Basic Data Science

DataRitz Technologies
Enhancing Technology Experience

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

In [ ]:

```
import numpy as np
```

2. Create a null vector of size 20

In [14]:

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

```
[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 [13]:

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

```
[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 [12]:

```
arr = np.arange(12).reshape(3,4)
print(arr)
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
```

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

In [17]:

```
arr = np.arange(100.0,200.0)
print(arr)
```

```
[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 [19]:

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

Out[19]:

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

7. Reverse a given Vector

In [26]:

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

Out[26]:

```
array([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 [16]:

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

Out[16]:

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

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

In [42]:

```
arr = np.array([1,2,3,4,5,6,7,8,9,10,11,12,13,14])
for i,j in enumerate(arr):
    if j%2==0:
        arr[i]=-1
print(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 [44]:

```
r = np.random.randint(100,300,size=(5,3))
print(r)

print("minimum value:-",r.min())
print("maximum value:-",r.max())
```

```
[[192 274 134]
 [143 298 166]
 [116 212 141]
 [203 137 149]
 [152 202 202]]
minimum value:- 116
maximum value:- 298
```

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

In [47]:

```
arr = np.random.random(30)
print(arr)
print("mean value:-", arr.mean())
```

```
[0.57231261 0.01587331 0.60727832 0.48720545 0.94232576 0.71385135
 0.13056111 0.61220031 0.68065722 0.07169328 0.16966223 0.3928507
 0.31636854 0.33562264 0.76946796 0.49891821 0.14779878 0.20879302
 0.09268164 0.826622   0.40548308 0.40222476 0.18041484 0.89833839
 0.21964461 0.37024823 0.50394416 0.90931225 0.19829551 0.86291299]
mean value:- 0.45145210964108357
```

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 [48]:

```
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 [51]:

```
z= np.random.random((5,5))
print("orignal",z)
zmax, zmin = z.max(), z.min()
z = (z - zmin)/(zmax - zmin)
print("After normalization:")
print(z)
```

```
orignal [[0.49869509 0.2521881  0.36589688 0.11640945 0.09690363]
 [0.40188974 0.46307517 0.85597262 0.21766209 0.51053373]
 [0.71218036 0.23222956 0.52171169 0.92208543 0.21686272]
 [0.48331908 0.92100061 0.11800644 0.53711357 0.85576892]
 [0.27168543 0.66850574 0.52895268 0.99098602 0.71090502]]
After normalization:
[[0.44938975 0.17368027 0.30085958 0.02181658 0.        ]
 [0.34111634 0.40955011 0.84899222 0.13506414 0.46263086]
 [0.68816558 0.15135734 0.47513302 0.92293709 0.13417006]
 [0.43219222 0.92172376 0.02360277 0.49235948 0.84876438]
 [0.19548736 0.63931704 0.48323181 1.         0.68673916]]
```

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

In [52]:

```
x = np.random.random((5,3))
print("First array:")
print(x)
y = np.random.random((3,2))
print("Second array:")
print(y)
z = np.dot(x, y)
print("Dot product of two arrays:")
print(z)
```

```
First array:
[[0.98844209 0.39381809 0.92779015]
 [0.45700657 0.73159369 0.73069635]
 [0.45663775 0.67321131 0.77098793]
 [0.50103153 0.72366411 0.72228511]
 [0.77612522 0.49481481 0.87184395]]
Second array:
[[0.03630497 0.63160018]
 [0.90316423 0.05324613]
 [0.69507773 0.18084438]]
Dot product of two arrays:
[[1.03645404 0.81305512]
 [1.18523162 0.4597423 ]
 [1.16049513 0.46368722]
 [1.17382176 0.48560513]
 [1.08107555 0.67421588]]
```

15. How to find common values between two arrays?

In [55]:

```
arr1 =np.array([0, 10, 20, 40, 60])
arr2 =np.array([10, 30, 40])
print("Common values between two arrays:")
print(np.intersect1d(array1, array2))
```

```
Common values between two arrays:
[10 40]
```

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

In [9]:

```
import numpy as np
one = np.arange(2,22)
print(one.reshape(4,5))
```

```
[[ 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 [66]: x = np.array([[3], [5], [7]])
         y = np.array([[5], [7], [9]])
         varr=np.vstack((x,y))
         varr

Out[66]: array([[3],
                [5],
                [7],
                [5],
                [7],
                [9]])
```

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

```
In [67]: x = np.array([[3], [5], [7]])
         y = np.array([[5], [7], [9]])
         harr=np.hstack((x,y))
         harr

Out[67]: array([[3, 5],
                [5, 7],
                [7, 9]])
```

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

```
In [12]: arr = np.random.randint(1,16,size=(4,4))
         print(arr)
         arr[:,[1,3]]=arr[:,[3,1]]
         arr

         [[ 6  8  1 12]
          [ 8 14 13 14]
          [13 13  7  7]
          [ 1  8 14  8]]

Out[12]: array([[ 6, 12,  1,  8],
                [ 8, 14, 13, 14],
                [13,  7,  7, 13],
                [ 1,  8, 14,  8]])
```

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

```
In [13]: arr = np.random.randint(1,16,size=(4,4))
         print(arr)
         arr[[1,3],:]=arr[[3,1],:]
         arr
```

```
[[ 1  1  3  6]
 [10  9 10  8]
 [12 13  9 15]
 [ 9 13 11  4]]
```

```
Out[13]: array([[ 1,  1,  3,  6],
                [ 9, 13, 11,  4],
                [12, 13,  9, 15],
                [10,  9, 10,  8]])
```

```
In [ ]:
```