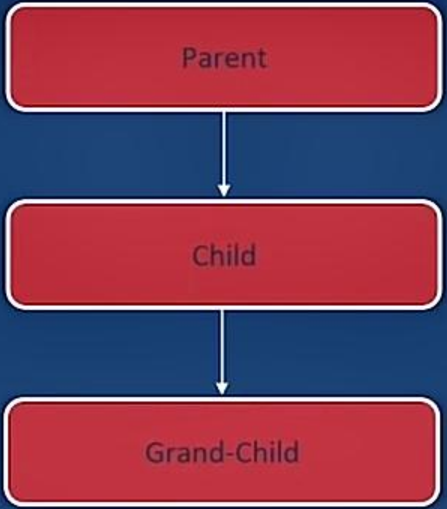
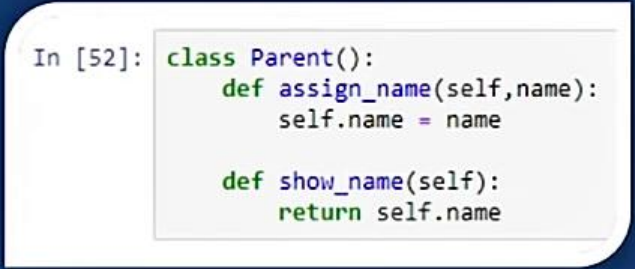
Multi-Level Inheritance

In multi-level Inheritance, we have Parent, child, grand-child relationship



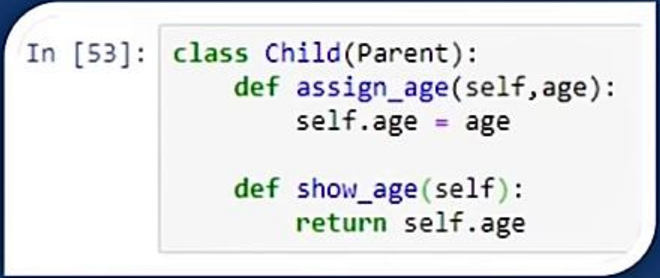
Multi-Level Inheritance in Python

Parent Class



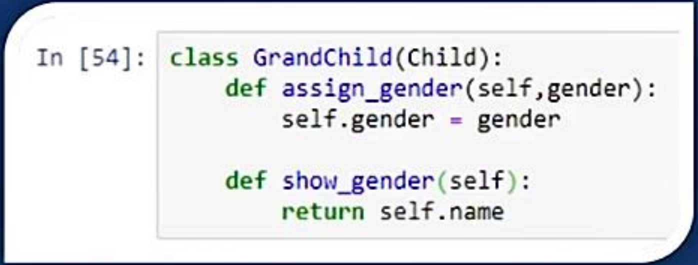
```
In [52]: class Parent():
             def assign_name(self,name):
                 self.name = name

             def show_name(self):
                 return self.name
```

Child Class



```
In [53]: class Child(Parent):
             def assign_age(self,age):
                 self.age = age

             def show_age(self):
                 return self.age
```

Grand-Child Class



```
In [54]: class GrandChild(Child):
             def assign_gender(self,gender):
                 self.gender = gender

             def show_gender(self):
                 return self.name
```

Libraries in Python

Python library is a collection of functions and methods that allows you to perform many actions without writing your code

NumPy

matplotlib

Pandas

Python NumPy

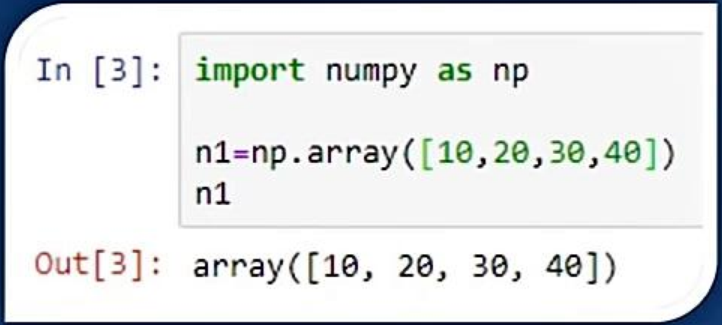
NumPy stands for Numerical python and is the core library for numeric and scientific computing

It consists of multi-dimensional array objects and a collection of routines for processing those arrays

NumPy

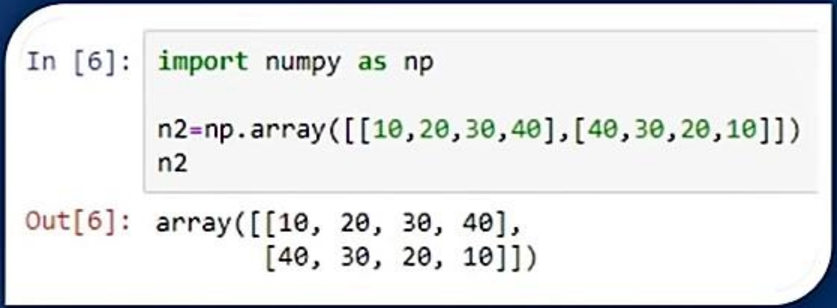
Creating NumPy Array

Single-dimensional Array



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

        n1=np.array([10,20,30,40])
        n1

Out[3]: array([10, 20, 30, 40])
```

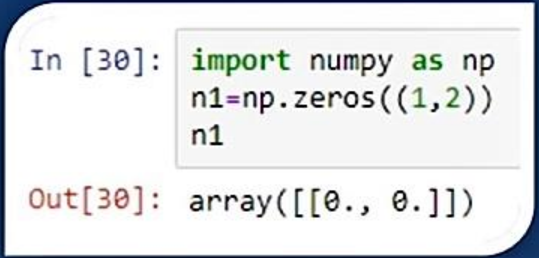
Multi-dimensional Array



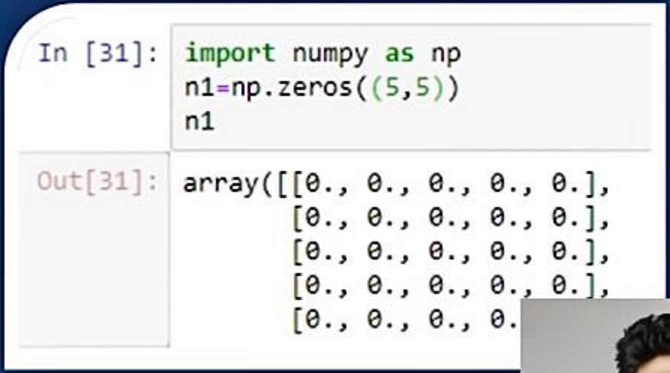
```
In [6]: import numpy as np

        n2=np.array([[10,20,30,40],[40,30,20,10]])
        n2

Out[6]: array([[10, 20, 30, 40],
               [40, 30, 20, 10]])
```

Initializing NumPy Array

Initializing NumPy array with zeros



```
In [30]:  import numpy as np
          n1=np.zeros((1,2))
          n1

Out[30]:  array([[0., 0.]])
```

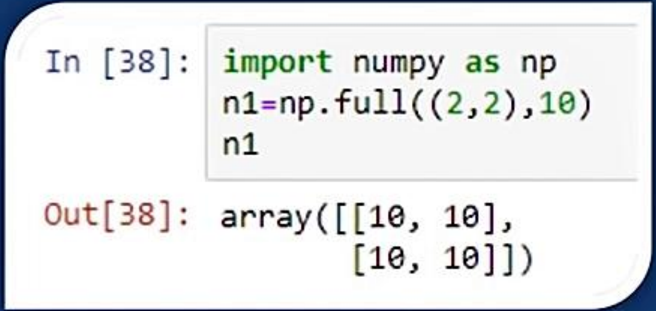


```
In [31]:  import numpy as np
          n1=np.zeros((5,5))
          n1

Out[31]:  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.
```

Initializing NumPy Array

greatlearning
Learning for Life

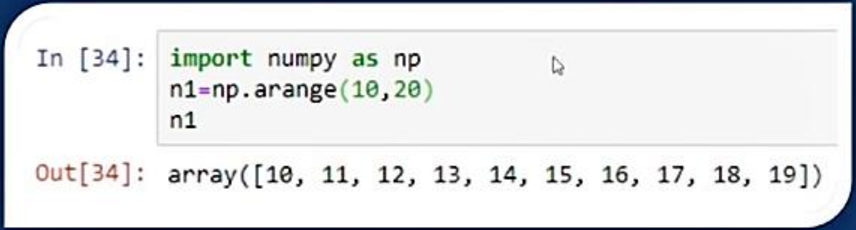
Initializing NumPy array with same number



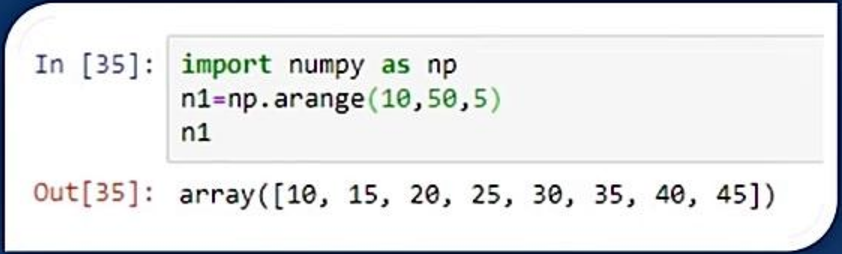
```
In [38]: import numpy as np
         n1=np.full((2,2),10)
         n1

Out[38]: array([[10, 10],
                [10, 10]])
```

Initializing NumPy Array

Initializing NumPy array within a range



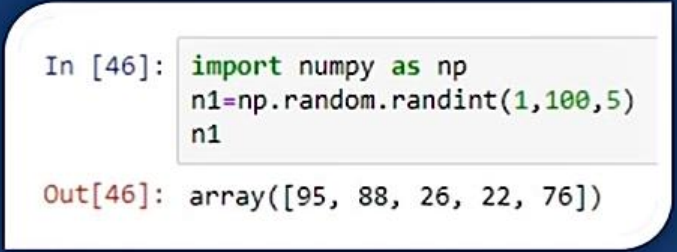
```
In [34]: import numpy as np
         n1=np.arange(10,20)
         n1

Out[34]: array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19])
```



```
In [35]: import numpy as np
         n1=np.arange(10,50,5)
         n1

Out[35]: array([10, 15, 20, 25, 30, 35, 40, 45])
```

Initializing NumPy Array

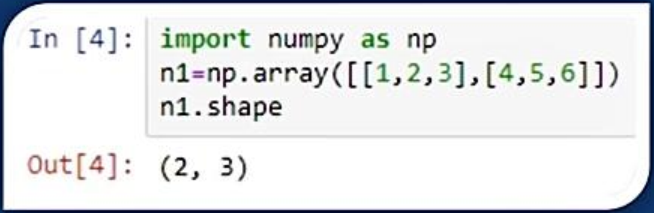
Initializing NumPy array with random numbers



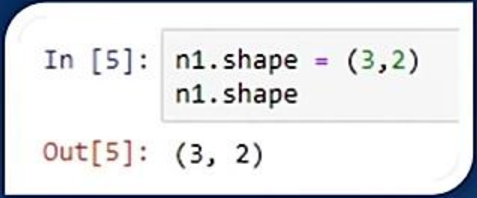
```
In [46]: import numpy as np
         n1=np.random.randint(1,100,5)
         n1

Out[46]: array([95, 88, 26, 22, 76])
```

NumPy-Shape

Checking the shape of NumPy arrays



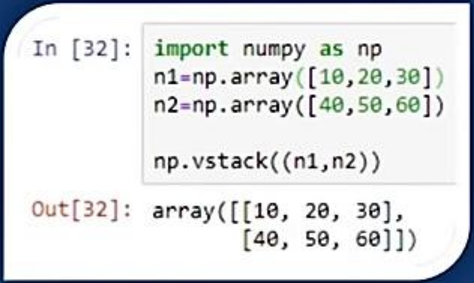
```
In [4]: import numpy as np
        n1=np.array([[1,2,3],[4,5,6]])
        n1.shape

Out[4]: (2, 3)
```



```
In [5]: n1.shape = (3,2)
        n1.shape

Out[5]: (3, 2)
```

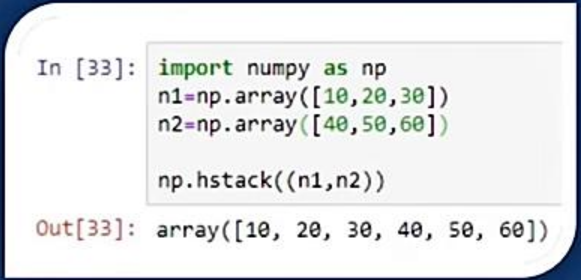
Joining NumPy Arrays

vstack()



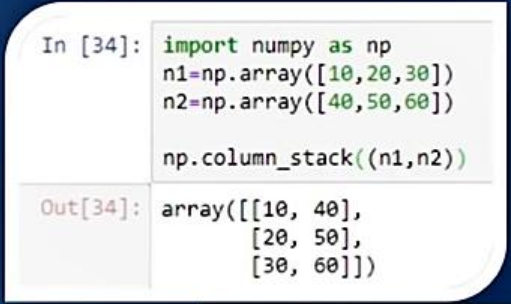
```
In [32]: import numpy as np
         n1=np.array([10,20,30])
         n2=np.array([40,50,60])

         np.vstack((n1,n2))

Out[32]: array([[10, 20, 30],
                [40, 50, 60]])
```

hstack()



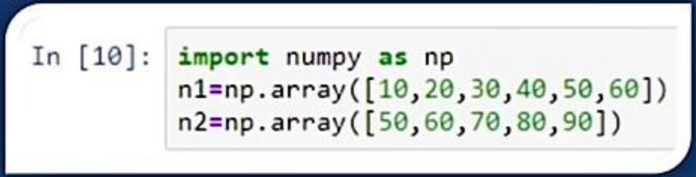
```
In [33]: import numpy as np
         n1=np.array([10,20,30])
         n2=np.array([40,50,60])

         np.hstack((n1,n2))

Out[33]: array([10, 20, 30, 40, 50, 60])
```

column_stack()

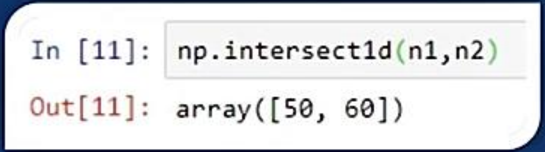


```
In [34]: import numpy as np
         n1=np.array([10,20,30])
         n2=np.array([40,50,60])

         np.column_stack((n1,n2))

Out[34]: array([[10, 40],
                [20, 50],
                [30, 60]])
```

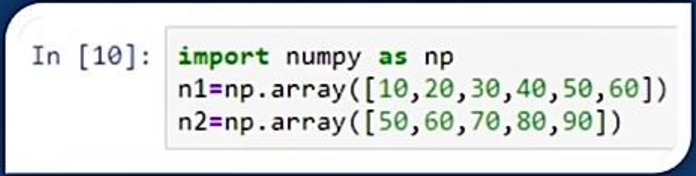
Numpy Intersection & Difference



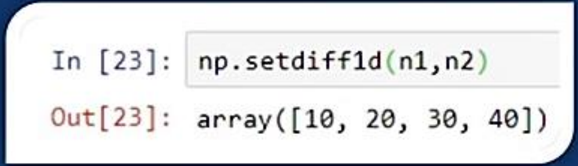
```
In [10]: import numpy as np
         n1=np.array([10,20,30,40,50,60])
         n2=np.array([50,60,70,80,90])
```



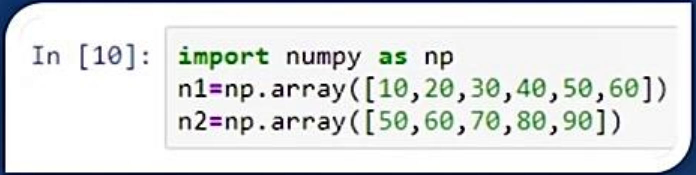
```
In [11]: np.intersect1d(n1,n2)

Out[11]: array([50, 60])
```



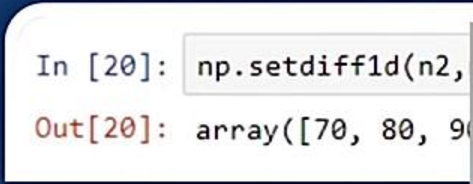
```
In [10]: import numpy as np
         n1=np.array([10,20,30,40,50,60])
         n2=np.array([50,60,70,80,90])
```



```
In [23]: np.setdiff1d(n1,n2)

Out[23]: array([10, 20, 30, 40])
```



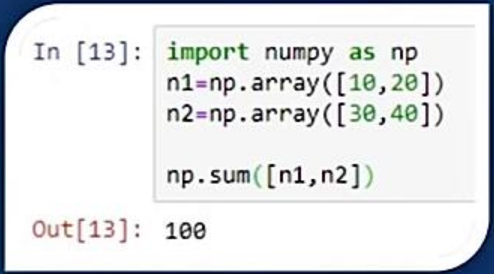
```
In [10]: import numpy as np
         n1=np.array([10,20,30,40,50,60])
         n2=np.array([50,60,70,80,90])
```



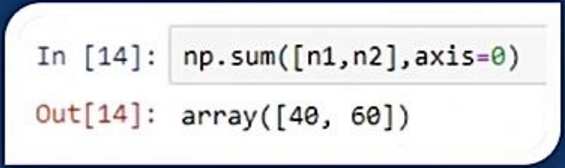
```
In [20]: np.setdiff1d(n2,

Out[20]: array([70, 80, 9
```

NumPy Array Mathematics

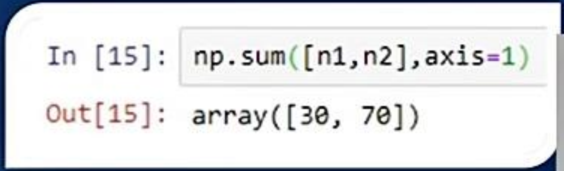
Addition of NumPy Arrays



```
In [13]: import numpy as np
         n1=np.array([10,20])
         n2=np.array([30,40])

         np.sum([n1,n2])

Out[13]: 100
```



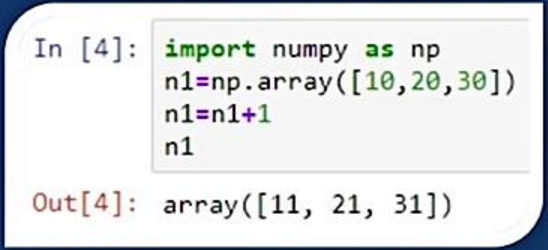
```
In [14]: np.sum([n1,n2],axis=0)

Out[14]: array([40, 60])
```



```
In [15]: np.sum([n1,n2],axis=1)

Out[15]: array([30, 70])
```

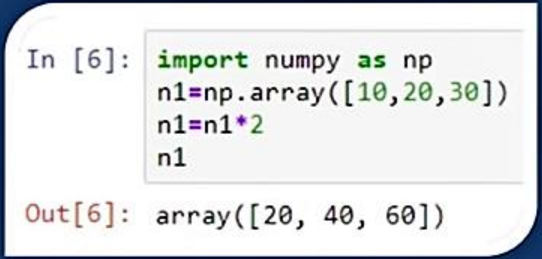
NumPy Array Mathematics

Basic Addition



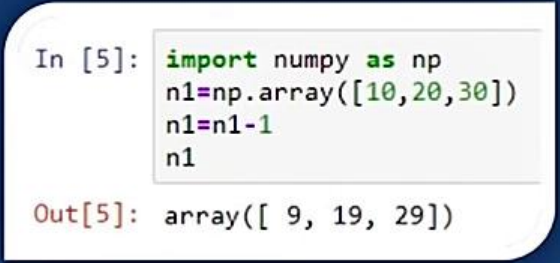
```
In [4]: import numpy as np
        n1=np.array([10,20,30])
        n1=n1+1
        n1

Out[4]: array([11, 21, 31])
```

Basic Multiplication



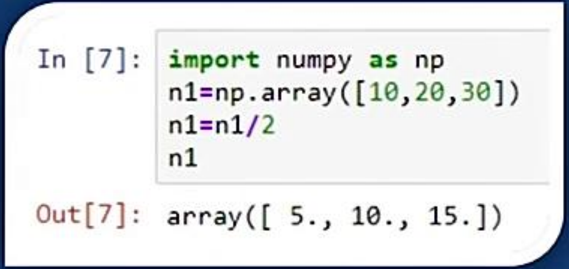
```
In [6]: import numpy as np
        n1=np.array([10,20,30])
        n1=n1*2
        n1

Out[6]: array([20, 40, 60])
```

Basic Subtraction



```
In [5]: import numpy as np
        n1=np.array([10,20,30])
        n1=n1-1
        n1

Out[5]: array([ 9, 19, 29])
```

Basic Division

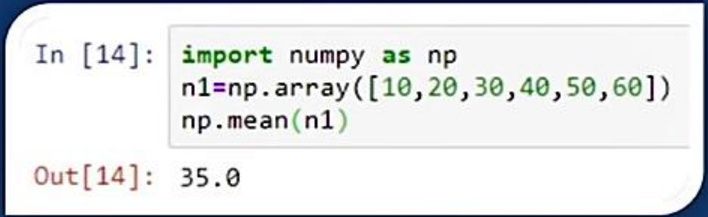


```
In [7]: import numpy as np
        n1=np.array([10,20,30])
        n1=n1/2
        n1

Out[7]: array([ 5., 10., 15.])
```

NumPy Math Functions

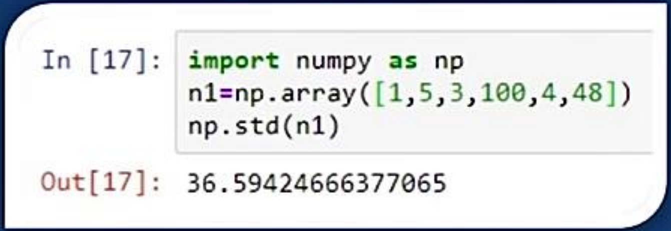
greatlearning
Learning for Life

Mean



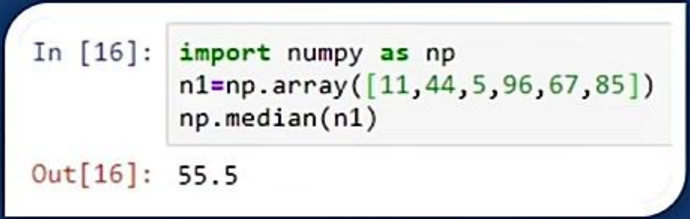
```
In [14]: import numpy as np
         n1=np.array([10,20,30,40,50,60])
         np.mean(n1)

Out[14]: 35.0
```

Standard Deviation



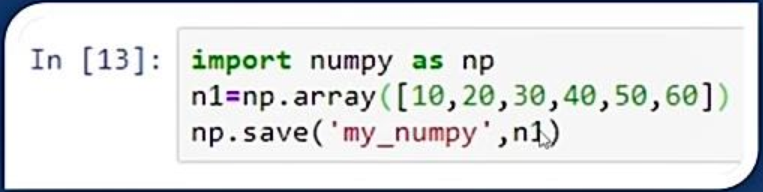
```
In [17]: import numpy as np
         n1=np.array([1,5,3,100,4,48])
         np.std(n1)

Out[17]: 36.59424666377065
```

Median



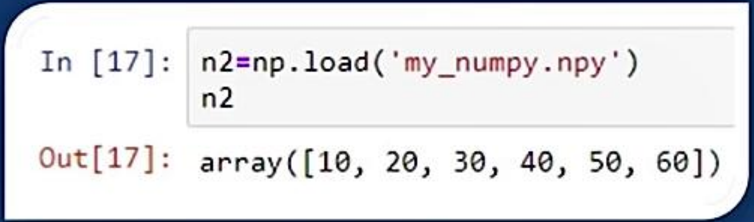
```
In [16]: import numpy as np
         n1=np.array([11,44,5,96,67,85])
         np.median(n1)

Out[16]: 55.5
```

NumPy Save & Load



```
In [13]: import numpy as np
         n1=np.array([10,20,30,40,50,60])
         np.save('my_numpy',n1)
```

→ Saving Numpy Array

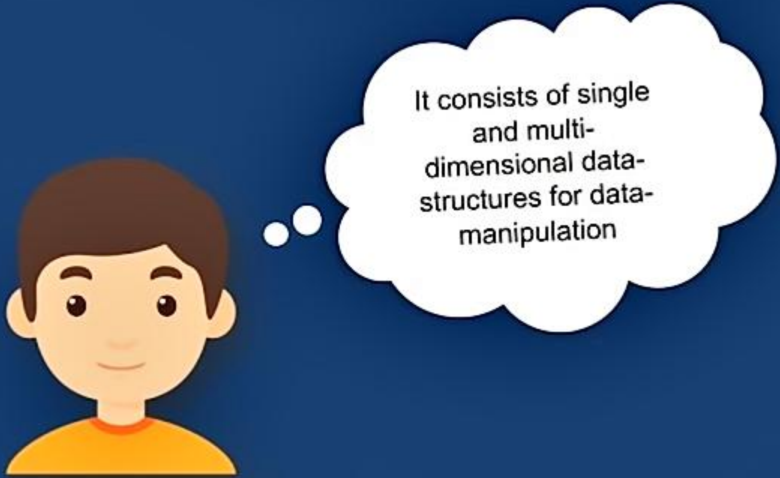


```
In [17]: n2=np.load('my_numpy.npy')
         n2

Out[17]: array([10, 20, 30, 40, 50, 60])
```

→ Loading Numpy Array

Python Pandas

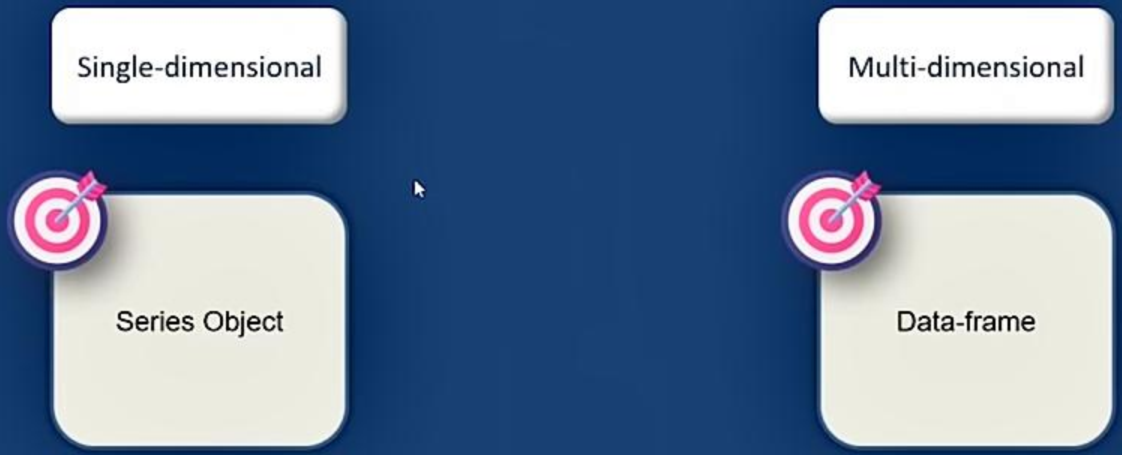
Pandas stands for Panel Data and is the core library for data manipulation and data analysis



It consists of single and multi-dimensional data-structures for data-manipulation

Pandas

Pandas Data-Structures



Single-dimensional

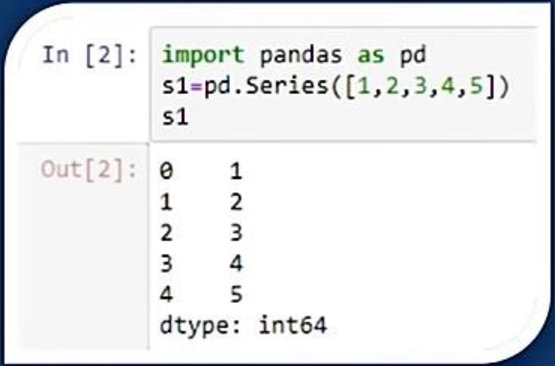
Series Object

Multi-dimensional

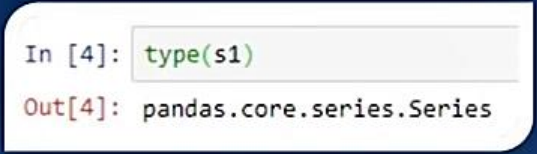
Data-frame

Pandas Series Object

Series Object is one-dimensional labeled array

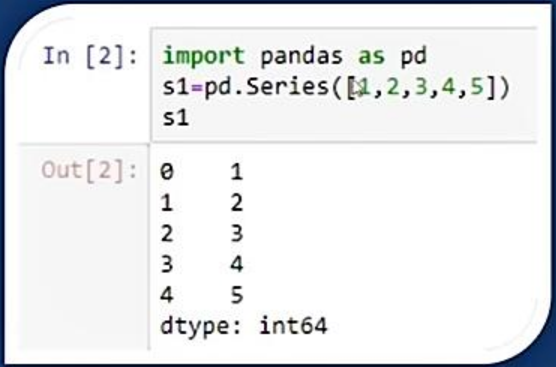


```
In [2]: import pandas as pd
        s1=pd.Series([1,2,3,4,5])
        s1

Out[2]: 0    1
        1    2
        2    3
        3    4
        4    5
        dtype: int64
```



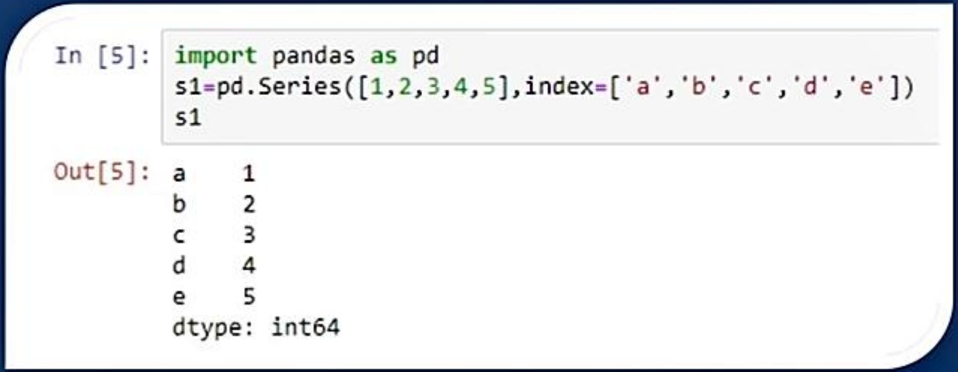
```
In [4]: type(s1)

Out[4]: pandas.core.series.Series
```

Changing Index



```
In [2]: import pandas as pd
        s1=pd.Series([1,2,3,4,5])
        s1
```

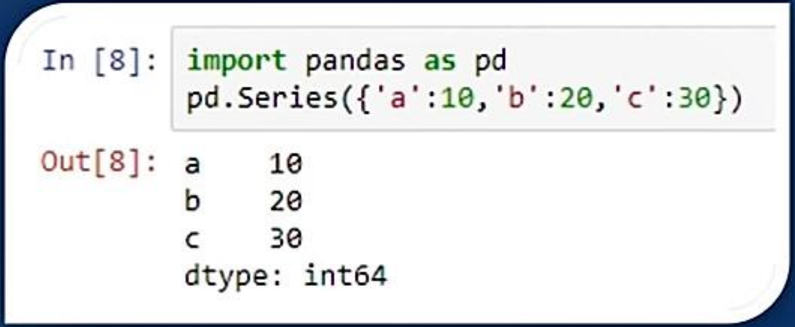
```
Out[2]: 0    1
        1    2
        2    3
        3    4
        4    5
        dtype: int64
```



```
In [5]: import pandas as pd
        s1=pd.Series([1,2,3,4,5],index=['a','b','c','d','e'])
        s1
```

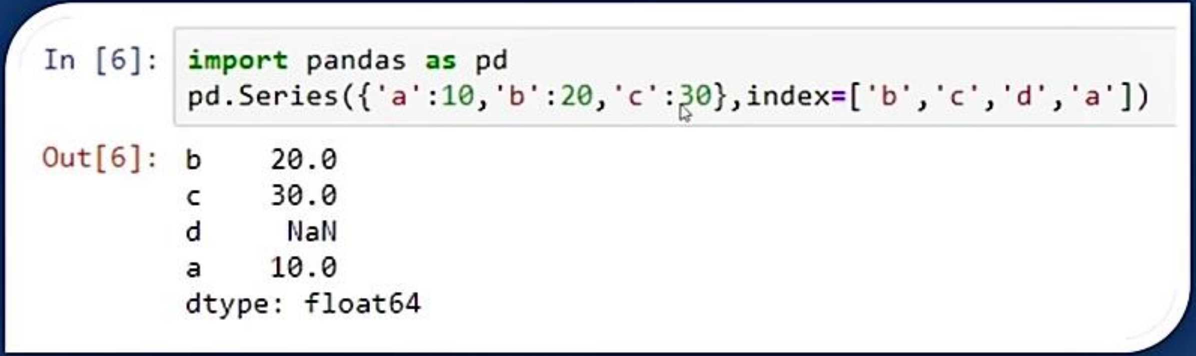
```
Out[5]: a    1
        b    2
        c    3
        d    4
        e    5
        dtype: int64
```

Series Object from Dictionary



```
In [8]:  import pandas as pd
          pd.Series({'a':10,'b':20,'c':30})

Out[8]:  a    10
         b    20
         c    30
         dtype: int64
```

Changing index position

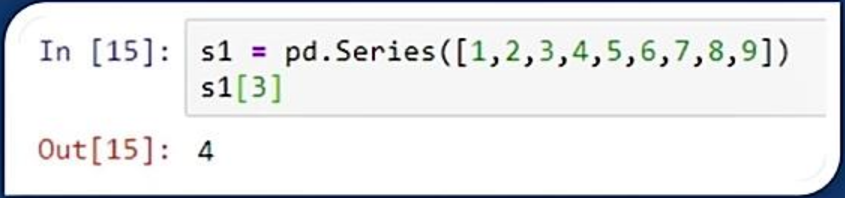


```
In [6]: import pandas as pd
        pd.Series({'a':10,'b':20,'c':30},index=['b','c','d','a'])

Out[6]: b    20.0
        c    30.0
        d     NaN
        a    10.0
        dtype: float64
```

Extracting Individual Elements

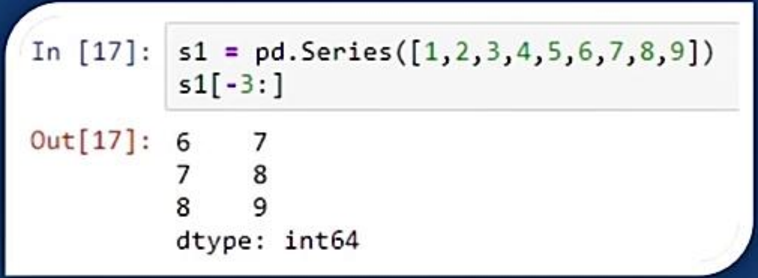
greatlearning
Learning for Life

Extracting a single element



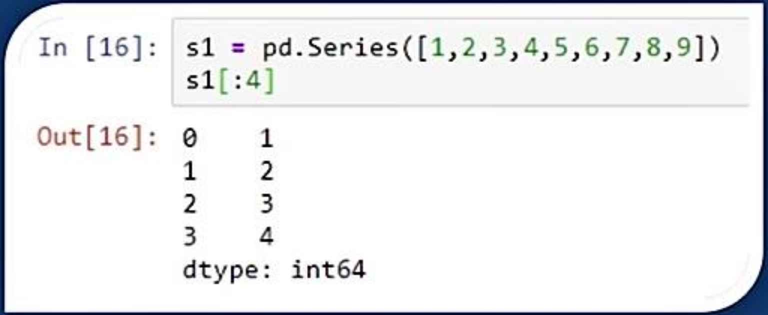
```
In [15]: s1 = pd.Series([1,2,3,4,5,6,7,8,9])
         s1[3]

Out[15]: 4
```

Extracting elements from back



```
In [17]: s1 = pd.Series([1,2,3,4,5,6,7,8,9])
         s1[-3:]

Out[17]: 6    7
         7    8
         8    9
         dtype: int64
```

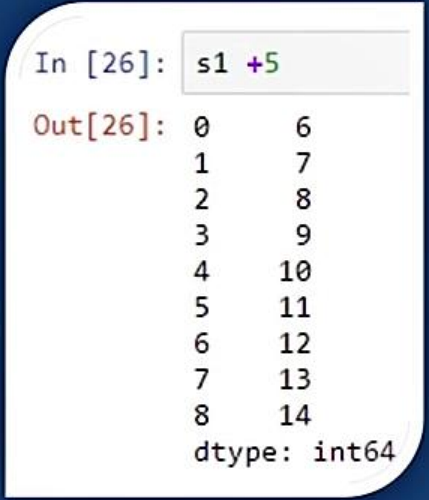
Extracting a sequence of elements



```
In [16]: s1 = pd.Series([1,2,3,4,5,6,7,8,9])
         s1[:4]

Out[16]: 0    1
         1    2
         2    3
         3    4
         dtype: int64
```

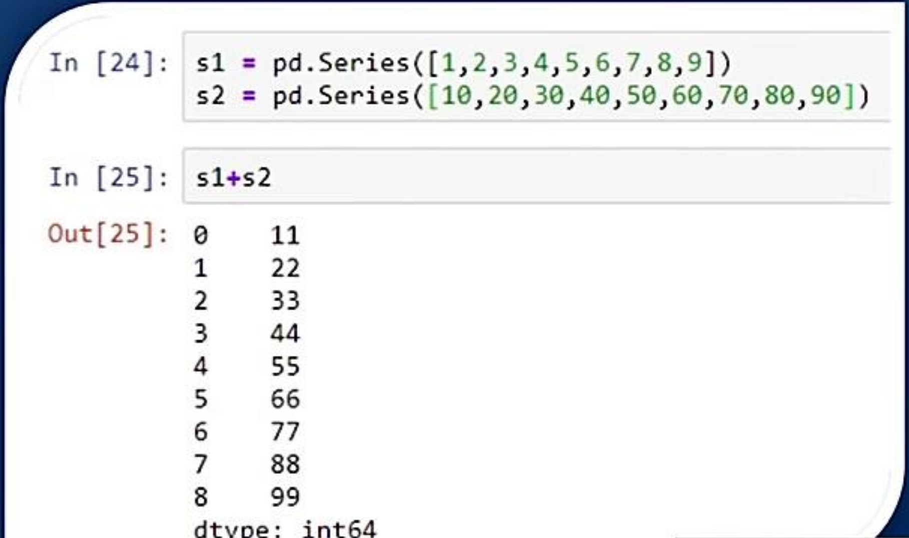
Basic Math Operations on Series

Adding a scalar value to Series Elements



```
In [26]: s1 +5

Out[26]: 0     6
         1     7
         2     8
         3     9
         4    10
         5    11
         6    12
         7    13
         8    14
         dtype: int64
```

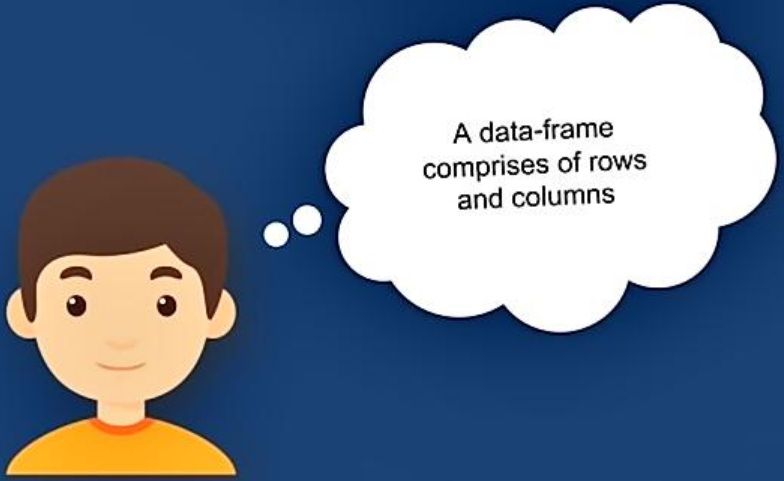
Adding two Series Objects

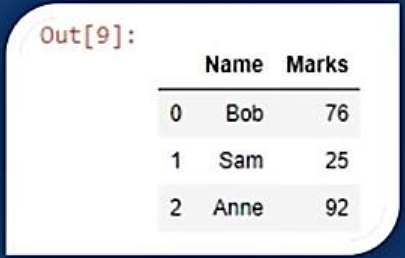


```
In [24]: s1 = pd.Series([1,2,3,4,5,6,7,8,9])
         s2 = pd.Series([10,20,30,40,50,60,70,80,90])

In [25]: s1+s2

Out[25]: 0    11
         1    22
         2    33
         3    44
         4    55
         5    66
         6    77
         7    88
         8    99
         dtype: int64
```

Pandas Dataframe

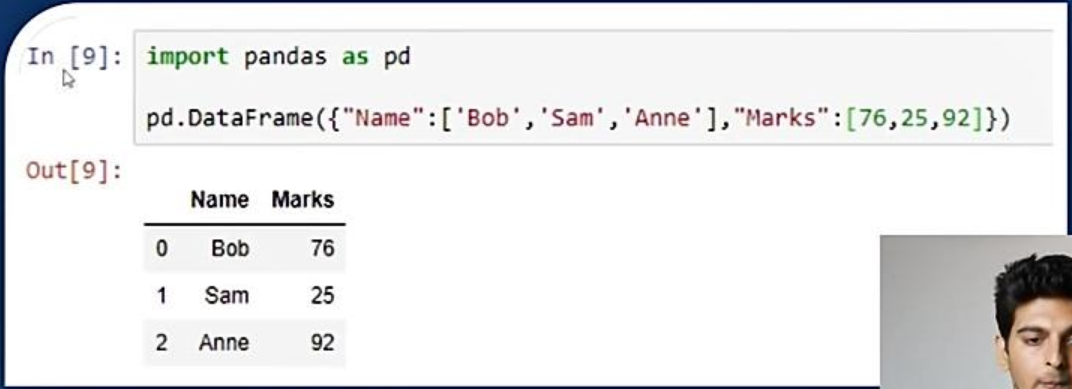
Dataframe is a 2-dimensional labelled data-structure





```
Out[9]:
        Name  Marks
    0    Bob     76
    1    Sam     25
    2   Anne     92
```

Creating a Dataframe



```
In [9]: import pandas as pd

        pd.DataFrame({"Name":['Bob','Sam','Anne'],"Marks":[76,25,92]})
```

```
Out[9]:
           Name  Marks
        0   Bob     76
        1   Sam     25
        2  Anne     92
```

Dataframe In-Built Functions

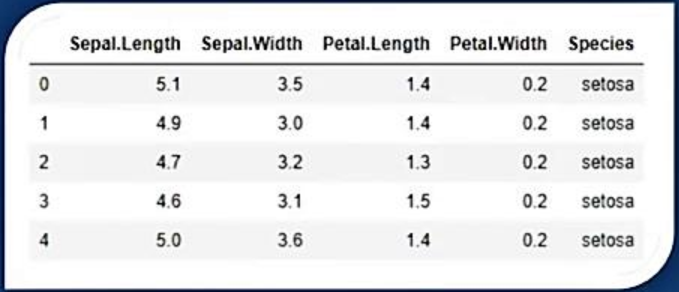
head()

shape()

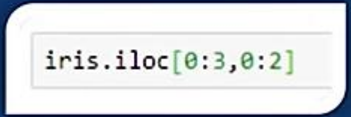
describe()

tail()

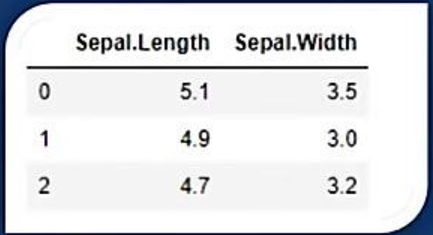
.iloc[]



	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

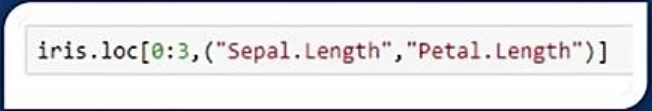


```
iris.iloc[0:3,0:2]
```

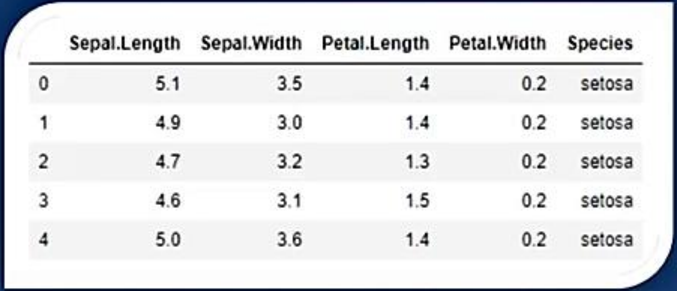


	Sepal.Length	Sepal.Width
0	5.1	3.5
1	4.9	3.0
2	4.7	3.2

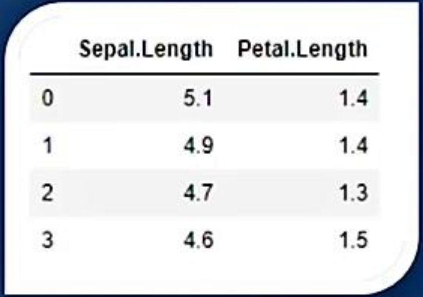
.loc[]



```
iris.loc[0:3,("Sepal.Length","Petal.Length")]
```

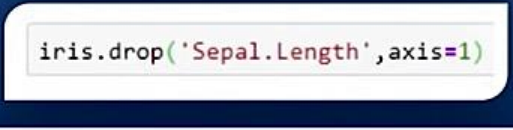


	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

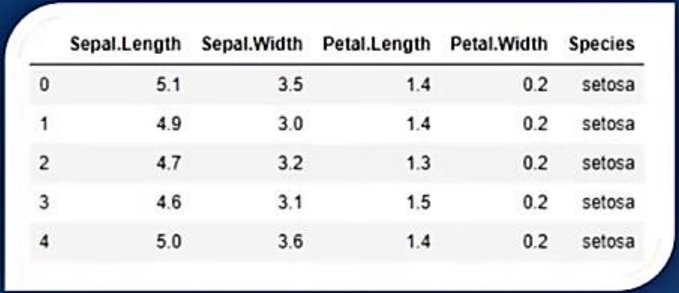


	Sepal.Length	Petal.Length
0	5.1	1.4
1	4.9	1.4
2	4.7	1.3
3	4.6	1.5

Dropping Columns

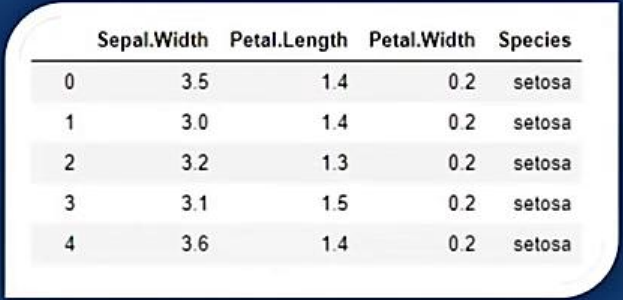


```
iris.drop('Sepal.Length',axis=1)
```



	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

→



	Sepal.Width	Petal.Length	Petal.Width	Species
0	3.5	1.4	0.2	setosa
1	3.0	1.4	0.2	setosa
2	3.2	1.3	0.2	setosa
3	3.1	1.5	0.2	setosa
4	3.6	1.4	0.2	setosa