convention for importing numpy

```
import numpy as np

arr = [6, 7, 8, 9]
print(type(arr)) # prints <class 'list'>

a = np.array(arr)
print(type(a))  # prints <class 'numpy.ndarray'>
print(a.shape)  # prints (4,) - a is a 1d array with 4 items
print(a.dtype)  # prints int64

# get the dimension of a with ndim
print(a.ndim)   # prints 1

b = np.array([[1, 2, 3], [4, 5, 6]])
print(b)        # prints [[1 2 3]
                         # [4 5 6]]
print(b.ndim)   # prints 2
b.shape         # prints (2, 3) - b a 2d array with 2 rows and     3 columns
import numpy as np

arr = [6, 7, 8, 9]
print(type(arr)) # prints <class 'list'>

a = np.array(arr)
print(type(a))  # prints <class 'numpy.ndarray'>
print(a.shape)  # prints (4,) - a is a 1d array with 4 items
print(a.dtype)  # prints int64

# get the dimension of a with ndim
print(a.ndim)   # prints 1

b = np.array([[1, 2, 3], [4, 5, 6]])
print(b)        # prints [[1 2 3]
                         #[4 5 6]]
print(b.ndim)   # prints 2
b.shape         # prints (2, 3) - b a 2d array with 2 rows and     3 columns
```

```
<class 'list'>
<class 'numpy.ndarray'>
(4,)
int64
1
[[1 2 3]
 [4 5 6]]
2
```

```
<class 'list'>
<class 'numpy.ndarray'>
(4,)
int64
1
[[1 2 3]
 [4 5 6]]
2
(2, 3)
```

Inbuilt functions that can be used to initialize numpy

```
# a 2x3 array with random values
np.random.random((2, 3))
# a 2x3 array of zeros
np.zeros((2, 3))
# a 2x3 array of ones
np.ones((2, 3))
# a 3x3 identity matrix
np.identity(3)
```

```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

Intra-operability of arrays and scalars

```
c = np.array([[9.0, 8.0, 7.0], [1.0, 2.0, 3.0]])
d = np.array([[4.0, 5.0, 6.0], [9.0, 8.0, 7.0]])
```

```
c + d # prints array([[13., 13., 13.],
             # [ 10., 10., 10.]])
```

```
c * d # prints array([[36., 40., 42.],
                #[9., 16., 21.]])
5 / d #prints array([[1.25      , 1.        , 0.83333333],
              #[0.55555556, 0.625     , 0.71428571]])

c ** 2 # prints array([[81., 64., 49.],[ 1.,  4.,  9.]])
```

```
array([[81., 64., 49.],
       [ 1.,  4.,  9.]])
```

Indexing with arrays & Using arrays for data processing

```
#The elements in the example arrays above can be accessed by indexing like lists in Python su
a[0] # prints 6
a[3] # prints 9
b[0, 0] #prints 1
b[1, 2] # prints 6
c[0, 1] # prints 8.

#Elements in arrays  can also be retrieved by slicing rows and columns or a combination of in
d[1,  0:2] # prints array([9., 8.])

e = np.array([[10, 11, 12],[13, 14, 15],
              [16, 17, 18],[19, 20, 21]])

# slicing
e[:3, :2] #prints array([[10, 11], [13, 14],[16, 17]])

#There are other advanced methods of indexing which are shown below.
# integer indexing
e[[2, 0, 3, 1],[2, 1, 0, 2]] #prints array([18, 11, 19, 15])

# boolean indexing meeting a specified condition
e[e>15] #prints array([16, 17, 18, 19, 20, 21])
```

```
array([16, 17, 18, 19, 20, 21])
```

+ Code + Text