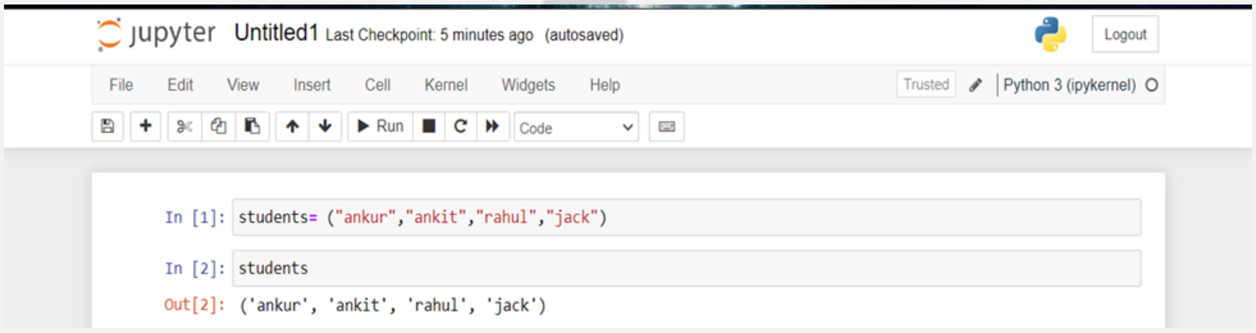
Python Session Summary 12-06-2023

- In **Tuple**, we can store the data we don't want to be changed or mutated in the future because tuple are immutable and the data cannot be changed once saved.

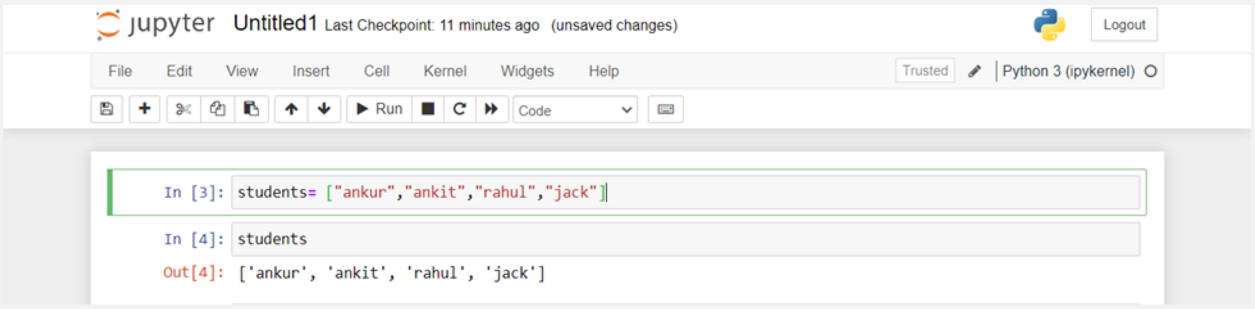
- In List, we have the power to change or mutate any data. We can perform CRUD operations in the list efficiently.

- Example of creating a Tuple:


jupyter Untitled1 Last Checkpoint: 5 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

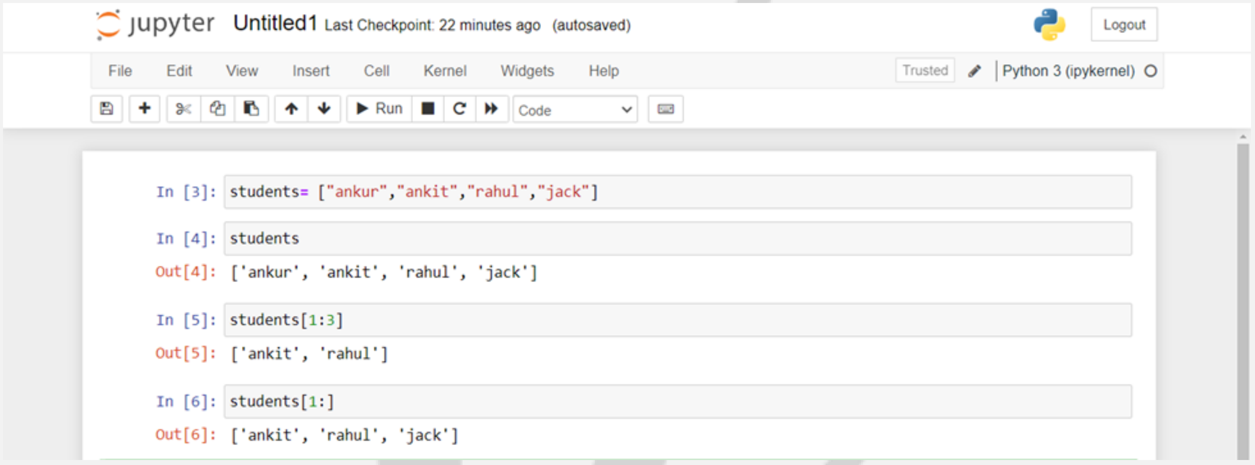
```
In [1]: students= ("ankur","ankit","rahul","jack")

In [2]: students
Out[2]: ('ankur', 'ankit', 'rahul', 'jack')
```

- Example of creating a List:


jupyter Untitled1 Last Checkpoint: 11 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

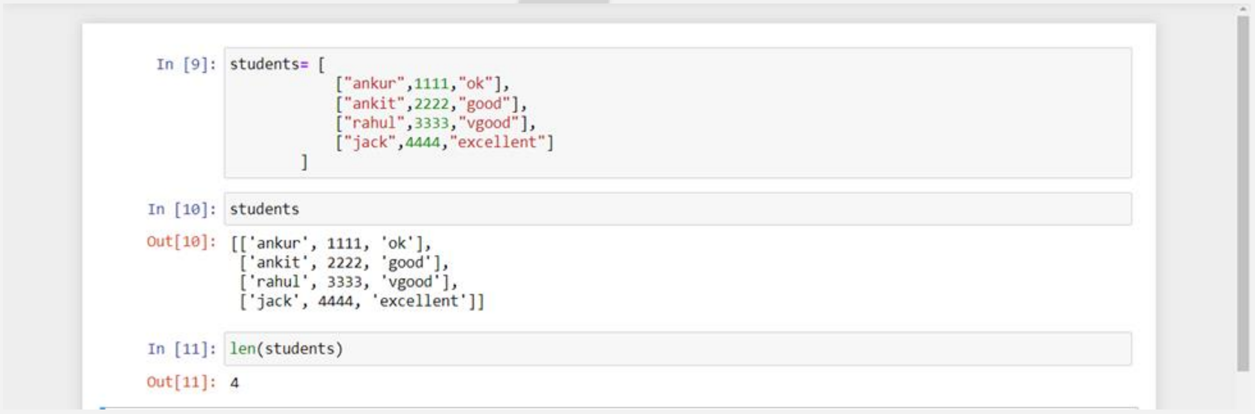
```
In [3]: students= ["ankur","ankit","rahul","jack"]

In [4]: students
Out[4]: ['ankur', 'ankit', 'rahul', 'jack']
```

- We can perform slicing operations in a list:


jupyter Untitled1 Last Checkpoint: 22 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

```
In [3]: students= ["ankur","ankit","rahul","jack"]

In [4]: students
Out[4]: ['ankur', 'ankit', 'rahul', 'jack']

In [5]: students[1:3]
Out[5]: ['ankit', 'rahul']

In [6]: students[1:]
Out[6]: ['ankit', 'rahul', 'jack']
```

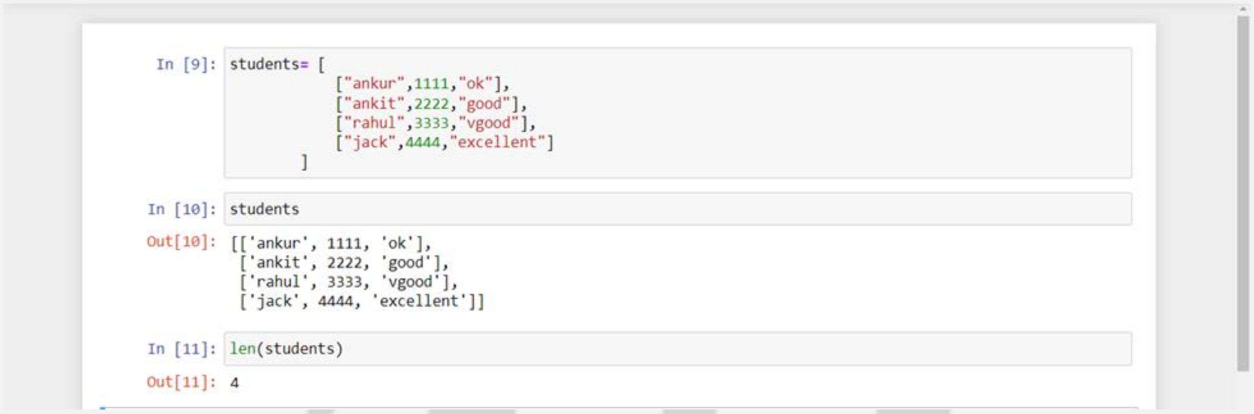
- Suppose we have to store a group of information about any individual then we can use a list inside a list or a Nested list.



```
In [9]: students= [
                    ["ankur",1111,"ok"],
                    ["ankit",2222,"good"],
                    ["rahul",3333,"vgood"],
                    ["jack",4444,"excellent"]
                ]

In [10]: students
Out[10]: [['ankur', 1111, 'ok'],
          ['ankit', 2222, 'good'],
          ['rahul', 3333, 'vgood'],
          ['jack', 4444, 'excellent']]

In [11]: len(students)
Out[11]: 4
```

- Above is an example of a 2-D list because we are storing data in both rows and columns.

- In the list, we have challenges, We can only print the row content of a list. The list does not allow column-wise operation.



```
In [9]: students= [
                    ["ankur",1111,"ok"],
                    ["ankit",2222,"good"],
                    ["rahul",3333,"vgood"],
                    ["jack",4444,"excellent"]
                ]

In [10]: students

Out[10]: [['ankur', 1111, 'ok'],
          ['ankit', 2222, 'good'],
          ['rahul', 3333, 'vgood'],
          ['jack', 4444, 'excellent']]

In [11]: len(students)

Out[11]: 4
```

- In the above example of a list, we cannot print any of the columns because the list doesn't support this operation.

- We can solve this problem with the help of the NumPy. NumPy is a library in Python that provides a function named Array. In arrays, we can perform column-wise operations on the data.

- **NumPy:** NumPy (Numerical Python) is an open-source Python library that's used in almost every field of science and engineering.

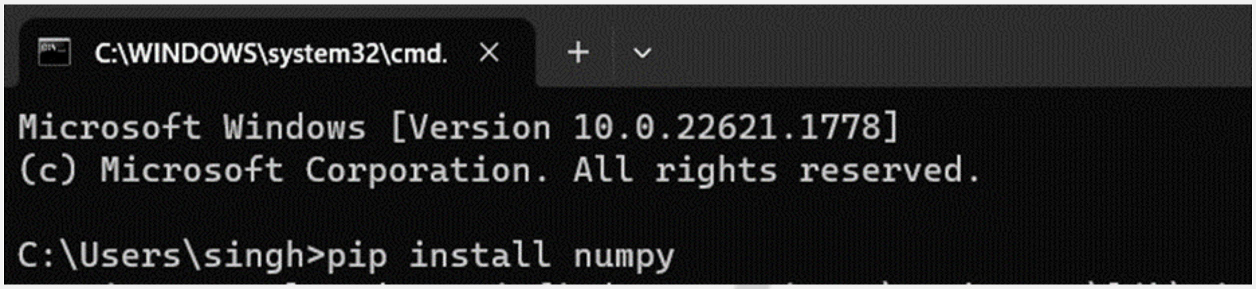
- It's the universal standard for working with numerical data in Python, and it's at the core of the scientific Python and PyData ecosystem.

- **Installing NumPy:**

  ➢ To install NumPy, we strongly recommend using a scientific Python distribution.

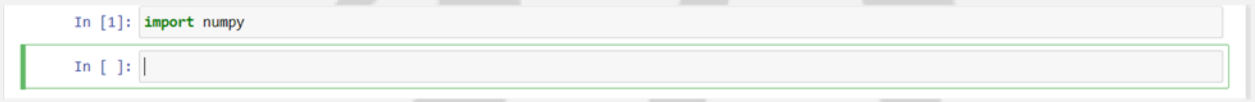
➢ If you're looking for the full instructions for installing NumPy on your operating system, see Installing NumPy.

➢ If you already have Python, you can install NumPy with:



```
C:\WINDOWS\system32\cmd.

Microsoft Windows [Version 10.0.22621.1778]
(c) Microsoft Corporation. All rights reserved.

C:\Users\singh>pip install numpy
```

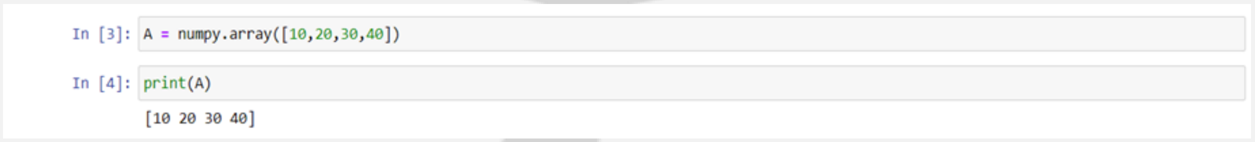
- How to import NumPy



```
In [1]: import numpy

In [ ]: 
```

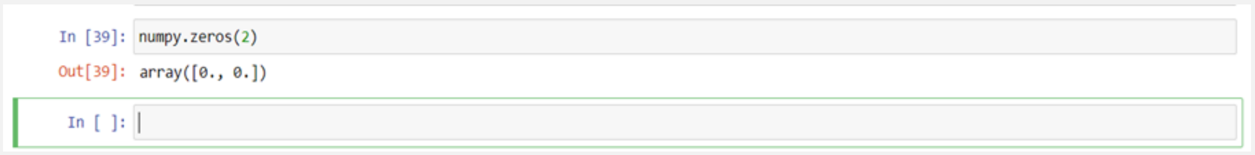
- How to create a basic array.

➢ All you need to do to create a simple array: Pass a list to it. If you choose to, you can also specify the type of data in your list.



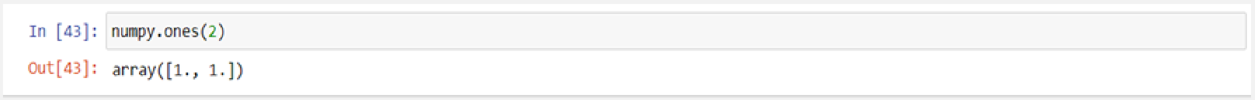
```
In [3]: A = numpy.array([10,20,30,40])

In [4]: print(A)
        [10 20 30 40]
```

➢ Create a zero array using the zeros ( ) function.



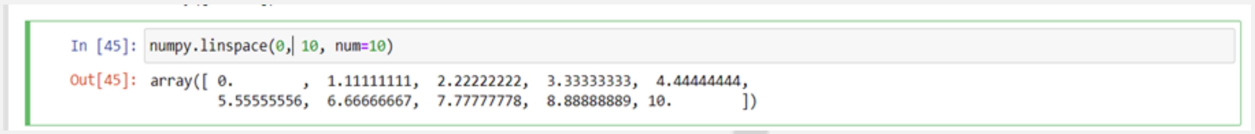
```
In [39]: numpy.zeros(2)
Out[39]: array([0., 0.])

In [ ]: 
```

➢ Create one's array using the ones ( ) function.



```
In [43]: numpy.ones(2)
Out[43]: array([1., 1.])
```

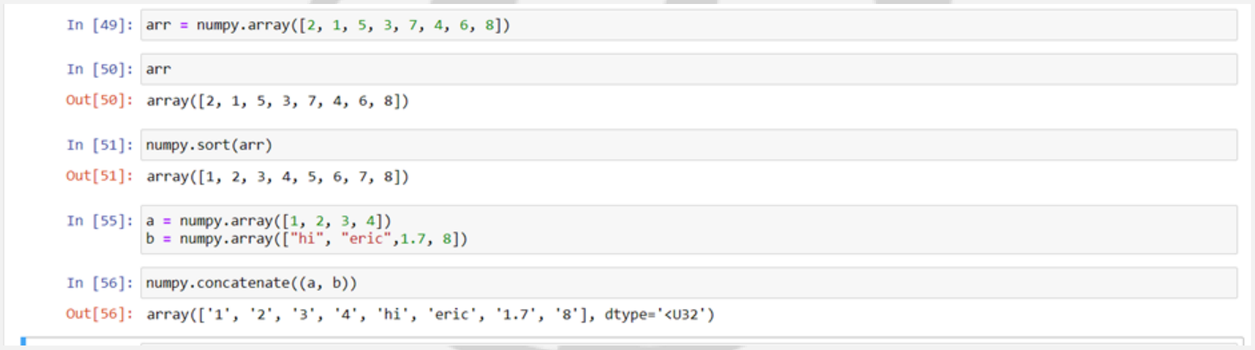
➢ You can also use np.linspace() to create an array with values that are spaced linearly in a specified interval:



```
In [45]: numpy.linspace(0, 10, num=10)
Out[45]: array([ 0.        ,  1.11111111,  2.22222222,  3.33333333,  4.44444444,
                 5.55555556,  6.66666667,  7.77777778,  8.88888889, 10.        ])
```

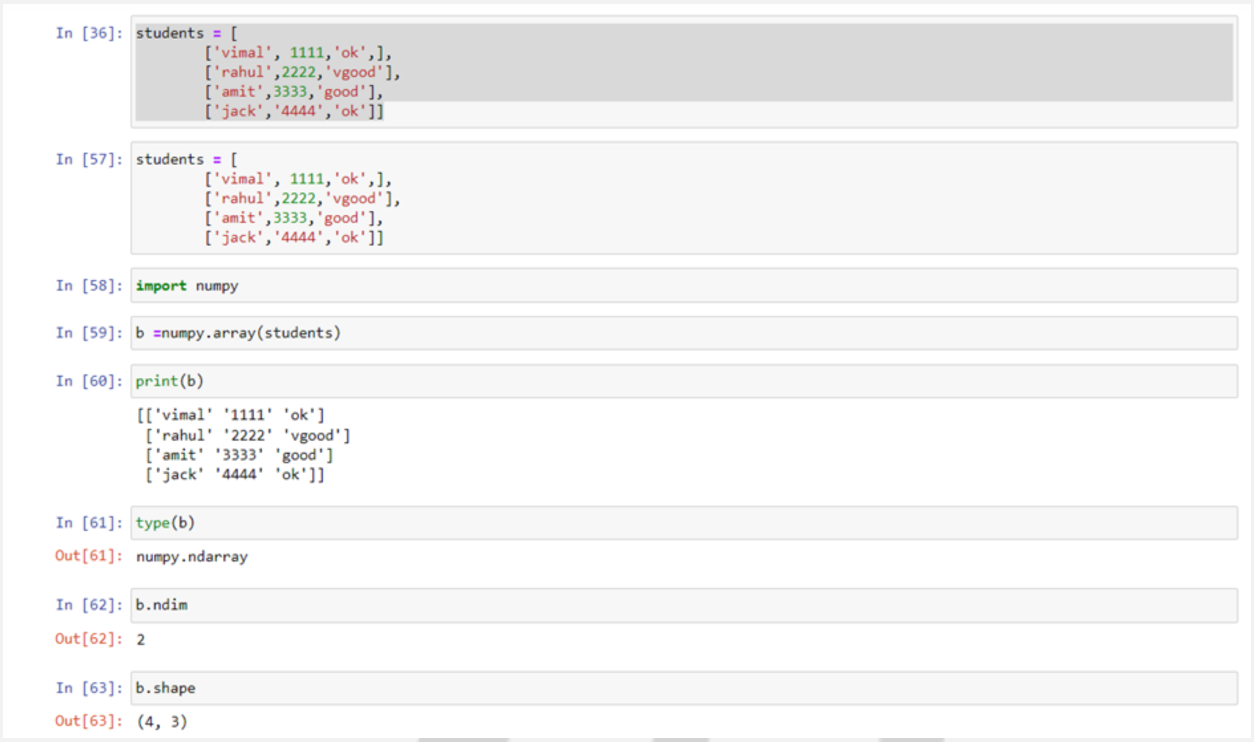
● Adding, removing, and sorting elements:

➢ This section covers np.sort(), np.concatenate()



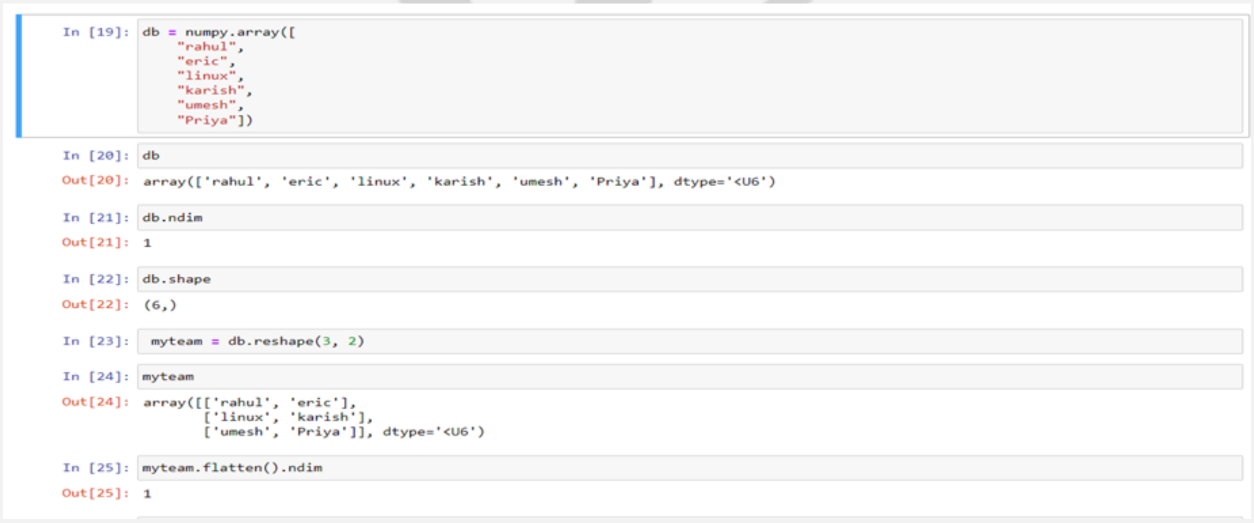
```
In [49]: arr = numpy.array([2, 1, 5, 3, 7, 4, 6, 8])

In [50]: arr
Out[50]: array([2, 1, 5, 3, 7, 4, 6, 8])

In [51]: numpy.sort(arr)
Out[51]: array([1, 2, 3, 4, 5, 6, 7, 8])

In [55]: a = numpy.array([1, 2, 3, 4])
         b = numpy.array(["hi", "eric",1.7, 8])

In [56]: numpy.concatenate((a, b))
Out[56]: array(['1', '2', '3', '4', 'hi', 'eric', '1.7', '8'], dtype='<U32')
```

➢ The shape and size of an array: covers ndarray.ndim, ndarray.size, ndarray.shape



```
In [36]: students = [
                 ['vimal', 1111,'ok',],
                 ['rahul',2222,'vgood'],
                 ['amit',3333,'good'],
                 ['jack','4444','ok']]

In [57]: students = [
                 ['vimal', 1111,'ok',],
                 ['rahul',2222,'vgood'],
                 ['amit',3333,'good'],
                 ['jack','4444','ok']]

In [58]: import numpy

In [59]: b =numpy.array(students)

In [60]: print(b)

         [['vimal' '1111' 'ok']
          ['rahul' '2222' 'vgood']
          ['amit' '3333' 'good']
          ['jack' '4444' 'ok']]

In [61]: type(b)

Out[61]: numpy.ndarray

In [62]: b.ndim

Out[62]: 2

In [63]: b.shape

Out[63]: (4, 3)
```

➢ flatten() method in Python is used to return a copy of a given array in such a way that it is collapsed into one dimension.



```
In [19]: db = numpy.array([
             "rahul",
             "eric",
             "linux",
             "karish",
             "umesh",
             "Priya"])

In [20]: db
Out[20]: array(['rahul', 'eric', 'linux', 'karish', 'umesh', 'Priya'], dtype='<U6')

In [21]: db.ndim
Out[21]: 1

In [22]: db.shape
Out[22]: (6,)

In [23]:  myteam = db.reshape(3, 2)

In [24]: myteam
Out[24]: array([['rahul', 'eric'],
                ['linux', 'karish'],
                ['umesh', 'Priya']], dtype='<U6')

In [25]: myteam.flatten().ndim
Out[25]: 1
```