# Q1 Create a Python program to demonstrate operations/methods of Numpy library on any set of data or array. ¶

In [1]:
```
#One dimension
import numpy as np
a = np.array([1,2,3])
print (a)
```

```
[1 2 3]
```

In [2]:
```
#More than one dimensions
import numpy as np
a = np.array([[1,2],[3,4]])
print(a)
```

```
[[1 2]
 [3 4]]
```

In [3]:
```
#Using dtype parameters
import numpy as np
a = np.array([1,2,3],dtype = complex)
print (a)
```

```
[1.+0.j 2.+0.j 3.+0.j]
```

In [4]:
```
#Checking the shape of an array
import numpy as np
a = np.array([[1,2],[3,4]])
print (a.shape)
print(a)
```

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

# Reshaping an array

In [5]:
```
#Method 1
import numpy as np
a = np.array([[1,2],[3,4],[5,6]])
a.shape=(2,3)
print(a)
print(a.reshape)
```

```
[[1 2 3]
 [4 5 6]]
<built-in method reshape of numpy.ndarray object at 0x0000023748F6A990>
```

In [6]:
```
#Method 2
a = np.array([[1,2],[3,4],[5,6]])
b=a.reshape(2,3)
print(b)
```

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

In [7]:
```
#Change data type of array from set to tuple
x = np.array([1,2,3])
a=np.asarray(x, dtype = float)
print(a)
```

```
[1. 2. 3.]
```

In [8]:
```
#Method 1
#Prinitng an array from numerical ranges
x = np.arange(5)
print(x)
```

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

In [9]:
```
#method 2
#dtype is set
x = np.arange(5, dtype = float)
print(x)
```

```
[0. 1. 2. 3. 4.]
```

In [10]:
```
#Method 3
#Start and stop parameters are given
x = np.arange(10,20,2)
print(x)
```

```
[10 12 14 16 18]
```

# Indexing and slicing

In [11]:
```
import numpy as np
a = np.arange(1,10)
print(a)
s = slice(2,8,2) # Slicing method 1
print(a[s])
```

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

In [12]:
```
a = np.arange(10)
print(a)
b = a[2:8:2]  #Slicing method 2
print(b)
```

```
[0 1 2 3 4 5 6 7 8 9]
[2 4 6]
```

In [13]:
```
#Slicing elements from starting index
a = np.arange(10)
print(a[2:])
```

```
[2 3 4 5 6 7 8 9]
```

In [14]:
```
#Slicing items between elements
a = np.arange(10)
print(a[2:6])
```

```
[2 3 4 5]
```

In [15]:
```
a = np.array([[1,2,3],[3,4,5],[4,5,6]])
print(a)
```

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

In [16]:
```
#Slice items staring from index
print('Now we will slice the array from the index at a[1:]')
print(a[1:])
```

```
Now we will slice the array from the index at a[1:]
[[3 4 5]
 [4 5 6]]
```

In [17]:
```
#Print only first row
print(a[:1])
```

```
[[1 2 3]]
```

In [18]:
```
#Print only second row
print(a[1:2])
```

```
[[3 4 5]]
```

In [19]:
```
#Items in the second column
print(a[1,...])
```

```
[3 4 5]
```

In [20]:
```
#Items from second column and onwards
print(a[...,1:])
```

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

## Mathematical Functions

In [21]:
```
import numpy as np
arr = np.array([0,30,60,90,120,150,180])
print("\n The sin value of the angles", end = "")
print(np.sin(arr))
print("\n The cosine value of the angles", end = "")
print(np.cos(arr))
print("\n The tangent value of the angles", end = "")
print(np.tan(arr))
```

```
 The sin value of the angles[ 0.         -0.98803162 -0.30481062  0.89399666
0.58061118 -0.71487643
 -0.80115264]

 The cosine value of the angles[ 1.          0.15425145 -0.95241298 -0.448073
62  0.81418097  0.69925081
 -0.59846007]

 The tangent value of the angles[ 0.         -6.4053312   0.32004039 -1.99520
041  0.71312301 -1.02234624
  1.33869021]
```

## Functions to round off the numbers: 1) Round 2) Floor 3) Ceil

In [22]:
```
arr = np.array([12.202, 90.23120, 123.020, 23.202])
print("Printing the original array values: ",end = " ")
print(arr)
print("Array values rounded off to 2 decimal position: ",np.around(arr,2))
```

```
Printing the original array values:  [ 12.202   90.2312 123.02    23.202 ]
Array values rounded off to 2 decimal position:  [ 12.2   90.23 123.02  23.2
]
```

# Floor and ceil functions

In [23]:
```
arr = np.array([12.202, 90.23120, 123.020, 23.202])
print(np.floor(arr))
```

```
[ 12.  90. 123.  23.]
```

In [24]:
```
arr = np.array([12.202, 90.23120, 123.020, 23.202])
print(np.ceil(arr))
```

```
[ 13.  91. 124.  24.]
```

# Statistical Functions

In [25]:
```
a = np.array([[2,10,20],[80,43,31],[22,43,10]])
print("The original array: \n")
print(a)
print("The minimum element among the array: ",np.amin(a))
print("The maximum element among the array: ",np.amax(a))

print("The minimum element among the rows of array: ",np.amin(a,1))
print("The maximum element among the rows of array: ",np.amax(a,1))

print("The minimum element among the columns of array: ",np.amin(a,0))
print("The maximum element among the columns of array: ",np.amax(a,0))

```

```
The original array:

[[ 2 10 20]
 [80 43 31]
 [22 43 10]]
The minimum element among the array:  2
The maximum element among the array:  80
The minimum element among the rows of array:  [ 2 31 10]
The maximum element among the rows of array:  [20 80 43]
The minimum element among the columns of array:  [ 2 10 10]
The maximum element among the columns of array:  [80 43 31]
```

In [26]:
```
a = np.array([[1,2,3],[1,5,6],[7,8,6]])
print("Array: \n",a)
print("\nMedian of array along axis 0: ",np.median(a,0))
print("\nMean of array along axis 0: ",np.mean(a,0))
print("\n Standard deviation of array: ",np.std([1,2,3,4]))
```

```
Array:
 [[1 2 3]
 [1 5 6]
 [7 8 6]]

Median of array along axis 0:  [1. 5. 6.]

Mean of array along axis 0:  [3. 5. 5.]

 Standard deviation of array:  1.118033988749895
```

## Q2 Create a Python program to create data frame using dictionary which contains columns like, ID, Name, Age, & Salary of 15 employees.Perform all statistical operation/methods on to it.

In [27]:
```
import pandas as pd
```

In [28]:
```
# Creating a dictionary with employee data
data = {
    'ID': [101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 11
    'Name': ['John', 'Emma', 'Sophia', 'James', 'Oliver', 'Isabella', 'Li
    'Age': [28, 34, 29, 45, 40, 31, 27, 36, 42, 25, 38, 33, 26, 41, 35],
    'Salary': [50000, 60000, 58000, 72000, 68000, 54000, 51000, 75000, 70
}
print(data)
```

```
{'ID': [101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114,
115], 'Name': ['John', 'Emma', 'Sophia', 'James', 'Oliver', 'Isabella', 'Lia
m', 'Ethan', 'Mason', 'Ella', 'Ava', 'Noah', 'Lucas', 'Mia', 'Amelia'], 'Ag
e': [28, 34, 29, 45, 40, 31, 27, 36, 42, 25, 38, 33, 26, 41, 35], 'Salary':
[50000, 60000, 58000, 72000, 68000, 54000, 51000, 75000, 70000, 52000, 67000,
62000, 49000, 71000, 65000]}
```

In [29]:
```
# Creating a DataFrame from the dictionary
df = pd.DataFrame(data)
df
```

Out[29]:

| | ID | Name | Age | Salary |
|---|---|---|---|---|
| **0** | 101 | John | 28 | 50000 |
| **1** | 102 | Emma | 34 | 60000 |
| **2** | 103 | Sophia | 29 | 58000 |
| **3** | 104 | James | 45 | 72000 |
| **4** | 105 | Oliver | 40 | 68000 |
| **5** | 106 | Isabella | 31 | 54000 |
| **6** | 107 | Liam | 27 | 51000 |
| **7** | 108 | Ethan | 36 | 75000 |
| **8** | 109 | Mason | 42 | 70000 |
| **9** | 110 | Ella | 25 | 52000 |
| **10** | 111 | Ava | 38 | 67000 |
| **11** | 112 | Noah | 33 | 62000 |
| **12** | 113 | Lucas | 26 | 49000 |
| **13** | 114 | Mia | 41 | 71000 |
| **14** | 115 | Amelia | 35 | 65000 |

In [30]:

```
# 1. Descriptive Statistics
print("\n1. Descriptive Statistics of the DataFrame:")
print(df.describe())

# 2. Mean of Age and Salary
print("\n2. Mean Age:", df['Age'].mean())
print("Mean Salary:", df['Salary'].mean())

# 3. Median of Age and Salary
print("\n3. Median Age:", df['Age'].median())
print("Median Salary:", df['Salary'].median())

# 4. Standard Deviation of Age and Salary
print("\n4. Standard Deviation of Age:", df['Age'].std())
print("Standard Deviation of Salary:", df['Salary'].std())

# 5. Minimum and Maximum values of Age and Salary
print("\n5. Minimum Age:", df['Age'].min())
print("Maximum Age:", df['Age'].max())
print("Minimum Salary:", df['Salary'].min())
print("Maximum Salary:", df['Salary'].max())

# 6. Sum of all Salaries
print("\n6. Total Salary of all Employees:", df['Salary'].sum())
```

```
1. Descriptive Statistics of the DataFrame:
               ID        Age        Salary
count   15.000000  15.000000     15.000000
mean   108.000000  34.000000  61600.000000
std      4.472136   6.301927   8862.440812
min    101.000000  25.000000  49000.000000
25%    104.500000  28.500000  53000.000000
50%    108.000000  34.000000  62000.000000
75%    111.500000  39.000000  69000.000000
max    115.000000  45.000000  75000.000000

2. Mean Age: 34.0
Mean Salary: 61600.0

3. Median Age: 34.0
Median Salary: 62000.0

4. Standard Deviation of Age: 6.301927142889365
Standard Deviation of Salary: 8862.440811811222

5. Minimum Age: 25
Maximum Age: 45
Minimum Salary: 49000
Maximum Salary: 75000

6. Total Salary of all Employees: 924000
```

# Q3 Create a Python program to demonstrate how we can perform indexing and slicing operation on Student.csv file.

In [31]:

```
# Importing the pandas library
import pandas as pd

# Load the Student.csv file into a DataFrame
df = pd.read_csv('Student.csv')

# Display the loaded DataFrame
print("Student DataFrame:")
print(df)

# 1. Indexing using loc (based on labels)
print("\n1. Using loc to access rows based on labels (Row 0 and 2, all co
print(df.loc[[0, 2]])  # Access rows with index labels 0 and 2

# 2. Indexing using iloc (based on positions)
print("\n2. Using iloc to access rows based on positions (Row 1 and 3, fi
print(df.iloc[[1, 3], :2])  # Access rows at index positions 1 and 3, firs

# 3. Slicing rows
print("\n3. Slicing rows 1 to 4 (all columns):")
print(df[1:5])  # Slicing rows 1 to 4 (upper limit excluded)

# 4. Slicing columns (by name)
print("\n4. Slicing specific columns ('Name' and 'Marks') for all rows:")
print(df[['Name', 'Marks']])  # Access the 'Name' and 'Marks' columns

# 5. Slicing columns (by position using iloc)
print("\n5. Slicing first two columns for first three rows:")
print(df.iloc[:3, :2])  # First three rows, first two columns

# 6. Conditional Indexing (Filter based on a condition)
print("\n6. Students with Marks greater than 80:")
print(df[df['Marks'] > 80])  # Filter rows where 'Marks' column is greate
```

```
Student DataFrame:
   ID     Name  Age  Marks Grade
0   1    Alice   20     85     A
1   2      Bob   21     90     A
2   3  Charlie   19     88    B+
3   4    David   22     75     B
4   5      Eva   20     95    A+

1. Using loc to access rows based on labels (Row 0 and 2, all columns):
   ID     Name  Age  Marks Grade
0   1    Alice   20     85     A
2   3  Charlie   19     88    B+

2. Using iloc to access rows based on positions (Row 1 and 3, first 2 column
s):
   ID   Name
1   2    Bob
3   4  David

3. Slicing rows 1 to 4 (all columns):
   ID     Name  Age  Marks Grade
1   2      Bob   21     90     A
2   3  Charlie   19     88    B+
3   4    David   22     75     B
4   5      Eva   20     95    A+

4. Slicing specific columns ('Name' and 'Marks') for all rows:
      Name  Marks
0    Alice     85
1      Bob     90
2  Charlie     88
3    David     75
4      Eva     95

5. Slicing first two columns for first three rows:
   ID     Name
0   1    Alice
1   2      Bob
2   3  Charlie

6. Students with Marks greater than 80:
   ID     Name  Age  Marks Grade
0   1    Alice   20     85     A
1   2      Bob   21     90     A
2   3  Charlie   19     88    B+
4   5      Eva   20     95    A+
```