In [1]:

```
import numpy as np
```

In [2]:

```
#We create a list called "Distance" and "Time"
distance = [10,15,17,26]
time = [.30,.47,.55,1.20]
distance_nw = np.array(distance)
time_nw = np.array(time)
```

In [3]:

```
speed=distance_nw/time_nw
```

In [4]:

```
speed
```

Out[4]:

```
array([33.33333333, 31.91489362, 30.90909091, 21.66666667])
```

In [5]:

```
#Deleting Elements of a list
list1 =[2,4,5,8,9,7]
```

In [8]:

```
del list1[2:5]
```

In [16]:

```
print (list1)
```

```
[2, 4, 7]
```

In [11]:

```
list1 =[2,4,5,8,9,7]
```

In [12]:

```
del list1[2:5]
```

In [13]:

```
print(list1)
```

```
[2, 4, 7]
```

In [17]:

```
#In NumPy, a scalar is any object that you put in an array.
#NumPy ensures all scalars in an array have same types.
#The basic ndarray is created using an array function in NumPy as follows –

#numpy.array
```

In [18]:

```
#The NumPy array object has a property called dtype that returns the data type of
import numpy as np

arr1 = np.array([8, 9, 3, 4])

print(arr1.dtype)
```

```
int32
```

In [19]:

```
#Get the data type of an array containing strings:

import numpy as np

arr2 = np.array(['apple', 'banana', 'cherry'])

print(arr2.dtype)       #U64' is a 64 character unicode. That's a normal string in
```

```
<U6
```

In [20]:

```
arr2 = np.array([[1, 2, 3], [4, 5, 6]])

arr2.dtype
```

```
Out[20]: dtype('int32')

In [23]: arr2

Out[23]: array([[1, 2, 3],
                [4, 5, 6]])

In [24]: import numpy as np

In [25]: arr1 = np.array([1,2,3,4])

In [27]: bcd = arr1.ndim

In [28]: bcd

Out[28]: 1

In [29]: import numpy as np

         arr21 = [[1, 2, 3], [4, 5, 6]]

         efg = np.ndim(arr21)
         print (efg)

         2

In [30]: arr21

Out[30]: [[1, 2, 3], [4, 5, 6]]

In [31]: np_city= np.array(['NYC','LA','Miami','Houston'])

In [34]: print(np_city.shape)

         (4,)

In [35]: np_city.ndim

Out[35]: 1

In [36]: np_city_with_state = np.array([['NYC','LA','Miami','Houston'],['NY','CA','FL','TX'

In [37]: np_city_with_state.ndim

Out[37]: 2

In [38]: #ndarray.shape #This array attribute returns a tuple consisting of array dimension

In [39]: import numpy as np

In [41]: l= np.array([[1,2,3],[4,5,6]])

In [42]: print(l.shape)

         (2, 3)

In [45]: # this resizes the ndarray l
         import numpy as np
```

```
m=np.array([[1,2,3],[4,5,6]])
m.shape=(3,2)
print(m)
```

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

In [46]:

```
#Reshaping arrays
#Reshaping means changing the shape of an array.
#The shape of an array is the number of elements in each dimension.
#By reshaping we can add or remove dimensions or change number of elements in each
```

In [50]:

```
#Reshape the following 1-D array with 12 elements into a 2-D array.

#The outermost dimension will have 4 arrays, each with 3 elements:

import numpy as np
arr51 = np.array([1,2,3,4,5,6,7,8,9,10,11,12])
m = arr51.reshape (3,4)
print(m)
```

```
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
```

In [52]:

```
#Convert the following 1-D array with 12 elements into a 3-D array.

#The outermost dimension will have 2 arrays that contains 3 arrays, each with 2 ele
import numpy as np
arr52 = np.array([1,2,3,4,5,6,7,8,9,10,11,12])
n= arr52.reshape(2,2,3)
print(n)
```

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

 [[ 7  8  9]
  [10 11 12]]]
```

In [ ]:

In [ ]: