# The Basics of NumPy Arrays

In [2]:

```
import numpy as np
```

**1)Create a Numpy array filled with all zeros[1d and 2d]**

In [3]:

```
a = np.zeros(5)
print(a)
b = np.zeros((2, 5))
print(b)
```

```
[0. 0. 0. 0. 0.]
[[0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]]
```

**2)Create a Numpy array filled with all ones[1d and 2d]**

In [30]:

```
a = np.ones(5)
print(a, a.shape)
b = np.ones((2, 5))
print(b)
```

```
[1. 1. 1. 1. 1.] (5,)
[[1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]]
```

**3)Slice elements from index 4 to the end of the array:**

In [49]:

```
a = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 0])
a[4:]
# doubt from index 1 to index 5 a[1:5]
```

Out[49]:

```
array([5, 6, 7, 8, 9, 0])
```

**4)Slice from the index 3 from the end to index 1 from the end**

In [50]:

```
a[-3:]
# doubt from index 3 to index 1 a[-3:-1]
```

Out[50]:

```
array([8, 9, 0])
```

**5)write the necessary function to get below output**

input-

arr1 = np.array([[1, 2], [3, 4]]) arr2 = np.array([[10, 20], [30, 40]])

**output-[ 1 2 3 4 10 20 30 40]**

In [35]:

```
arr1 = np.array([[1, 2], [3, 4]])
arr2 = arr1 * 10
# cat1 = np.ndarray.flatten(arr1)
# cat2 = np.ndarray.flatten(arr2)
# cat1 = np.concatenate(arr1)
# cat2 = np.concatenate(arr2)
cat1 = arr1.reshape((-1,)) # or explicitely specify 4
cat2 = arr2.reshape((-1,))
output = np.concatenate([cat1, cat2])
output
```

Out[35]:

```
array([ 1,  2,  3,  4, 10, 20, 30, 40])
```

**6)arr1 = np.array([[1, 2], [3, 4]]) arr2 = np.array([[10, 20], [30, 40]])**

**merge above arrays along axis=0 and axis=1**

In [15]:

```
arr1 = np.array([[1, 2], [3, 4]])
arr2 = arr1 * 10
output = np.concatenate([arr1, arr2], axis=0)
print("axis0", output)
output = np.concatenate([arr1, arr2], axis=1)
print("axis1", output)
```

```
axis0 [[ 1  2]
 [ 3  4]
 [10 20]
 [30 40]]
axis1 [[ 1  2 10 20]
 [ 3  4 30 40]]
```

**7)Create a numpy array and find the Sum of All the Elements in the Array**

In [17]:

```
a = np.array([[1, 2], [3, 4]])
print(np.sum(a))
```

```
10
```

**8)Create a numpy array and find the Sum of Array Elements Along the Axis=0 & axis=1**

In [36]:

```
a = np.array([[1, 2], [3, 4]])
print("axis0", np.sum(a, axis=0))
print("axis1", np.sum(a, axis=1))
```

```
axis0 [4 6]
axis1 [3 7]
```

**9)Generate a linear sequence from 0.2 (included) until 2 (excluded) with a step size of 0.1, so there will be (2 – 0.2)/0.1 – 1 = 20 elements in the sequence, which is the length of the resulting numpy array.**

In [37]:

```
a = np.arange(0.2, 2, 0.1)
print(a, a.size) # doubt
```

```
[0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.  1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9] 18
```

**10)Convert the 1-D array with 12 elements into a 4*3 2-D array.**

In [40]:

```
a = np.arange(12)
b = a.reshape((4, 3))
print(a, b, sep='\n')
```

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

**11)Convert the 1-D array with 12 elements into a 2* 3 * 2 3-D array.**

In [41]:

```
a = np.arange(12)
b = a.reshape((2, 3, 2))
print(a, b, sep='\n')
```

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

 [[ 6  7]
  [ 8  9]
  [10 11]]]
```

**12)Assume the students are sitting in the form of 3x4 array. Write a program to create the array of students from the given array using slicing concept from Numpy**

**Create slicing element from below index position:**

1. 1nd Person in the 1st row
2. 3th Person in the 2nd row
3. 2rd Person in the 2st row

In [44]:

```
a = np.arange(12).reshape((3, 4))
print(a)
print(a[0, 0])
print(a[1, 2])
print(a[1, 1])
```

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

**13)Split the 2-D array into three 2-D arrays.**

In [51]:

```
a = np.arange(16).reshape((4, 4))
print(a)
# x1, x2, x3 = np.split(a, 3) will throw error because it doesn't
x1, x2, x3 = np.split(a, [1, 2])
print(x1, x2, x3, sep='\n')
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]
[[0 1 2 3]]
[[4 5 6 7]]
[[ 8  9 10 11]
 [12 13 14 15]]
```

**14)Split the 2-D array into three 2-D arrays along rows**

In [6]:

```
a = np.arange(16).reshape((4, 4))
print(a)
# x1, x2, x3 = np.split(a, 3) will throw error because it doesn't
# x1, x2, x3, x4 = np.split(a, 4, axis=0)
x1, x2, x3 = np.split(a, [1, 2], axis=0)
# x1, x2, x3, x4 = np.split(a, 4, axis=1)
print(x1, x2, x3, sep='\n')
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]
[[0 1 2 3]]
[[4 5 6 7]]
[[ 8  9 10 11]
 [12 13 14 15]]
```

In [5]:

```
a = np.arange(16).reshape((4, 4))
print(a)
x1, x2, x3, x4 = np.split(a, 4, axis=1)
print(x1, x2, x3, sep='\n')
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]
[[ 0]
 [ 4]
 [ 8]
 [12]]
[[ 1]
 [ 5]
 [ 9]
 [13]]
[[ 2]
 [ 6]
 [10]
 [14]]
```

# Aggregations: Min, Max, and Everything In Between

**1) Write the Python code to print the maximum of 4,12,43.3,19,100**

In [7]:

```
a = np.array([4, 12, 43.3, 19, 100])
a.max() # or np.max(a)
```

Out[7]:

```
100.0
```

**2) Write the python code to print the minimum of 4,12,43.3,19,100**

In [8]:

```
a.min() # or np.min(a)
```

Out[8]:

```
4.0
```

**3) Check whether your able to find the minimum from the given set of values :: 4,12,43.3,19, "HelloProgramming"**

In [21]:

```
a = np.array([4, 12, 43.3, 19, 100, "HelloProgramming"])
# a.dtype
# NOT POSSIBLE
a.min() # np.min(a) # min won't work for only strings also, but SORT works
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_28012/3403580322.py in <module>
      2 # a.dtype
      3 # NOT POSSIBLE
----> 4 a.min() # np.min(a)

c:\users\deven\appdata\local\programs\python\python39\lib\site-packages\nump
y\core\_methods.py in _amin(a, axis, out, keepdims, initial, where)
     41 def _amin(a, axis=None, out=None, keepdims=False,
     42           initial=_NoValue, where=True):
---> 43     return umr_minimum(a, axis, None, out, keepdims, initial, where)
     44
     45 def _sum(a, axis=None, dtype=None, out=None, keepdims=False,

TypeError: cannot perform reduce with flexible type
```

**4) Write the python code to print the word occurring 1st among these in dict::** "GoodMorning", "Evening", "algorithm", "programming"

In [40]:

```
# dict HERE THEY MEAN ACTUAL DICTIONARY AND NOT dict DATATYPE
# a = np.array(["GoodMorning", "Evening", "algorithm", "programmin"], dtype='S')
# a.min()
# min("GoodMorning", "Evening", "algorithm", "programming")
a = np.array(["GoodMorning", "Evening", "algorithm", "programming"], dtype='S')
a.sort()
a[0]
```

Out[40]:

```
b'Evening'
```

**5) Write the python code to print the min and max values from the given list of tuple:** [(2, 3), (4, 7), (8, 11), (3, 6)]

In [41]:

```
# max("GoodMorning", "Evening", "algorithm", "programming")
a[-1]
```

Out[41]:

```
b'programming'
```

# SORTING

***1)Create a list [[4,3,2],[2,1,4]], convert it to a numpy array and sort it along axis 1 ***

In [32]:

```
a = np.array([[4, 3, 2], [2, 1, 4]])
a.sort(axis=1) # or b = np.sort(a, axis=1) and ouput b
a
```

Out[32]:

```
array([[2, 3, 4],
       [1, 2, 4]])
```

**2)Implement a program to take fruits names from array of fruits. To sort the array in alphabetical manner and display their index position.**

In [35]:

```
fruits = ["banana", "apple", "jackfruit", "mango"]
a = np.array(fruits)
i = a.argsort()
print(i, a)
a.sort()
print(a)
```

```
[1 0 2 3] ['banana' 'apple' 'jackfruit' 'mango']
['apple' 'banana' 'jackfruit' 'mango']
```

**3) Write a NumPy program to partition a given array in a specified position and move all the smaller elements values to the left of the partition, and the remaining values to the right, in arbitrary order (based on random choice).**

In [37]:

```
a = np.random.randint(10, size=10)
a, np.partition(a, 2), np.partition(a, 4)
```

Out[37]:

```
(array([2, 0, 7, 6, 4, 8, 5, 9, 7, 9]),
 array([0, 2, 4, 6, 7, 8, 5, 9, 7, 9]),
 array([0, 2, 4, 5, 6, 8, 7, 9, 7, 9]))
```

In [ ]:

```

```