Basic Data Science

1. Import the numpy package under the name `np`

In [2]:

```
import numpy as np
```

2. Create a null vector of size 20

In [3]:

```
y=np.zeros(20)
y
```

Out[3]:

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

3. Create a Ones Vector of size 20

In [4]:

```
np.ones(20)
```

Out[4]:

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

4. Create a boolean array of 3X4.

In [6]:

```
y=np.random.randint(1,2,size=(3,4),dtype=bool)
y
```

Out[6]:

```
array([[ True,  True,  True,  True],
       [ True,  True,  True,  True],
       [ True,  True,  True,  True]])
```

5. Create a vector with values ranging from 100 to 200 of float64 data type

In [7]:

```
y=np.arange(100,200,dtype=float)
y
```

Out[7]:

```
array([100., 101., 102., 103., 104., 105., 106., 107., 108., 109., 110.,
       111., 112., 113., 114., 115., 116., 117., 118., 119., 120., 121.,
       122., 123., 124., 125., 126., 127., 128., 129., 130., 131., 132.,
       133., 134., 135., 136., 137., 138., 139., 140., 141., 142., 143.,
       144., 145., 146., 147., 148., 149., 150., 151., 152., 153., 154.,
       155., 156., 157., 158., 159., 160., 161., 162., 163., 164., 165.,
       166., 167., 168., 169., 170., 171., 172., 173., 174., 175., 176.,
       177., 178., 179., 180., 181., 182., 183., 184., 185., 186., 187.,
       188., 189., 190., 191., 192., 193., 194., 195., 196., 197., 198.,
       199.])
```

6. Create an array of five values evenly spaced between 0 and 1

In [8]:

```
y=np.linspace(0,1,5)
y
```

Out[8]:

```
array([0.  , 0.25, 0.5 , 0.75, 1.  ])
```

7. Reverse a given Vector

In [9]:

```
myarray = np.array([9, 8, 7, 6, 5, 4, 3, 2, 1, 0])
myarray=myarray[::-1]
print(myarray)
```

```
[0 1 2 3 4 5 6 7 8 9]
```

8. Find indices of non-zero elements from [12,34,0,4,0,2,3,0,123]

In [10]:

```
y=np.array([12,34,0,4,0,2,3,0,123])
np.nonzero(y)
```

Out[10]:

```
(array([0, 1, 3, 5, 6, 8], dtype=int64),)
```

9. Replace all even numbers in given arr vector with -1

In [11]:

```
arr = np.array([1,2,3,4,5,6,7,8,9,10,11,12,13,14])
print(np.where(arr%2==0,-1,arr))
```

```
[ 1 -1  3 -1  5 -1  7 -1  9 -1 11 -1 13 -1]
```

10. Create a 5x3 array with random values (In - between 100 to 300) and find the minimum and maximum values (Hints : Use np.random.random)

In [13]:

```
r = np.random.randint(100,300,size=(5,3))
print(r)
print('The maximum Value is:',np.max(r))
print('The minimum Value is:',np.min(r))
```

```
[[213 129 208]
 [239 280 142]
 [225 180 110]
 [132 186 273]
 [297 108 293]]
The maximum Value is: 297
The minimum Value is: 108
```

11. Create a random vector of size 30 and find the mean value

In [14]:

```
y=np.random.randint(1,100,30)
print(y)
print(np.mean(y))
```

```
[ 1 16 65 15 66 71 56 17 26 35 96 21 44 58 42 81  5 16 35  8 13 52 92 83
 31 21  5 77 99 24]
42.36666666666667
```

12. What is the result of the following expression?

```
0 * np.nan
np.nan == np.nan
np.inf > np.nan
np.nan - np.nan
np.nan in set([np.nan])
0.3 == 3 * 0.1
```

In [15]:

```
print(0 * np.nan)
print(np.nan == np.nan)
print(np.inf > np.nan)
print(np.nan - np.nan)
print(np.nan in set([np.nan]))
print(0.3 == 3 * 0.1)
```

```
nan
False
False
nan
True
False
```

13. Normalize a 5x5 random matrix (Hints - fourmula (x - mean) / std)

In [17]:

```
Z = np.random.random((5,5))
zmean, zstd = np.mean(Z), np.std(Z)
Z=(Z-zmean)/(zstd)
Z
```

Out[17]:

```
array([[-0.6155269 ,  0.78486174,  0.92922348,  1.45679594,  0.10106439],
       [-0.97190059,  0.95435563, -1.34884758,  0.50452436, -1.39884394],
       [-1.63046292, -1.54888924, -0.59079637,  0.74957396,  1.23032706],
       [-0.0596961 ,  1.22717068, -1.08609602,  0.94524756,  1.00384458],
       [ 0.04368942, -1.44756848,  0.47905731, -0.43956371,  0.72845574]])
```

14. Multiply a 5x3 matrix by a 3x2 matrix (real matrix product)

In [22]:

```
y1=np.random.randint(1,10,size=(5,3))
y2=np.random.randint(1,10,size=(3,2))
print("random matrix:",y1)
print("random matrix:",y2)
y3=np.dot(y1,y2)
print("multiplication of matrix y1 and y2 :",y3)
```

```
random matrix: [[3 2 9]
 [7 9 2]
 [2 6 5]
 [3 6 5]
 [6 7 5]]
random matrix: [[2 3]
 [3 4]
 [3 5]]
multiplication of matrix y1 and y2 : [[39 62]
 [47 67]
 [37 55]
 [39 58]
 [48 71]]
```

15. How to find common values between two arrays?

In [21]:

```
y=np.random.randint(10,20,20)
s=np.random.randint(20,40,10)
print(y)
print(s)
print(np.intersect1d(y,s))
```

```
[18 10 15 11 10 16 18 11 13 16 12 17 12 15 13 10 10 15 18 14]
[37 36 22 39 38 31 35 21 22 24]
[]
```

16. Convert a 1D array to a 2D array with 4 rows

In [23]:

```
one = np.arange(2,22)
one.reshape(4,5)
```

Out[23]:

```
array([[ 2,  3,  4,  5,  6],
       [ 7,  8,  9, 10, 11],
       [12, 13, 14, 15, 16],
       [17, 18, 19, 20, 21]])
```

17. Create two array (a and b) and stack them vertically?.(concatenate vertically?)

In [24]:

```
a=np.random.randint(1,10,size=(2,2))
b=np.random.randint(1,10,size=(2,2))
print(a)
print('\n')
print(b)
print('\n')
print('The Stacked arrays are:')
print(np.concatenate((a,b),axis=0))
```

```
[[9 3]
 [2 2]]


[[5 6]
 [2 6]]


The Stacked arrays are:
[[9 3]
 [2 2]
 [5 6]
 [2 6]]
```

18. Create two 2Darray (a and b) and stack them horizontally.(concatenate horizontally)

In [26]:

```
a=np.random.randint(1,10,size=(2,2))
b=np.random.randint(1,10,size=(2,2))
print(a)
print('\n')
print(b)
print('\n')
print('Arrays that has been stacked:')
print(np.concatenate((a,b),axis=1))
```

```
[[3 6]
 [2 5]]


[[6 2]
 [3 8]]


Arrays that has been stacked:
[[3 6 6 2]
 [2 5 3 8]]
```

19. Create a 2darray of 4X4 and swap 2nd and 4th column .

In [27]:

```
my_array=np.random.randint(1,50,size=(4,4))
print(my_array)
my_array[:,[1, 3]] = my_array[:,[3, 1]]
my_array
```

```
[[30 34 45 20]
 [30  5 37 44]
 [28 18 13 22]
 [29  3 28  6]]
```

Out[27]:

```
array([[30, 20, 45, 34],
       [30, 44, 37,  5],
       [28, 22, 13, 18],
       [29,  6, 28,  3]])
```

20. Create a 2darray of 4X4 and swap 2nd and 4th rows

In [28]:

```
my_array=np.random.randint(1,50,size=(4,4))
print(my_array)
my_array[[1,3],:] = my_array[[3,1],:]
my_array
```

```
[[13 29 10 48]
 [ 8 25 49  8]
 [13 33  1 34]
 [25 12 36 16]]
```

Out[28]:

```
array([[13, 29, 10, 48],
       [25, 12, 36, 16],
       [13, 33,  1, 34],
       [ 8, 25, 49,  8]])
```

In [ ]:

```

```