Numpy

In [3]:
```
# range() and arange()

import numpy as np
```

In [4]:
```
a=np.array([10,20,30])
a
```

Out[4]:
```
array([10, 20, 30])
```

In [5]:
```
ar=np.array(range(1,101))
ar
```

Out[5]:
```
array([  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,
        27,  28,  29,  30,  31,  32,  33,  34,  35,  36,  37,  38,  39,
        40,  41,  42,  43,  44,  45,  46,  47,  48,  49,  50,  51,  52,
        53,  54,  55,  56,  57,  58,  59,  60,  61,  62,  63,  64,  65,
        66,  67,  68,  69,  70,  71,  72,  73,  74,  75,  76,  77,  78,
        79,  80,  81,  82,  83,  84,  85,  86,  87,  88,  89,  90,  91,
        92,  93,  94,  95,  96,  97,  98,  99, 100])
```

In [6]:
```
# create an array with even numbers in the range of 100
ae=np.array(range(2,101,2))
ae
```

Out[6]:
```
array([  2,   4,   6,   8,  10,  12,  14,  16,  18,  20,  22,  24,  26,
        28,  30,  32,  34,  36,  38,  40,  42,  44,  46,  48,  50,  52,
        54,  56,  58,  60,  62,  64,  66,  68,  70,  72,  74,  76,  78,
        80,  82,  84,  86,  88,  90,  92,  94,  96,  98, 100])
```

In [7]:
```
# array with odd numbers from 1 to 100
ae=np.array(range(1,100,2))
ae
```

Out[7]:
```
array([ 1,  3,  5,  7,  9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33,
       35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63, 65, 67,
       69, 71, 73, 75, 77, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97, 99])
```

In [8]:
```
# arange() -> belongs to numpy
a=np.arange(1,101)
a
```

Out[8]:
```
array([  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,
        27,  28,  29,  30,  31,  32,  33,  34,  35,  36,  37,  38,  39,
        40,  41,  42,  43,  44,  45,  46,  47,  48,  49,  50,  51,  52,
        53,  54,  55,  56,  57,  58,  59,  60,  61,  62,  63,  64,  65,
        66,  67,  68,  69,  70,  71,  72,  73,  74,  75,  76,  77,  78,
        79,  80,  81,  82,  83,  84,  85,  86,  87,  88,  89,  90,  91,
        92,  93,  94,  95,  96,  97,  98,  99, 100])
```

In [9]:
```
a=np.arange(25)
print(a.ndim)
b=a.reshape(5,5)
# converting 1D array into 2D
print(b)
print(b.ndim)
```

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

- linspace-> returns no of spaces evenly w.r.to given interval

In [10]:
```
a=np.linspace(1,20,5)
a
```

Out[10]:
```
array([ 1.  ,  5.75, 10.5 , 15.25, 20.  ])
```

In [11]:
```
a=np.linspace(1,5)
a
```

Out[11]:
```
array([1.        , 1.08163265, 1.16326531, 1.24489796, 1.32653061,
       1.40816327, 1.48979592, 1.57142857, 1.65306122, 1.73469388,
       1.81632653, 1.89795918, 1.97959184, 2.06122449, 2.14285714,
       2.2244898 , 2.30612245, 2.3877551 , 2.46938776, 2.55102041,
       2.63265306, 2.71428571, 2.79591837, 2.87755102, 2.95918367,
       3.04081633, 3.12244898, 3.20408163, 3.28571429, 3.36734694,
       3.44897959, 3.53061224, 3.6122449 , 3.69387755, 3.7755102 ,
       3.85714286, 3.93877551, 4.02040816, 4.10204082, 4.18367347,
       4.26530612, 4.34693878, 4.42857143, 4.51020408, 4.59183673,
       4.67346939, 4.75510204, 4.83673469, 4.91836735, 5.        ])
```

array Initialization

- np.ones() -> create an array with given shape and type with ones

In [12]:
```
1 np.ones(10)
```

Out[12]:
```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

In [13]:
```
1 np.ones([2,2],dtype=int)
```

Out[13]:
```
array([[1, 1],
       [1, 1]])
```

In [14]:
```
1 np.ones((3,2,1))
```

Out[14]:
```
array([[[1.],
        [1.]],

       [[1.],
        [1.]],

       [[1.],
        [1.]]])
```

In [15]:
```
1 np.zeros(10)
```

Out[15]:
```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

In [16]:
```
1 np.zeros(10,dtype=int)
```

Out[16]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

In [17]:
```
1 np.zeros([5,5])
```

Out[17]:
```
array([[0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.]])
```

In [18]:
```
1 a=np.full([2,2],45)
2 a
```

Out[18]:
```
array([[45, 45],
       [45, 45]])
```

In [19]:
```
1 np.full([2,2])
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_1408\2854243726.py in <module>
----> 1 np.full([2,2])

TypeError: full() missing 1 required positional argument: 'fill_value'
```

In [20]:
```
1 #(3,3) ,fillvalue 100
```

In [21]:
```
1 np.empty(9)
```

Out[21]:
```
array([0.0000000e+000, 0.0000000e+000, 0.0000000e+000, 0.0000000e+000,
       0.0000000e+000, 4.3477777e-321, 7.5660388e-307, 8.4560344e-307,
       3.4969892e-317])
```

In [22]:
```
1 np.empty([2,2])
```

Out[22]:
```
array([[4.3477777e-321, 7.5660388e-307],
       [8.4560344e-307, 3.4969892e-317]])
```

- eye() -> create 2D array with 1's on diagonal and 0's elsewhere

In [23]:
```
1 np.eye(2)
```

Out[23]:
```
array([[1., 0.],
       [0., 1.]])
```

In [24]:
```
1 np.eye(3)
```

Out[24]:
```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

In [25]:
```
1 np.identity(2)
```

Out[25]:
```
array([[1., 0.],
       [0., 1.]])
```

In [26]:
```
1 np.diag([3,4,5])
```

Out[26]:
```
array([[3, 0, 0],
       [0, 4, 0],
       [0, 0, 5]])
```

In [27]:
```
# accessing 1D array elements
a=np.array([39,89,76,55])
print(a)
print(a[0])
print(a[1])
print(a[0]+a[1])
```

```
[39 89 76 55]
39
89
128
```

In [28]:
```
# creating 2Dimensional array
l1=[10,20,30]
l2=[44,55,66]
b=np.array([l1,l2])
print(b.ndim)
print(b)
```

```
2
[[10 20 30]
 [44 55 66]]
```

In [29]:
```
# accessing 2D array elements
print(b[0,0])  # row index,col index
print(b[1,1])
```

```
10
55
```

In [30]:
```
# creating 3D array

c=np.array([ [[10,20],[30,40]],
             [[50,60],[70,80]] ])
print(c.ndim)
print(c)
```

```
3
[[[10 20]
  [30 40]]

 [[50 60]
  [70 80]]]
```

In [31]:
```
c[0]
```

Out[31]:
```
array([[10, 20],
       [30, 40]])
```

In [32]:
```
1 c[1]
```

Out[32]:
```
array([[50, 60],
       [70, 80]])
```

In [33]:
```
1 c[1,0,0]
```

Out[33]:
```
50
```

In [34]:
```
1 c[0,1,1]
```

Out[34]:
```
40
```

In [35]:
```
1 c[1,0,1]
```

Out[35]:
```
60
```

In [36]:
```
1 # random
2
3 np.random.random()  # random value b/w 0 to 1
```

...

In [38]:
```
1 # randint
2
3 np.random.randint(700)
```

...

In [39]:
```
1 np.random.randint(50,100,5)
```

Out[39]:
```
array([75, 67, 72, 68, 91])
```

In [42]:
```
1 #  100,200,10
2
3 a=np.random.randint(100,200,10)
4 a
```

Out[42]:
```
array([189, 174, 193, 149, 188, 128, 168, 186, 165, 175])
```

In [43]:
```
1 # fancy indexing or boolean indexing
2 a>150
```

Out[43]:
```
array([ True,  True,  True, False,  True, False,  True,  True,  True,
        True])
```

In [44]:
```
1 a[a>150]
```

Out[44]:
```
array([189, 174, 193, 188, 168, 186, 165, 175])
```

In [46]:
```
# slicing [start:stop:step]
a=np.random.randint(1,10,5)
a
```

Out[46]: array([5, 6, 3, 9, 7])

In [47]:
```
a[:3]
```

Out[47]: array([5, 6, 3])

In [48]:
```
a[::-1]
```

Out[48]: array([7, 9, 3, 6, 5])

In [49]:
```
a[::2]
```

Out[49]: array([5, 3, 7])

In [50]:
```
a[1::2]
```

Out[50]: array([6, 9])

In [52]:
```
a=np.arange(32)
a
```

...

In [53]:
```
a.reshape(4,8)
```

...

In [54]:
```
a.reshape(8,4)
```

...

In [55]:
```
a.reshape(4,4,2)
```

...

In [56]:
```
a.reshape(-1,4)
```

...

In [57]:
```
a.reshape(8,-1)
```

...

In [60]:
```
def grt(a,b):
    if a>b:
        return a
    else:
        return b
grt(90,870)
```

...

In [61]:
```
grt([10,2,340],[90,89,87])
```

...

In [62]:
```
vgrt=np.vectorize(grt)
vgrt([10,2,340],[90,89,87])
```

...

In [64]:
```
a=np.array([[1,2],[3,4]])
b=np.identity(2)
print(a)
print(b)
```

...

In [65]:
```
a+b
```

...

In [66]:
```
a*b   # normal multiplication
```

Out[66]:
```
array([[1., 0.],
       [0., 4.]])
```

In [67]:
```
np.dot(a,b)  # matrix multiplication
```

Out[67]:
```
array([[1., 2.],
       [3., 4.]])
```

Pandas

- pandas stands for python data analysis library
- It is a free and open source
- It is an excellent tool for Data processing and analysing real world data

Data structures :

- 1. Series: Used to create 1Dimensional array with named index eg: single column in excel sheet

- 2. Dataframe: Used to create 2 Dimensional array with row index and column index eg: table in an excel sheet

In [68]:
```
import pandas as pd
```

In [69]:
```
pd.__version__
```

...

In [72]:
```
# creating series using list
l=[90,98,78,65]
s=pd.Series(l)
print(s)
print(type(s))
```

...

In [71]:
```
# changing the index
s.index=['a','b','c','d']
s
```

...

In [73]:
```
s=pd.Series([9,"hello",7],index=['x','y','z'])
s
```

...

In [74]:
```
# creating series using numpy array
ar=np.array([90,87,65])
s1=pd.Series(ar)
print(s1)
```

```
0    90
1    87
2    65
dtype: int32
```

In [75]:
```
###creating series using tuple
s=pd.Series((10,89,76))
s
```

...

In [86]:
```
# creating series by using dictionary
d={"stu1":100,"stu2":89,"stu3":78}
s=pd.Series(d)
s
```

...

In [87]:
```
# accessing series
s["stu1"]
```

...

In [88]:
```
[i**2 for i in range(1,11)] # list comprehension
```

Out[88]: [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

In [89]:
```
s=pd.Series([i**2 for i in range(10,21)])
s
```

...

In [78]:
```
# Date-range()

s=pd.date_range(start="2023-10-1",end="2023-10-10")
s
```

...

DataFrame:

- data frame is a two-dimensional size mutable heterogeneous tabular data
- like a table in an excel sheet

In [79]:
```
# creating a dataframe
l=[10,20,30,40]
df=pd.DataFrame(l)
df
```

...

In [80]:
```
l=[[11,12,13],[14,5,16]]
df=pd.DataFrame(l)
df
```

...

In [82]:
```
df.index=['x','y']  # row index
df
```

...

In [84]:
```
df.columns=["col1","col2","col3"]
df
```

...

In [92]:
```
#creating a data frame by using series
df=pd.DataFrame(pd.Series([10,20,30],index=[1,2,3]))
df
```

. . .

In [93]:
```
# dictionary
d={"a":pd.Series([10,20,30,40],index=[1,2,3,4]),
   "b":pd.Series([50,60,70,80],index=[2,3,4,5]),
   "c":pd.Series([11,12,3],index=[1,2,3]) }
df=pd.DataFrame(d)
df
```

Out[93]:

	a	b	c
1	10.0	NaN	11.0
2	20.0	50.0	12.0
3	30.0	60.0	3.0
4	40.0	70.0	NaN
5	NaN	80.0	NaN

In [97]:
```
# combining dataframe (concat,append,merging)
d={"emp":pd.Series(['a','b','c','d'],index=[1,2,3,4]),
   "year":pd.Series([2018,2016,2015,2014],index=[1,2,3,4])}
df1=pd.DataFrame(d)
df1
```

Out[97]:

	emp	year
1	a	2018
2	b	2016
3	c	2015
4	d	2014

In [96]:
```
d={"emp":pd.Series(['a','b','c','e'],index=[1,2,3,4]),
   "dept":pd.Series(["hr","op","finance","pro"],index=[1,2,3,4])}
df2=pd.DataFrame(d)
df2
```

. . .

In [98]:
```
pd.merge(df1,df2) # based on emp names
```

. . .

In [99]:
```
pd.merge(df1,df2,how="left")
```

...

In [100]:
```
pd.merge(df1,df2,how="right")
```

Out[100]:

	emp	year	dept
0	a	2018.0	hr
1	b	2016.0	op
2	c	2015.0	finance
3	e	NaN	pro

Data analysis:

- Data analysis is the process of obtaining raw data and subsequently converting it into information,useful for decision making by users

Data analysis

- Analysing numerical data by using Numpy
- Tabular data using pandas
- Visualization using Matplotlib

Data

- Data is facts and statistics collected together for analysis or reference

Data collection

- Primary data: data that is collected by research himself/Herself ex:Interviews,experiments etc
- Secondary data collection: Data that is generated by someone else earlier Ex:Websites,books

```
### types of data
* Structed data: rows/columns t.e table ex:excel,.csv
* Unstructed data: audio,video,pictures etc
* Semi structured data: Json,html etc
```