# Data Sciences using Python (05101305)

**Dr. Kamini Solanki, (**Associate Professor)
Parul Institute of Computer Application - BCA

Parul® University

DIGITAL LEARNING CONTENT

# CHAPTER-3

# Mathematical Computing with Python (Numpy)

# Brief About numpy package

**What is numpy?**

- Numpy is inbuilt package of python which is introduce by Travis Oliphant in year 2005.
- It is stand as numeric python and basically use to perform mathematical operation basically on matrix.
- If numpy is not present in your machine than you need to install it using python command prompt.
- For that use below command but make sure you connected with internet at the time of package installation.
- Or else one can install a full version of scipy or anaconda.
- Command to install numpy is
- **$pip install numpy**

# Cont…

**Why numpy?**

- **Using list of tuple, set or dictionary one can change individual value by passing its index value but can not perform mathematical operation on it.**
- **Numpy provide multidimensional array on which one can easily mathematical operation.**
- Numpy is use for very large amount of data, it is also convenient with matrix manipulation and scientific calculation within fraction of second.

# Why numpy cont…

## numpy vs list

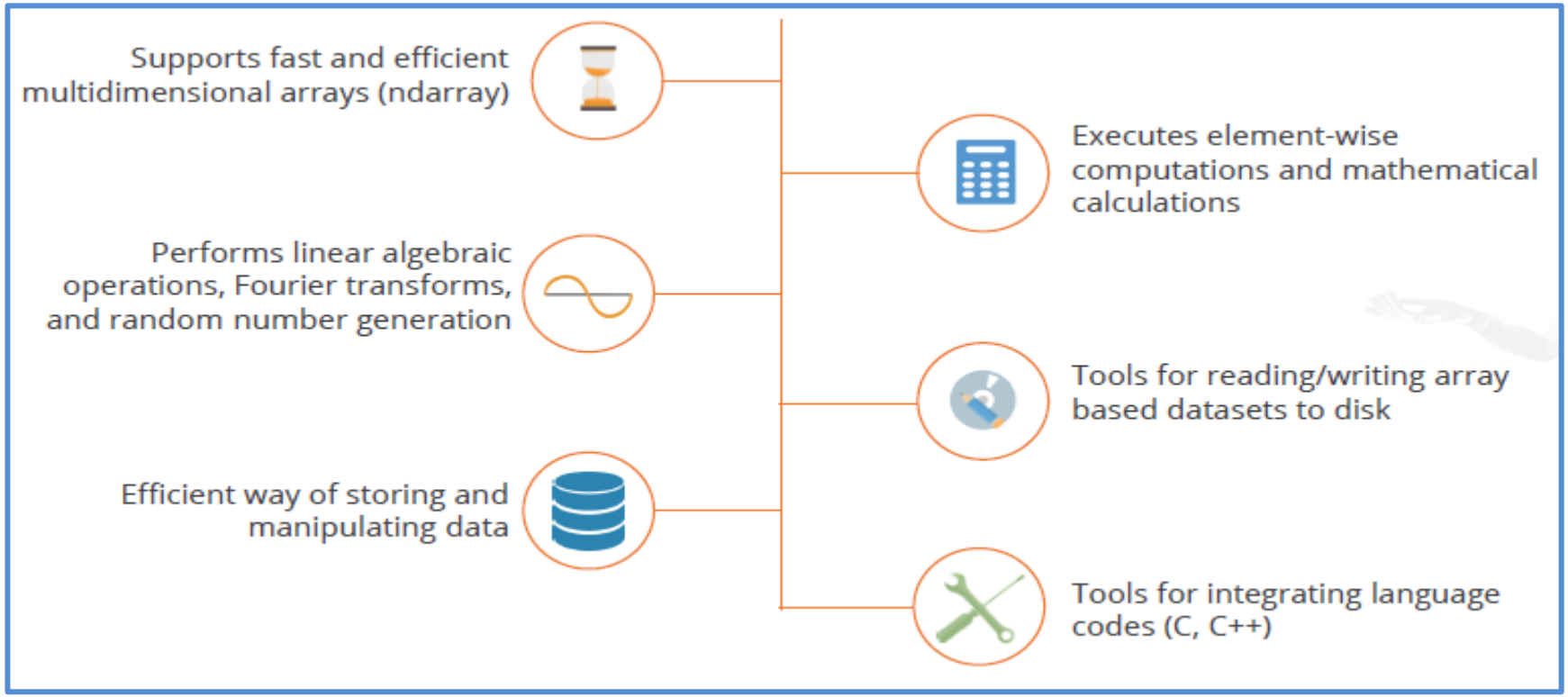
In [1]:
```
#clculate distance based on given speed and time
list_time = [0.20,1,0.50,1.20,1.05]
list_speed = [30,40,10,50,20]
total_distance = list_time * list_speed
print(total_distance)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-1-0464463c80ac> in <module>
      2 list_time = [0.20,1,0.50,1.20,1.05]
      3 list_speed = [30,40,10,50,20]
----> 4 total_distance = list_time * list_speed
      5 print(total_distance)

TypeError: can't multiply sequence by non-int of type 'list'
```

# Why numpy cont…

```
In [4]: print(list_time)
        print(list_speed)
        #converting list in numpy array
        #np.array() method use of numpy package to convert in into array
        import numpy as np
        time = np.array(list_time)
        speed = np.array(list_speed)
        distance = time * speed
        print("Calculated Distance is ")
        print(distance)

        [0.2, 1, 0.5, 1.2, 1.05]
        [30, 40, 10, 50, 20]
        Calculated Distance is
        [ 6. 40.  5. 60. 21.]
```

# Some properties of numpy




As you know that numpy is use to perform mathematical computation

Image source : simplilearn

# Cont…

- Numpy array create using array() and ndarray() method of numpy package.
- It is collection of value of same data type.
- So its known as homogenous.
- Numpy array is multidimensional array one create 1D, 2D, 3D or ND array.
- Is also allow to perform basic add, remove and update operation.
- Widely use to perform mathematical computation based operation.
- It’s easy to access that is why it is fast and efficient.
- It’s easy to transfer data from one algorithm to another algorithm for the data analytics or data science purpose.

# Array can be 1D, 2D or 3D or ND

## One-Dimensional Array

Printed as rows

array([5, 7,9]) ← 1 axis rank 1

Length = 3

| | | |
|---|---|---|
| **5** | **7** | **9** |
| 0 | 1 | 2 |

*x* axis

## Two-Dimensional Array

Printed as matrices (2x3)

array([[ 0, 1, 2],
[ 5, 6, 7]]) ← 2 axes rank 2

Length = 3

*y* axis

| | | |
|---|---|---|
| **0** (0,0) | **1** (0,1) | **2** (0,2) |
| **5** (1,0) | **6** (1,1) | 7 (1,2) |

*x* axis

## Three-Dimensional Array

Printed as list of matrices (3x3x3)

array([[[ 0, 1, 2],
[ 3, 4, 5],
[ 6, 7, 8]],

[[ 9, 10, 11],
[12, 13, 14],
[15, 16, 17]],

[[18, 19, 20],
[21, 22, 23],
[24, 25, 26]]]) ← 3 axes rank 3

Length = 3

*y* axis

*x* axis

*z* axis

# Sequence to create numpy array

```
#converting list in numpy array
#np.array() method use of numpy package to convert in into array
#1. import numpy package
#2. use array or ndarray to create array for given sequect
#3. print or perform operation array
import numpy as np
time = np.array(list_time)
speed = np.array(list_speed)
print(time)
print(speed)
distance = time * speed
print("Calculated Distance is ")
print(distance)
```

```
[0.2, 1, 0.5, 1.2, 1.05]
[30, 40, 10, 50, 20]
[0.2  1.   0.5  1.2  1.05]
[30 40 10 50 20]
Calculated Distance is
[ 6. 40.  5. 60. 21.]
```

# Creating and printing numpy array

```
import numpy as np
#creating 1D array
arr1 = np.array([1,2,3,4])
#creating 2D array
arr2 = np.array([[10,20,30],[11,22,33]])
#creating 3D array
arr3=np.array([[[1,2,3],[4,5,6]],[[10,20,30],[40,50,60]]])
print("1D array")
print(arr1)
print("2D array")
print(arr2)
print("3D array")
print(arr3)
```

```
1D array
[1 2 3 4]
2D array
[[10 20 30]
 [11 22 33]]
3D array
[[[ 1  2  3]
  [ 4  5  6]]

 [[10 20 30]
  [40 50 60]]]
```

# Class and attribute of numpy array - ndarray

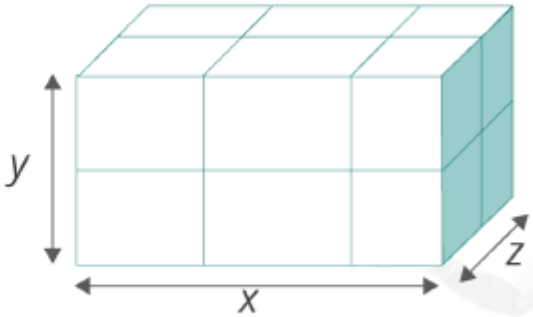
- An array is a combination of rows and columns or we can say table of elements (normally numeric) of same data type.
- **Numpy array is multidimensional array so ndarray is class of numpy for an array.**
- Elements of numpy array is accessed using square bracket and it's a combination of nested list. So we have to use list while creating numpy array.
- Based index of numpy array start with 0, all the element of numpy array occupy the same size in memory when you declare an array.
- Numpy.ndarray have attribute like **ndim, shape, dtype, size and itemsize.**

# ndarray.ndim

ndim represent the number of axis of an array. Its give that array is 1D, 2D, 3D or ND. So it gives dimension of an array.

This refers to the number of axes (dimensions) of the array. It is also called the rank of the array.


Two axes or 2D array




Three axes or 3D array

# Cont…

```
In [7]: import numpy as np
        #creating 1D array
        arr1 = np.array([1,2,3,4])
        #creating 2D array
        arr2 = np.array([[10,20,30],[11,22,33]])
        #creating 3D array
        arr3=np.array([[[1,2,3],[4,5,6]],[[10,20,30],[40,50,60]]])
        print("dimension of arr1 is : ",arr1.ndim)
        print("dimension of arr2 is : ",arr2.ndim)
        print("dimension of arr3 is : ",arr3.ndim)

        dimension of arr1 is :  1
        dimension of arr2 is :  2
        dimension of arr3 is :  3
```

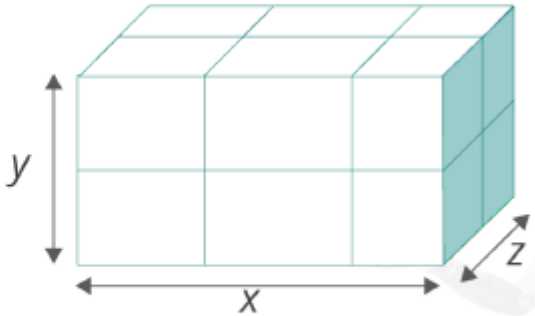
# ndarray.shape

- Shape of array : to find the shape of array numpy provide **shape** command which will **return how many raws and column numpy array have**

This consists of a tuple of integers showing the size of the array in each dimension. The length of the **shape tuple** is the rank or ndim.


2 rows, 3 columns

Shape: (2, 3)




2 rows, 3 columns, 2 ranks

Shape: (2, 3, 2)

# Cont…

```
In [5]: import numpy as np
        #creating 1D array
        arr1 = np.array([1,2,3,4])
        #creating 2D array
        arr2 = np.array([[10,20,30],[11,22,33]])
        #creating 3D array
        arr3=np.array([[[1,2,3],[4,5,6]],[[10,20,30],[40,50,60]]])
        #first number indecate total rows and second indecate total columns
        print("shape of arr1 is : ",arr1.shape)
        print("shape of arr2 is : ",arr2.shape)
        print("shape of arr3 is : ",arr3.shape)

        shape of arr1 is :  (4,)
        shape of arr2 is :  (2, 3)
        shape of arr3 is :  (2, 2, 3)
```

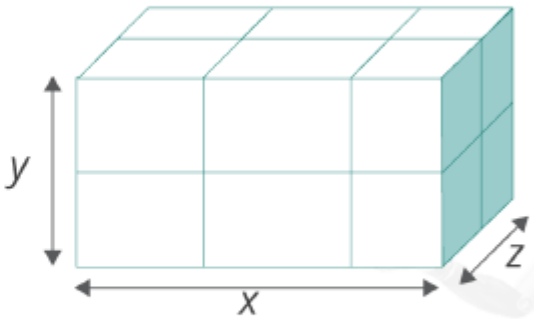
# ndarray.size

ndarray.size give how many element present into array. Multiplication of shapes.

It gives the total number of elements in the array. It is equal to the product of the elements of the shape tuple.


Array contains 6 elements

Array a = (2, 3)
Size = 6




Array contains 12 elements

Array b = (2, 3, 2)
Size = 12

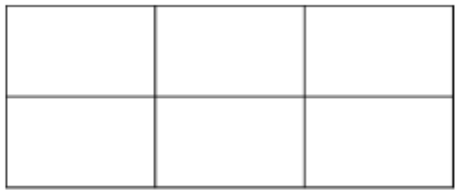
# Cont…

```
In [6]: import numpy as np
        #creating 1D array
        arr1 = np.array([1,2,3,4])
        #creating 2D array
        arr2 = np.array([[10,20,30],[11,22,33]])
        #creating 3D array
        arr3=np.array([[[1,2,3],[4,5,6]],[[10,20,30],[40,50,60]]])
        #size will retruns how many elements present in array
        print("size of arr1 is : ",arr1.size)
        print("size of arr2 is : ",arr2.size)
        print("size of arr3 is : ",arr3.size)

        size of arr1 is :  4
        size of arr2 is :  6
        size of arr3 is :  12
```

# ndarray.dtype

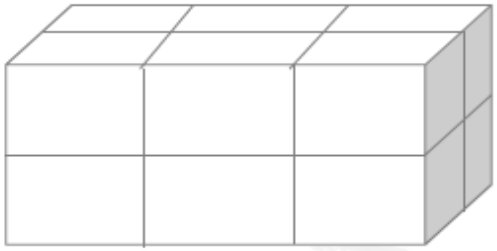
- ndarry.dtype is use to print data type of numpy array element like below.
- You can also define data type of array element by passing it at the time of array declaration.

It's an object that describes the type of the elements in the array. It can be created or specified using Python.



Array contains integers

Array a = [3, 7, 4]
[2, 1, 0]



Array contains floats

Array b = [1.3, 5.2, 6.7]
[0.2, 8.1, 9.4]

[2.6, 4.2, 3.9]
[7.8, 3.4, 0.8]

# Cont…

```
In [10]: import numpy as np
         #array of character
         Arra1=np.array(['a','b','c','d','e']) #1D array
         #array of integer value
         Arra2=np.array([[10,20,30],[40,50,60]]) #2D array
         #array of complex data type
         Arra3=np.array([1,2,3,4,5], dtype=complex)
         #dtype is use to print data type of array element
         #make sure all element into array is of same data type
         print("data type of array1 is : ",Arra1.dtype)
         print("data type of array2 is : ",Arra2.dtype)
         print("data type of array3 is : ", Arra3.dtype)

         data type of array1 is :  <U1
         data type of array2 is :  int32
         data type of array3 is :  complex128
```

# ndarray.itemsize

ndarray.itemsize will return size (in bytes) for each element by array elements.

```
In [14]: import numpy as np
         #array of character
         Arra1=np.array(['a','b','c','d','e']) #1D array
         #array of integer value
         Arra2=np.array([[10,20,30],[40,50,60]]) #2D array
         #array of complex data type
         Arra3=np.array([1,2,3,4,5], dtype=complex)
         #size of array elements
         print("size of array1 : ",Arra1.itemsize)
         print("size of array2 : ",Arra2.itemsize)
         print("size of array3 : ",Arra3.itemsize)

         size of array1 :  4
         size of array2 :  4
         size of array3 :  16
```

# Basic Operation on Array

Using the following operands, you can easily apply various mathematical, logical, and comparison operations on an array.

**Mathematical Operations**

| | |
|---|---|
| Addition | + |
| Subtraction | - |
| Multiplication | * |
| Division | / |
| Exponentiation | ** |

**Logical Operations**

| | |
|---|---|
| And | & |
| Or | \| |
| Not | ~ |

**Comparison Operations**

| | |
|---|---|
| Greater | > |
| Greater or equal | >= |
| Less | < |
| Less or equal | <= |
| Equal | == |
| Not equal | != |

# Array addition

```
In [15]: #addition without array
         list1 = [1,2,3,4]
         list2 = [10,2,11,2]
         print(list1 + list2) #it will concate two list but not perform additon

         [1, 2, 3, 4, 10, 2, 11, 2]

In [17]: import numpy as np
         list_a1 = np.array(list1)
         list_a2 = np.array(list2)
         print("array1 is : ",list_a1)
         print("array2 is : ",list_a2)
         print(list_a1 + list_a2)

         array1 is :  [1 2 3 4]
         array2 is :  [10  2 11  2]
         [11  4 14  6]
```

# Array Multiplication

```
In [20]: #Multiplication without array
         list1 = [1,2,3,4]
         list2 = [10,2,11,2]
         list1 * 2

Out[20]: [1, 2, 3, 4, 1, 2, 3, 4]

In [21]: print(list1 * list2)

         ---------------------------------------------------------------------------
         TypeError                                 Traceback (most recent call last)
         <ipython-input-21-0f6fd37ccef4> in <module>
         ----> 1 print(list1 * list2)

         TypeError: can't multiply sequence by non-int of type 'list'
```

# Cont…

In [22]:
```
import numpy as np
list_a1 = np.array(list1)
list_a2 = np.array(list2)
print("array1 is : ",list_a1)
print("array2 is : ",list_a2)
print("array mulitplication : ",list_a1 * list_a2)
```

```
array1 is :  [1 2 3 4]
array2 is :  [10  2 11  2]
array mulitplication :  [10  4 33  8]
```

In [25]:
```
list_a3 = np.array([[2],[3]])
print("mulitplication of array 1 and 3 : ")
print(list_a1 * list_a3)
```

```
mulitplication of array 1 and 3 :
[[ 2  4  6  8]
 [ 3  6  9 12]]
```

# Array Division

```
In [26]: #Division without array
         list1 = [1,2,3,4]
         list2 = [10,2,11,2]
         list1 / 2

         ---------------------------------------------------------------------------
         TypeError                                 Traceback (most recent call last)
         <ipython-input-26-4a605eba1faa> in <module>
               2 list1 = [1,2,3,4]
               3 list2 = [10,2,11,2]
         ----> 4 list1 / 2

         TypeError: unsupported operand type(s) for /: 'list' and 'int'

In [27]: list1 / list2

         ---------------------------------------------------------------------------
         TypeError                                 Traceback (most recent call last)
         <ipython-input-27-f45b7d221e4c> in <module>
         ----> 1 list1 / list2

         TypeError: unsupported operand type(s) for /: 'list' and 'list'
```

# Cont…

```
In [28]: import numpy as np
         list_a1 = np.array(list1)
         list_a2 = np.array(list2)
         print("array1 is : ",list_a1)
         print("array2 is : ",list_a2)
         print("array Division : ",list_a1 / list_a2)

         array1 is :  [1 2 3 4]
         array2 is :  [10  2 11  2]
         array Division :  [0.1        1.         0.27272727 2.        ]
```

Like wise one can perform any arithmetical (+,-, * , / ) operation on numpy array.

# Accessing array element by passing index

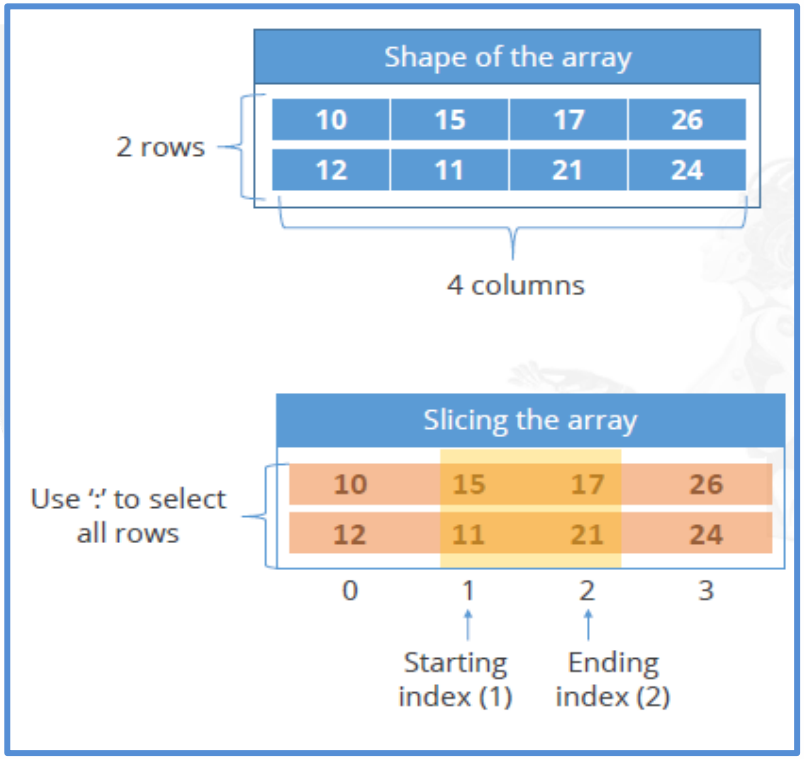
One can access only single row, column or only single element by passing index value.

```
In [39]: #accessing array element by passing index
         import numpy as np
         arr1 = np.array([1,2,3,4])
         #creating 2D array
         arr2 = np.array([[10,20,30],[11,22,33]])
         #creating 3D array
         arr3=np.array([[[1,2,3],[4,5,6]],[[10,20,30],[40,50,60]]])
         print(arr1[2])
         print(arr2[0]) #printing 1st row
         print(arr3[0])

         3
         [10 20 30]
         [[1 2 3]
          [4 5 6]]
```

# Slicing operation Array

To perform slicing operation need to pass starting index value and end index value

```
In [43]: import numpy as np
         #creating 2D array
         arr2 = np.array([[10,15,17,26],[12,11,21,24]])
         print(arr2)

         [[10 15 17 26]
          [12 11 21 24]]

In [49]: print(arr2[:,1])
         print(arr2[1:,2])
         print(arr2[1:,])
         print(arr2[0:,2])

         [15 11]
         [21]
         [[12 11 21 24]]
         [17 21]
```



# Cont…

```
In [50]: #creating 3D array
         arr3=np.array([[[1,2,3],[4,5,6]],[[10,20,30],[40,50,60]]])
         print(arr3)

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

          [[10 20 30]
           [40 50 60]]]

In [51]: print(arr3[:,1])

         [[ 4  5  6]
          [40 50 60]]

In [55]: print(arr3[1:,0])
         print(arr3[1:,0:1])
         print(arr3[:,1:3])

         [[10 20 30]]
         [[[10 20 30]]]
         [[[ 4  5  6]]

          [[40 50 60]]]
```

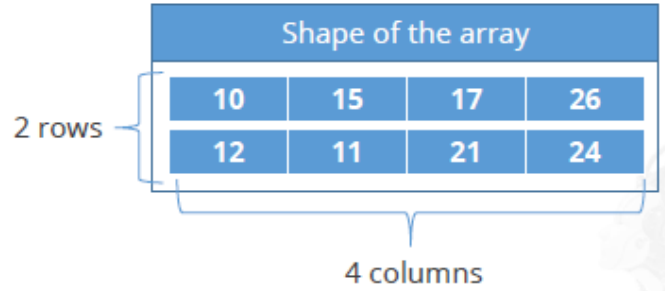
# Cont...

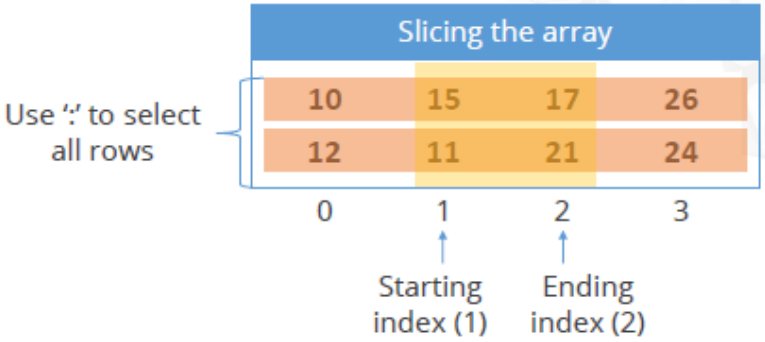
Use the slicing method to access a range of values within an array.

Shape of the array

```
In [152]: cyclist_trials.shape
Out[152]: (2L, 4L)

In [153]: two_cyclist_trial_data=cyclist_trials[:,1:3]

In [154]: two_cyclist_trial_data
Out[154]: array([[15, 17],
                 [11, 21]])
```

Slicing the array data [ : , 1 : 3]
where 1 is inclusive but 3 is not



# Iterating array using for loop

```
In [56]: #iterating array using loop
         myarr = np.array([[1,3,90,11],[23,11,34,10]])
         for i in myarr:
             print(i)

         [ 1  3 90 11]
         [23 11 34 10]

In [57]: for i in myarr[:,1]:
             print(i)

         3
         11

In [58]: for i in myarr[1:,0:3]:
             print(i)

         [23 11 34]
```

# Indexing with Boolean value

When wants select data according to given criteria than indexing using Boolean value is useful.

Here, the original dataset contains test scores of two students. You can use a Boolean array to choose only the scores that are above a given value.

Test scores

| | Test 1 | Test 2 | Test 3 | Test 4 |
|---|---|---|---|---|
| Student 1 | 83 | 71 | 57 | 63 |
| Student 2 | 54 | 68 | 81 | 45 |

Test score > 60

| | | | |
|---|---|---|---|
| 83 | 71 | 57 | 63 |
| 54 | 68 | 81 | 45 |

# Cont…

```
In [60]: testarray = np.array([[83,71,57,63],[54,68,81,45]])
         passingscore = testarray>65
         print(passingscore)

         [[ True  True False False]
          [False  True  True False]]

In [63]: #printing only those value which have index value as True
         testarray[passingscore]

Out[63]: array([83, 71, 68, 81])
```

# Numpy empty array creation

Numpy empty() create numpy array of given dimension / shape with the random value.

By default data type of numpy array is float. If data type is not mention

```
In [30]: import numpy as np
         emp_arr = np.empty(2, dtype=int)
         #here dtype indicate datatype of and array
         print(emp_arr)

         [         0 1073741824]

In [31]: emp_arr2 = np.empty([2,2])
         print(emp_arr2)

         [[0. 0.]
          [0. 0.]]
```

# Cont…

```
In [32]: emp_arr3 = np.empty([2,3,4], dtype=complex)
         print(emp_arr3)

[[[ 1.15319323e-311+1.15315813e-311j  2.70781692e+243+9.22730380e-143j
    9.00687745e-313+2.34247115e+166j  7.52180888e-313+3.68180845e+228j]
  [ 1.33360300e+241+4.58775490e-311j  1.26927730e-277+1.57989218e-313j
    2.84327065e-308+1.33360294e+241j  6.76067859e-311+4.31912682e-239j]
  [ 3.27748881e-313+2.70420353e-308j  1.33360297e+241+1.43659115e-310j
    1.77678572e-176+6.24828292e-313j  4.23394181e-308+1.33360304e+241j]]

 [[ 5.83676162e-310+4.13690163e-186j  8.58247829e-313+6.79907823e-308j
   -3.38460708e+125+3.69028917e+180j  2.92805813e-306+2.79806838e-309j]
  [ 2.78138476e-309+2.78138476e-309j  2.78681707e-309+8.20252618e-304j
    2.78138476e-309+2.78138485e-309j  1.09491781e-303+1.27631071e-303j]
  [ 5.48248335e-294+1.07171760e-014j  1.78428230e-125+4.09515068e+265j
    3.15765223e-053+2.61363525e-077j  7.63737327e-116+7.97224326e-320j]]]
```

# Array of zeros and ones

To create numpy array with all zeors and once we have to use zeros() and ones() method.

```
In [33]: #please import numpy if not import it
         zero_arr = np.zeros([2,2])
         print(zero_arr)

         [[0. 0.]
          [0. 0.]]

In [35]: one_arr = np.ones([3,2,2], dtype=int)
         print(one_arr)

         [[[1 1]
           [1 1]]

          [[1 1]
           [1 1]]

          [[1 1]
           [1 1]]]
```

# Numpy array within a numeric range

- To create numpy array within given range one have to use arrange() function.
- **arange(start, stop, step , dtype)** as argument
- It return evenly space value between given interval.

- **Start** : starting value of an array
- **Stop** : stopping value of an array, mean any random value between start and stop value. So stop value is not going to consider.
- **Step:** step value is value by which interval value is going to change.
- **dtype** : metion datatype of an array element.

** default datatype of numpy array is float

# Cont..

```
In [47]: #creating array using arange()
         #here reshape is optional is use to mention the dimenssion of an array
         a1 = np.arange(4).reshape(2,2)
         print(a1)

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

```
In [48]: a2 = np.arange(1,9,dtype=float).reshape(2,2,2)
         print(a2)

         [[[1. 2.]
           [3. 4.]]

          [[5. 6.]
           [7. 8.]]]
```

# Cont…

```
In [49]: a3 = np.arange(1,9,0.5)
         print(a3)

         [1.  1.5 2.  2.5 3.  3.5 4.  4.5 5.  5.5 6.  6.5 7.  7.5 8.  8.5]

In [50]: a4 = np.arange(1,9,0.5).reshape(4,2,2)
         print(a4)
         #but it gives an error is numbers of element not match with the total elements
         # as given in dimesion

         [[[1.  1.5]
           [2.  2.5]]

          [[3.  3.5]
           [4.  4.5]]

          [[5.  5.5]
           [6.  6.5]]

          [[7.  7.5]
           [8.  8.5]]]
```

# Numpy linspace

- **linspace is almost similar to arrange of numpy.**
- It also create array of evenly separated numbers but it not allow us to pass step value.
- It divide element between given interval
- **linspace(start, stop, num, endpoint, retstep, dtype)**
- **Start** : indicate the stating value
- **Stop** : indicate the stopping value
- **num** : total numbers of sample (default is 50) if not pass.
- **Endpoint** : value is True/ False. True indicate include stop value.
- **Retstep** : Boolean value which represent step and the sample value between interval.
- **Dtype** : represent datatype of array element.

# Cont…

In [1]:
```
#numpy array using linspace()method
#numpy.linspace(start, stop, num = 50, endpoint = True, retstep = False, dtype = None)
#endpoint=true mean stop value is included
#retstep = ture mean step value is visible
import numpy as np
a1_lin = np.linspace(1,5,num=20,endpoint=True)
print(a1_lin)
```

```
[1.         1.21052632 1.42105263 1.63157895 1.84210526 2.05263158
 2.26315789 2.47368421 2.68421053 2.89473684 3.10526316 3.31578947
 3.52631579 3.73684211 3.94736842 4.15789474 4.36842105 4.57894737
 4.78947368 5.        ]
```

In [2]:
```
a1_lin = np.linspace(1,5,num=20,retstep=True)
print(a1_lin)
```

```
(array([1.        , 1.21052632, 1.42105263, 1.63157895, 1.84210526,
       2.05263158, 2.26315789, 2.47368421, 2.68421053, 2.89473684,
       3.10526316, 3.31578947, 3.52631579, 3.73684211, 3.94736842,
       4.15789474, 4.36842105, 4.57894737, 4.78947368, 5.        ]), 0.21052631578947367)
```

In [3]:
```
a1_lin = np.linspace(1,5,num=20,retstep=True,dtype=int)
print(a1_lin)
```

```
(array([1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 4, 4, 4, 4, 5]), 0.21052631578947367)
```

# Cont…

```
In [12]: a2_lin = np.linspace(1,10,num=20,endpoint=True,dtype=int).reshape([2,10])
         print(a2_lin)

[[ 1  1  1  2  2  3  3  4  4  5]
 [ 5  6  6  7  7  8  8  9  9 10]]

In [10]: a2_lin = np.linspace(1,10,num=20,retstep=True)
         print(a2_lin)

(array([ 1.        ,  1.47368421,  1.94736842,  2.42105263,  2.89473684,
        3.36842105,  3.84210526,  4.31578947,  4.78947368,  5.26315789,
        5.73684211,  6.21052632,  6.68421053,  7.15789474,  7.63157895,
        8.10526316,  8.57894737,  9.05263158,  9.52631579, 10.        ]), 0.47368421052631576)

In [14]: a3_lin = np.linspace(1,10,num=20,endpoint=False,dtype=int).reshape([4,5])
         print(a3_lin)

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

# More Function of numpy array

| Method | Description |
|---|---|
| identity() | Create identity array for given dimension |
| copy() | Returns copy of array for given object |
| longspace() | Return array of evenly space with element with long space |
| sum() | Return sum of all element present in an array |
| max() | Return maximum element form an array. |
| sqrt() | Return square root of array element |
| min() | Return Minimum element from an array. |
| For more : https://www.javatpoint.com/numpy-array-iteration<br>https://www.geeksforgeeks.org/numpy-array-creation/?ref=lbp<br>https://numpy.org/doc/stable/ | |

# Mathematical function of numpy

As numpy is use to perform numerical operation so numpy package have lot many function to perform some mathematical operation like trigonometric operation, arithmetic operation, operation on complex data etc.

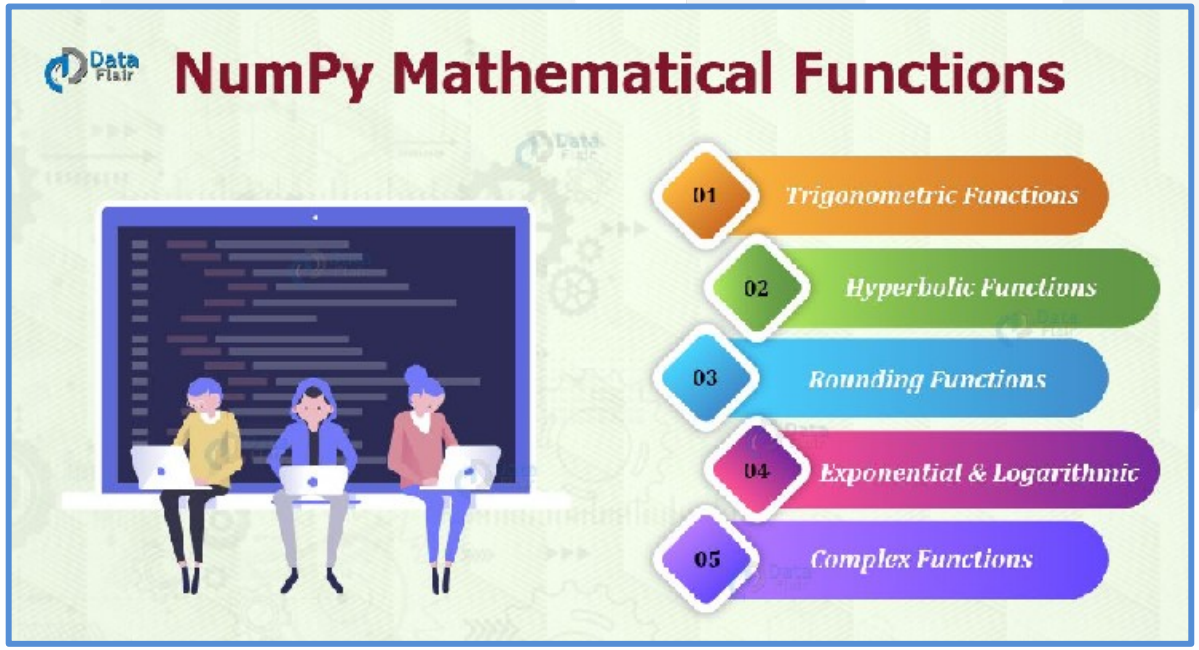



Image Source : DataFlair

# Numpy Trigonometric function

In [16]:
```
#numpy have trignomatric function like cos,sin, tan function
# to calculate cosin, sin and tangent value for given data
import numpy as np;
A1 = np.array([30,45,60,90,120])
#calculating sin value for given array
sin_a1 = np.sin(A1)
print("sin value for given array is : ", sin_a1)
```

```
sin value for given array is :  [-0.98803162  0.85090352 -0.30481062  0.89399666  0.58061118]
```

In [19]:
```
#now creating array of pi value
import math
A2 = np.array([0,math.pi,math.pi/2,math.pi/3,np.pi/2,np.pi/3, np.pi])
print(A2)
```

```
[0.         3.14159265 1.57079633 1.04719755 1.57079633 1.04719755
 3.14159265]
```

In [20]:
```
cos_a2 = np.cos(A2)
print("cosine value for given array is : ", cos_a2)
```

```
cosine value for given array is :  [ 1.000000e+00 -1.000000e+00  6.123234e-17  5.000000e-01  6.123234e-17
  5.000000e-01 -1.000000e+00]
```

# Cont…

```
In [21]: #calculating tangent value
         tan_a1 = np.tan(A1)
         print("Tangent Value : ",tan_a1)

         Tangent Value :  [-6.4053312   1.61977519  0.32004039 -1.99520041  0.71312301]

In [22]: print("Tangent value of second array : ",np.tan(A2))

         Tangent value of second array :  [ 0.00000000e+00 -1.22464680e-16  1.63312394e+16  1.73205081e+00
           1.63312394e+16  1.73205081e+00 -1.22464680e-16]

In [23]: #like wise numpy have so many trignomatric funcitons like
         # arcsin(), arccos(), and arctan()
         #which return inverse Trignomatric value
```

For more please refer :
https://www.geeksforgeeks.org/numpy-mathematical-function/?ref=lbp
https://www.javatpoint.com/numpy-mathematical-functions

# Rounding Functions

```
In [45]: #roundoff function return decimal value for desired decimal value
         #2.5 consider as 3 likewise for other value
         # Python program explaining
         # around() function

         import numpy as np

         in_array = [.5, 1.5, 2.5, 3.5, 4.5, 10.1]
         print ("Input array : \n", in_array)

         round_off_values = np.around(in_array)
         print ("\nRounded values : \n", round_off_values)

         Input array :
          [0.5, 1.5, 2.5, 3.5, 4.5, 10.1]

         Rounded values :
          [ 0.  2.  2.  4.  4. 10.]
```

# Cont…

```
In [46]: in_array = [.53, 1.54, .71]
         print ("\nInput array : \n", in_array)

         round_off_values = np.around(in_array)
         print ("\nRounded values : \n", round_off_values)

         in_array = [.5538, 1.33354, .71445]
         print ("\nInput array : \n", in_array)

         round_off_values = np.around(in_array, decimals = 3)
         print ("\nRounded values : \n", round_off_values)

         Input array :
          [0.53, 1.54, 0.71]

         Rounded values :
          [1. 2. 1.]

         Input array :
          [0.5538, 1.33354, 0.71445]

         Rounded values :
          [0.554 1.334 0.714]
```

Like around one more function is there **round()** both function work same.

# floor() and ceil() funciton

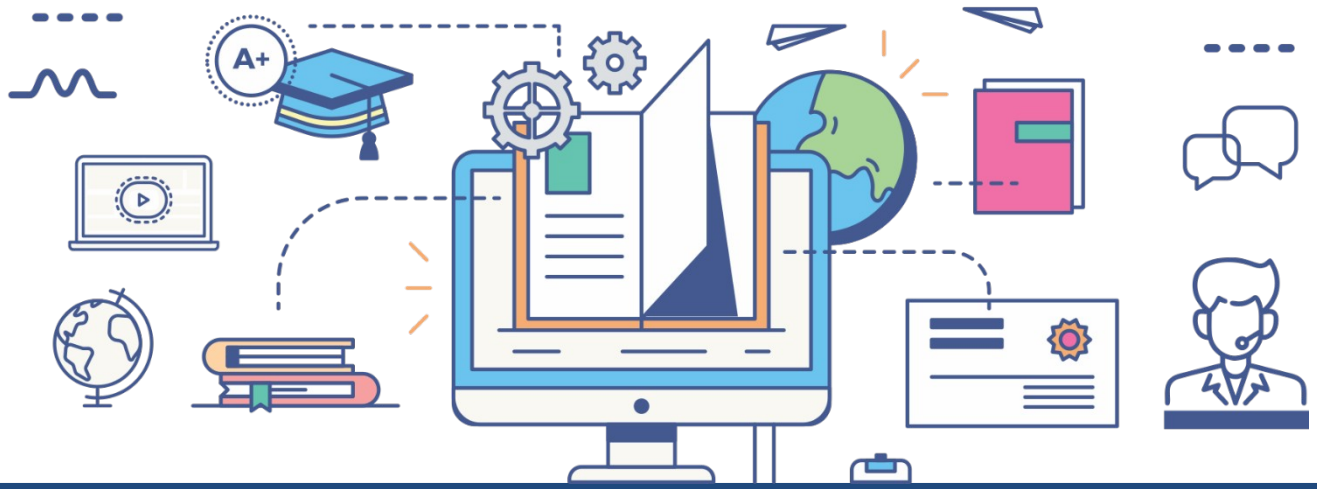
```
In [47]: #floor() function
         #return the floor value of the input data which is the largest integer not greater than the input value.
         import numpy as np
         arr = np.array([12.202, 90.23120, 123.020, 23.202])
         print(np.floor(arr))

         [ 12.  90. 123.  23.]

In [48]: #ceil() function
         #return the ceiling value of the array values which is the smallest integer
         arr = np.array([12.202, 90.23120, 123.020, 23.202])
         print(np.ceil(arr))

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

Like this numpy have lots of function like logarithm, statistical and many more for that please visit given link on next slide

# For more about numpy

https://www.geeksforgeeks.org/numpy-mathematical-function/?ref=lbp
https://www.javatpoint.com/numpy-tutorial
https://numpy.org/doc/stable/reference/index.html
https://www.w3schools.com/python/numpy_intro.asp
https://www.tutorialspoint.com/numpy/index.htm
https://www.guru99.com/numpy-tutorial.html

# DIGITAL LEARNING CONTENT




Parul® University

www.paruluniversity.ac.in