
```
import numpy as np
```

In [36]:
```
# Create an array from a list
a = np.array([4, 5,6])
print("Array a:", a)
```

```
Array a: [4 5 6]
```

In [37]:
```
# Create an array with evenly spaced values
b = np.arange(0, 10, 2)  # Values from 0 to 10 with step 2
print("Array b:", b)
```

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

In [38]:
```
# Create an array filled with zeros
d = np.zeros((2, 3),)  # 2x3 array of zeros
print("Array d:\n", d)
```

```
Array d:
 [[0. 0. 0.]
 [0. 0. 0.]]
```

In [39]:
```
# Create an array filled with zeros
d = np.zeros((2, 3),dtype=int)  # 2x3 array of zeros
print("Array d:\n", d)
```

```
Array d:
 [[0 0 0]
 [0 0 0]]
```

In [40]:
```
# Create an array filled with ones
e = np.ones((3, 2))  # 3x2 array of ones
print("Array e:\n", e)
```

```
Array e:
 [[1. 1.]
 [1. 1.]
 [1. 1.]]
```

In [41]:
```
# Create an identity matrix
f = np.eye(4)  # 4x4 identity matrix
```

```
print("Identity matrix f:\n", f)
```

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

2.Array Manipulation Function

In [ ]:
```
# Reshape an array
a1 = np.array([4, 5, 3])
reshaped = np.reshape(a1, (1, 3))  # Reshape to 1x3
print("Reshaped array:", reshaped)
```

In [46]:
```
# Reshape an array
a1 = np.array([1, 2, 3])
reshaped = np.reshape(a1, (3, 1))  # Reshape to 1x3
print("Reshaped array:", reshaped)
```

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

In [47]:
```
# Flatten an array
f1 = np.array([[1, 2], [3, 4], [5, 6], [7, 8]])
flattened = np.ravel(f1)  # Flatten to 1D array
print("Flattened array:", flattened)
```

```
Flattened array: [1 2 3 4 5 6 7 8]
```

In [48]:
```
# Transpose an array
e1 = np.array([[2, 3], [4, 5]])
transposed = np.transpose(e1)  # Transpose the array
print("Transposed array:\n", transposed)
```

```
Transposed array:
 [[2 4]
 [3 5]]
```

In [49]:
```
# Stack arrays vertically
a2 = np.array([2, 3])
b2 = np.array([4, 5])
stacked = np.vstack([a2, b2])  # Stack a and b vertically
print("Stacked arrays:\n", stacked)
```

```
Stacked arrays:
 [[2 3]
 [4 5]]
```

3.Mathematical Function

In [50]:
```
# Add two arrays
g = np.array([4, 5, 6, 7])
added = np.add(g, 2)  # Add 2 to each element
print("Added 2 to g:", added)
```

```
Added 2 to g: [6 7 8 9]
```

In [51]:
```
# Square each element
squared = np.power(g, 2)  # Square each element
print("Squared g:", squared)
```

```
Squared g: [16 25 36 49]
```

In [52]:
```
# Square root of each element
sqrt_val = np.sqrt(g)  # Square root of each element
print("Square root of g:", sqrt_val)
```

```
Square root of g: [2.         2.23606798 2.44948974 2.64575131]
```

In [53]:
```
print(a1)
print(g)
```

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

In [55]:
```
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)
```

```
Dot product of a1 and a: 32
```

4.Stastical Function

In [57]:
```
s = np.array([4,5,6,7])
mean = np.mean(s)
print("Mean of s:", mean)
```

```
Mean of s: 5.5
```

In [58]:
```
# Standard deviation of an array
std_dev = np.std(s)
print("Standard deviation of s:", std_dev)
```

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

In [59]:
```
# Minimum element of an array
minimum = np.min(s)
print("Min of s:", minimum)
```

```
Min of s: 4
```

In [60]:
```
# Minimum element of an array
maximum= np.max(s)
print("Max of s:", maximum)
```

```
Max of s: 7
```

5. Linear Algebra Functions

In [61]:
```
# Create a matrix
matrix = np.array([[4,5], [6,7]])
```

6.Random Sampling Function

In [62]:
```
# Generate random values between 0 and 1
random_vals = np.random.rand(5)  # Array of 5 random values between 0 and 1
print("Random values:", random_vals)
```

```
Random values: [0.93714536 0.96419488 0.78219384 0.4895906  0.96862802]
```

In [63]:
```
# Set seed for reproducibility
np.random.seed(0)

# Generate random values between 0 and 1
random_vals = np.random.rand(5)  # Array of 5 random values between 0 and 1
print("Random values:", random_vals)
```

```
Random values: [0.5488135  0.71518937 0.60276338 0.54488318 0.4236548 ]
```

In [64]:
```
# Generate random integers
rand_ints = np.random.randint(0, 10, size=5)  # Random integers between 0 and 10
print("Random integers:", rand_ints)
```

```
Random integers: [5 2 4 7 6]
```

In [65]:
```
# Set seed for reproducibility
np.random.seed(0)

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

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

7. Boolean & Logical Functions

In [66]:
```
# Check if all elements are True
# all
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 [67]:
```
# Check if all elements are True
# all
logical_test = np.array([True, True, True])
all_true = np.all(logical_test)  # Check if all are True
print("All elements True:", all_true)
```

```
All elements True: True
```

In [68]:
```
# Check if all elements are True
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 [69]:
```
# Check if all elements are True
logical_test = np.array([False, True, False])
all_true = np.all(logical_test)  # Check if all are True
print("All elements True:", all_true)
```

```
All elements True: False
```

In [70]:
```
# Check if any elements are True
# any
any_true = np.any(logical_test)  # Check if any are True
print("Any elements True:", any_true)
```

```
Any elements True: True
```

8. Set Operation

In [71]:
```
# Intersection of two arrays
set_a = np.array([4,5,6,7])
set_b = np.array([6,7,8,9])
intersection = np.intersect1d(set_a, set_b)
print("Intersection of a and b:", intersection)
```

```
Intersection of a and b: [6 7]
```

In [72]:
```
# Union of two arrays
union = np.union1d(set_a, set_b)
print("Union of a and b:", union)
```

```
Union of a and b: [4 5 6 7 8 9]
```

9. Array Attribute Functions

In [73]:
```
# Array attributes
a = np.array([4,5,6])
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: (3,)
Size of a: 3
Number of dimensions of a: 1
Data type of a: int64
```

In [74]:
```
# Array attributes
a = np.array([1, 2, 3])
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: (3,)
Size of a: 3
Number of dimensions of a: 1
Data type of a: int64
```

10. Other Function

In [75]:
```python
# Create a copy of an array
a = np.array([4,5,6])
copied_array = np.copy(a)  # Create a copy of array a
print("Copied array:", copied_array)
```

```
Copied array: [4 5 6]
```

In [76]:
```python
# Size in bytes of an array
array_size_in_bytes = a.nbytes  # Size in bytes
print("Size of a in bytes:", array_size_in_bytes)
```

```
Size of a in bytes: 24
```

In [77]:
```python
# Check if two arrays share memory
shared = np.shares_memory(a, copied_array)  # Check if arrays share memory
print("Do a and copied_array share memory?", shared)
```

```
Do a and copied_array share memory? False
```

In [ ]:
```python

```