## 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.

**#Number Creation array**

print("Numpy Array Creation using 1D tuple")

import numpy as np

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

arr[1]=20

print(arr)

print("Numpy Array Creation using 2D tuple")

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

print(arr)

print("Numpy Array Creation using 1D list")

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

arr[1]=20

print(arr)

print("Numpy Array Creation using 2D list")

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

print(arr)

**Output**

Numpy Array Creation using 1D tuple

[ 1 20 3]

Numpy Array Creation using 2D tuple

[[1 2 3]

[4 5 6]]

Numpy Array Creation using 1D list

[ 1 20 3]

Numpy Array Creation using 2D list

[[1 2 3]

[4 5 6]]

# NumPy Array Slicing

## Slicing arrays

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].

If we don't pass start its considered 0

If we don't pass end its considered length of array in that dimension

If we don't pass step its considered 1

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

import numpy as np  
  
arr = np.array([1, 2, 3, 4, 5, 6, 7])  
  
print(arr[1:5])

**Note:** The result includes the start index, but excludes the end index.

Slice elements from index 4 to the end of the array:

import numpy as np  
  
arr = np.array([1, 2, 3, 4, 5, 6, 7])  
  
print(arr[4:])

Slice elements from the beginning to index 4 (not included):

import numpy as np  
  
arr = np.array([1, 2, 3, 4, 5, 6, 7])  
  
print(arr[:4])

## Negative Slicing

Slice from the index 3 from the end to index 1 from the end:

import numpy as np  
  
arr = np.array([1, 2, 3, 4, 5, 6, 7])  
  
print(arr[-3:-1])

## STEP

Use the step value to determine the step of the slicing:

### Example

Return every other element from index 1 to index 5:

import numpy as np  
  
arr = np.array([1, 2, 3, 4, 5, 6, 7])  
  
print(arr[1:5:2])

## Slicing 2-D Arrays

### Example

From the second element, slice elements from index 1 to index 4 (not included):

**import numpy as np**

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

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

From both rows, return index 2:

**import numpy as np**

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

**[6, 7, 8, 9, 10]])**

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

**#output**

**[3 8]**

import numpy as np

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

[6, 7, 8, 9, 10]])

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

**output:**

[[2 3 4]

[7 8 9]]

## Dimensions in Arrays

A dimension in arrays is one level of array depth (nested arrays).

**nested array:** are arrays that have arrays as their elements.

**NumPy Array Copy vs View**

## he Difference Between Copy and View

The main difference between a copy and a view of an array is that the copy is a new array, and the view is just a view of the original array.

The copy owns the data and any changes made to the copy will not affect original array, and any changes made to the original array will not affect the copy.

The view does not own the data and any changes made to the view will affect the original array, and any changes made to the original array will affect the view.

The copy SHOULD NOT be affected by the changes made to the original array.

The view SHOULD be affected by the changes made to the original array.

print("copy")

import numpy as np

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

x = arr.copy()

arr[0] = 42

print("\noriginal data",arr)

print("\ncopied data",x)

print("\nData after change arr[0]=42\n\n")

print("original data",arr)

print("copied data",x)

print("view")

import numpy as np

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

x = arr.view()

arr[0] = 42

print("\noriginal data",arr)

print("\ncopied data",x)

print("\nData after change arr[0]=42\n\n")

print("original data",arr)

print("copied data",x)

**output:**

copy

original data [42 2 3 4 5]

copied data [1 2 3 4 5]

Data after change arr[0]=42

original data [42 2 3 4 5]

copied data [1 2 3 4 5]

view

original data [42 2 3 4 5]

copied data [42 2 3 4 5]

Data after change arr[0]=42

original data [42 2 3 4 5]

copied data [42 2 3 4 5]