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 [4]:

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

Out[4]:

```
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 [5]:

```
x = np.ones(20)
x
```

Out[5]:

```
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]:

```
x = np.random.randint(0,2,size=(3,4),dtype=bool)
x
```

Out[6]:

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

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

In [8]:

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

Out[8]:

```
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 [9]:

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

Out[9]:

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

7. Reverse a given Vector

In [10]:

```
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 [11]:

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

Out[11]:

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

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

In [12]:

```
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 max value is: ',np.max(r))
print('The min value is: ',np.min(r))
```

```
[[146 229 175]
 [128 184 150]
 [279 125 129]
 [230 228 140]
 [156 216 146]]
The max value is:  279
The min value is:  125
```

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

In [14]:

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

```
[75 63  7 36 41  2 60 24 51 63 98 52 75 10 62 11 51 55 82 19 37  4 52 94
 48 26 20 24 23 18]
42.766666666666666
```

12. What is the result of the following expression?

```
```python
print(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 [16]:

```
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 [18]:

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

Out[18]:

```
array([[-0.74175462, -0.53214471,  0.60032807,  0.47945713,  0.70364376],
       [ 0.79331945,  0.87657027, -0.06429886, -1.00409855,  0.0183671 ],
       [-1.28454962, -1.75982194, -0.49158557, -0.17785839,  0.73319595],
       [ 0.99433533, -1.67882423,  0.47869177,  0.66517983,  1.25972628],
       [-1.27141929,  0.8275982 ,  1.27592506,  1.21091212, -1.91089455]])
```

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

In [19]:

```
m1=np.random.randint(1,10,size=(5,3))
m2=np.random.randint(1,10,size=(3,2))
print(m1)
print(m2)
m3=np.dot(m1,m2)
m3
```

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

Out[19]:

```
array([[154,  93],
       [ 65,  33],
       [108,  68],
       [ 68,  35],
       [100,  64]])
```

15. How to find common values between two arrays?

In [21]:

```
x = np.random.randint(10,30,10)
y = np.random.randint(20,40,10)
print(x)
print(y)
print(np.intersect1d(x,y))

```

```
[14 20 13 22 19 29 15 10 10 22]
[33 30 38 32 24 21 21 21 20 34]
[20]
```

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

In [22]:

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

Out[22]:

```
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]:

```
x = np.random.randint(1,10,size=(2,2))
y = np.random.randint(1,10,size=(2,2))
print(x)
print('\n')
print(y)
print('\n')
print('Stacked arrays: ')
print(np.concatenate((x,y),axis=0))
```

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


[[2 5]
 [7 6]]


Stacked arrays:
[[2 3]
 [5 3]
 [2 5]
 [7 6]]
```

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

In [25]:

```
x = np.random.randint(1,10,size=(2,2))
y = np.random.randint(1,10,size=(2,2))
print(x)
print('\n')
print(y)
print('\n')
print('Stacked arrays: ')
print(np.concatenate((x,y),axis=1))
```

```
[[8 4]
 [8 2]]


[[8 6]
 [7 9]]


Stacked arrays:
[[8 4 8 6]
 [8 2 7 9]]
```

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

In [28]:

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

```
[[ 7 45 27 37]
 [33 26 38 35]
 [37 47  7 45]
 [27 24 44 38]]
```

Out[28]:

```
array([[ 7, 37, 27, 45],
       [33, 35, 38, 26],
       [37, 45,  7, 47],
       [27, 38, 44, 24]])
```

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

In [29]:

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

```
[[26 24 49 47]
 [10 49 14 46]
 [49 17 28 29]
 [14 30 43 15]]
```

Out[29]:

```
array([[26, 24, 49, 47],
       [14, 30, 43, 15],
       [49, 17, 28, 29],
       [10, 49, 14, 46]])
```

In [ ]:

```

```