Numpy For Data Science

October 3, 2023

1 Numpy Tutorial by Asad Mujeeb

[4]:
```
# install the library
# pip install numpy
```

2 Chapter : 01 [— Basics—]

[3]:
```
# import the library

import numpy as np

# we use np as an allias so that we use use "as" instead of "numpy"
```

[7]:
```
# to check the version of module
print(np.__version__)
```

```
1.24.3
```

3 Create a simple Array

[8]:
```
arr = np.array([1,2,3,4,5])
```

[9]:
```
print(arr)
```

```
[1 2 3 4 5]
```

[10]:
```
print(type(arr))
```

```
<class 'numpy.ndarray'>
```

4 use a tuple to create an Array

[11]:
```
arr = np.array((3,4,5,6))
```

[12]:
```
print(arr)
```

```
[3 4 5 6]
```

[13]:
```
type(arr)
```

[13]:
```
numpy.ndarray
```

5 0D Array

[14]:
```
a = np.array(43)
```

[15]:
```
print(a)
```

```
43
```

6 1D Array

[17]:
```
b = np.array([3,4,5,6])
```

[18]:
```
print(b)
```

```
[3 4 5 6]
```

7 2D Array

[20]:
```
c = np.array([[1,2,3,4], [5,6,7,8]])
```

[21]:
```
print(c)
```

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

8 3D Array

[22]:
```
d = np.array([[[1,2,3],[4,5,6],[7,8,9]]])
```

[23]:
```
print(d)
```

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

9 Check the Diamension of an Array

[24]:
```
print(a.ndim)
```

```
0
```

[25]:
```
print(b.ndim)
```

```
1
```

[26]:
```
print(c.ndim)
```

```
2
```

[27]:
```
print(d.ndim)
```

```
3
```

10 Higher Dimensional Arrays

[28]:
```
# to create a higher diamensional array , we use " ndmin "
```

[29]:
```
arr = np.array([1,3,5,7], ndmin = 5)
```

[32]:
```
print(arr)
print("\n")
print("The number of diamension are :", arr.ndim)
```

```
[[[[[1 3 5 7]]]]]


The number of diamension are : 5
```

11 Chapter : 02 [Access the Array Element]

[ ]:
```
# access the first element
```

[4]:
```
#             0 1 2 3
a = np.array([3,4,5,6])
```

[5]:
```
print(a[0])
```

```
3
```

[6]:
```
# to access the 2nd element
```

[7]:
```
print(a[1])
```

```
4
```

[8]:
```
# to access the 3rd element
```

[9]:
```
a[2]
```

[9]: 5

[12]:
```
# Get second and third elements from the following array and add them.
a[2] + a[3]
```

[12]:
```
11
```

[13]:
```
# to access the 2D Array
```

[14]:
```
arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])

print('2nd element on 1st row: ', arr[0, 1])
```

```
2nd element on 1st row:  2
```

[17]:
```
arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])

print('5th element on 2nd row: ', arr[1, 4])
```

```
5th element on 2nd row:  10
```

[ ]:
```
# Access 3D Array
```

[18]:
```
arr = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])

print(arr[0, 1, 2])
```

```
6
```

[19]:
```
# Use negative indexing to access an array from the end.
```

[20]:
```
arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])

print('Last element from 2nd dim: ', arr[1, -1])
```

```
Last element from 2nd dim:  10
```

[ ]:
```

```

12 Chapter: 03 [Array Sciling]

[21]:
```
# [ start : end ]
```

[22]:
```
# [start : end : step]
```

[24]:
```
#               0  1  2  3  4  5  6
arr = np.array([1, 2, 3, 4, 5, 6, 7])

print(arr[1:5])
```

```
[2 3 4 5]
```

[25]:
```
# Slice elements from index 4 to the end of the array:

arr = np.array([1, 2, 3, 4, 5, 6, 7])

print(arr[4:])
```

```
[5 6 7]
```

[26]:
```
# Slice elements from the beginning to index 4 (not included):

arr = np.array([1, 2, 3, 4, 5, 6, 7])

print(arr[:4])
```

```
[1 2 3 4]
```

[28]:
```
# Use the minus operator to refer to an index from the end:(last index did not␣
↪include)

arr = np.array([1, 2, 3, 4, 5, 6, 7])

print(arr[-3:-1])
```

```
[5 6]
```

[29]:
```
# Use the step value to determine the step of the slicing (start : end : step)

arr = np.array([1, 2, 3, 4, 5, 6, 7])

print(arr[1:5:2])
```

```
[2 4]
```

[31]:
```
# Return every other element from the entire array:

arr = np.array([1, 2, 3, 4, 5, 6, 7])

print(arr[::3])
```

```
[1 4 7]
```

13 Slicing 2-D Arrays

[32]:
```
arr = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])

print(arr[1, 1:4])
```

```
[7 8 9]
```

[33]:
```
# From both elements, return index 2:

arr = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])

print(arr[0:2, 2])
```

```
[3 8]
```

[34]:
```
# From both elements, slice index 1 to index 4 (not included), this will return␣
 ↪a 2-D array:

arr = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])

print(arr[0:2, 1:4])
```

```
[[2 3 4]
 [7 8 9]]
```

14 Chapter : 04 [Data Types]

[35]:
```
arr = np.array([1, 2, 3, 4])

print(arr.dtype)
```

```
int32
```

[36]:
```
# Get the data type of an array containing strings:

arr = np.array(['apple', 'banana', 'cherry'])

print(arr.dtype)
```

```
<U6
```

[41]:
```
# Create an array with data type string:

arr = np.array([9,9,8,7,6], dtype = "S")

print(arr.dtype)
```

```
|S1
```

[42]:
```
#  A non integer string like 'a' can not be converted to integer (will raise an␣
 ↪error):


arr = np.array(['a', '2', '3'], dtype='i')

print(arr)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[42], line 4
      1 #  A non integer string like 'a' can not be converted to integer (will␣
 ↪raise an error):
----> 4 arr = np.array(['a', '2', '3'], dtype='i')
      6 print(arr)

ValueError: invalid literal for int() with base 10: 'a'
```

[45]:
```
# Change data type from float to integer by using 'i' as parameter value:

arr = np.array([1.2, 4.5, 7.8])

new_arr = arr.astype("i")

print(new_arr)

print(new_arr.dtype)
```

```
[1 4 7]
int32
```

[46]:
```
# Change data type from integer to boolean:

arr = np.array([1, 0, 3])

newarr = arr.astype(bool)

print(newarr)
print(newarr.dtype)
```

```
[ True False  True]
bool
```

[ ]:

15 continue ...!!!