Day 12

Numpy Library

In [1]:
```
import numpy as np
```

In [2]:
```
print(np.__version__)
```

```
1.21.5
```

Create a NumPy ndarray Object

NumPy is used to work with arrays. The array object in NumPy is called ndarray.

We can create a NumPy ndarray object by using the array() function.

In [4]:
```
import numpy as np

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

print(arr)

print(type(ar))
```

```
[1 2 3 4 5]
<class 'numpy.ndarray'>
```

To create an ndarray, we can pass a list, tuple or any array-like object into the array() method, and it will be converted into an ndarray:

Use a tuple to create a NumPy array:

In [6]:
```
import numpy as np

arr = np.array((1, 2, 3, 4, 5))

print(arr)
print(type(arr))
```

```
[1 2 3 4 5]
<class 'numpy.ndarray'>
```

Dimensions in Arrays

0-D Arrays 0-D arrays, or Scalars, are the elements in an array. Each value in an array is a 0-D array.

Create a 0-D array with value 20

In [8]:
```
arr = np.array(42)

print(arr)
```

```
42
```

1-D Arrays

1-D Arrays An array that has 0-D arrays as its elements is called uni-dimensional or 1-D array.

In [10]:
```
arr = np.array([1, 2, 3, 4, 5])

print(arr)
print ( type (arr))
print(arr.ndim)
```

```
[1 2 3 4 5]
<class 'numpy.ndarray'>
1
```

Creating 2 D array

In [11]:
```

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

print(arr)
```

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

In [12]:
```
arr = np.array([[1, 2, 3], [4, 5, 6]])

print(arr)
print ( type (arr))
print(arr.ndim)
```

```
[[1 2 3]
 [4 5 6]]
<class 'numpy.ndarray'>
2
```

In [13]:
```
arr = np.array([[1.1, 2.2, 3.3], [4.4, 5.5, 6.6]])

print(arr)
print ( type (arr))
print(arr.ndim)
```

```
[[1.1 2.2 3.3]
 [4.4 5.5 6.6]]
<class 'numpy.ndarray'>
2
```

3-D arrays

An array that has 2-D arrays (matrices) as its elements is called 3-D array.

In [14]:
```
import numpy as np

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

print(arr)
```

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

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

In [15]:
```
import numpy as np

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

print(arr)
print(type(arr))
print(arr.ndim)
```

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

 [[1 2 3]
  [4 5 6]]]
<class 'numpy.ndarray'>
3
```

Check Number of Dimensions?

In [16]:
```
a = np.array(42)
b = np.array([1, 2, 3, 4, 5])
c = np.array([[1, 2, 3], [4, 5, 6]])
d = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])

print(a.ndim)
print(b.ndim)
print(c.ndim)
print(d.ndim)
```

```
0
1
2
3
```

Higher Dimensional Arrays

An array can have any number of dimensions.

When the array is created, you can define the number of dimensions by using the ndmin argument.

Create an array with 5 dimensions and verify that it has 5 dimensions:

In [20]:
```
arr = np.array([1, 2, 3, 4], ndmin=5)

print(arr)
print('number of dimensions :', arr.ndim)
```

```
[[[[[1 2 3 4]]]]]
number of dimensions : 5
```

In [21]:
```
a = np.array([[[[[1,2,3,4]]]]])
print(a)
print(a.ndim)
```

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

NumPy Array Indexing

Array indexing is the same as accessing an array element.

You can access an array element by referring to its index number.

The indexes in NumPy arrays start with 0, meaning that the first element has index 0, and the second has index 1 etc.

Get the first element from the following array:

In [22]:
```


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

print(arr[0])
```

```
1
```

get third and sixth and 8th elements from the following array and add them.

In [25]:
```
m = np.array([41,87,34,56.8, 39.23, 90,15, 10,19,56])
print("the second element is", m[2])
print("the sixth element is", m[5])
print("the 8th element is",m[7])
print("sum of this elenets are", m[2]+m[5]+m[7])
```

```
the second element is 34.0
the sixth element is 90.0
the 8th element is 10.0
sum of this elenets are 134.0
```

Access 2-D Arrays

To access elements from 2-D arrays we can use comma separated integers representing the dimension and the index of the element.

Think of 2-D arrays like a table with rows and columns, where the dimension represents the row and the index represents the column.

In [27]:
```
arr = np.array([[10,20,30,40,50], [60,70,80,90,100]])

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

```
2nd element on 1st row:  20
```

In [28]:
```
arr = np.array([[10,20,30,40,50], [60,70,80,90,100]])

print('4th element on 1st row: ', arr[0, 3])
```

```
4th element on 1st row:  40
```

In [29]:
```
arr = np.array([[10,20,30,40,50], [60,70,80,90,100]])

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

```
1st element on 1st row:  60
```

In [30]:
```
arr = np.array([[10,20,30,40,50], [60,70,80,90,100]])

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

```
last element on 2nd row:  100
```

add the first element of row 1 and last element

In [33]:
```
arr = np.array([[10,20,30,40,50], [60,70,80,90,100]])

print('First element of 1st row: ', arr[0, 0])
print('last element of 2nd row: ', arr[1, 4])
print( "sum is" , arr[0,0]+arr[1,4])
```

```
First element of 1st row:  10
last element of 2nd row:  100
sum is 110
```

Access 3-D Arrays

To access elements from 3-D arrays we can use comma separated integers representing the dimensions and the index of the element.

Access the third element of the second array of the first array:

In [36]:
```
arr = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])

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

```
6
```

In [37]:
```
arr = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])

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

```
12
```

In [38]:
```
arr = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])

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

```
8
```

In [39]:
```
arr = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])

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

```
3
```

Negative Indexing

Use negative indexing to access an array from the end.

In [40]:
```
import numpy as np

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
```

NumPy Array Slicing

Slicing in python means taking elements from one given index to another given index.

We pass slice instead of index like this: [start:end].

We can also define the step, like this: [start:end:step].

Slice elements from index 1 to index 5 from the following array:

In [41]:
```
import numpy as np

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

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

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

Slice elements from index 4 to the end of the array

In [42]:
```
arr = np.array([1, 2, 3, 4, 5, 6, 7])

print(arr[4:])
```

```
[5 6 7]
```

In [43]:
```
arr = np.array([1, 2, 3, 4, 5, 6, 7])

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

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

Slicing 2 D array

In [44]:
```
a = np.array([[10,20,30,40,50],[60,70,80,90,100]])
print(a[1,1:4])
```

```
[70 80 90]
```

In [45]:
```
a = np.array([[10,20,30,40,50],[60,70,80,90,100]])
print(a[0,0:3])
```

```
[10 20 30]
```

From both elements, return index 2

In [49]:
```
arr = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])

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

```
[3 8]
```

slice index 1 to index 4 (not included), this will return a 2-D array

In [50]:
```
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]]
```

In [ ]: