## Flattening the arrays

Flattening array means converting a multidimensional array into a 1D array.
We can use reshape(-1) to do this.

Convert the array into a 1D array:

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

O/P

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

## Numpy Array Iterating

Iterating means going through elements one by one.
As we deal with multi-dimensional arrays in numpy, we can do this using basic for loop of python.

Iterate on the elements of the following 1-D array:

```
import numpy as np
arr = np.array([1,2,3])
for x in arr:
    print(x)
```

O/P

```
1
2
3
```

## Iterating 2-D Arrays

Iterating on the elements of the following 2-D array:

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

Output

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

2) Iterate on each scalar element of the 2-D array:

```
import numpy as np
arr = np.array([[1,2,3],[4,5,6]])
for x in arr:
        for y in x
            print(y)
```

Output

```
1
2
3
4
5
6
```

## Iterating 3D Arrays

In a 3-D array it will go through all the 2-D arrays.

Iterate on the elements of the following 3-D array:

1.)
```
import numpy as np
arr = np.array([[[1,2,3],[4,5,6]],[[7,8,9],
                        [10,11,12]]])
for x in arr:
    print(x)
```

Output

x represents the 2-D array:
```
[[1 2 3]
 [4 5 6]]
```
x represents the 2-D array:
```
[[7 8 9]
 [10 11 12]]
```
To return the actual values, the scalars, we have to iterate the arrays in each dimension.

2) Iterate down to the scalars:
```
import numpy as np
arr = np.array([[[1,2,3],[4,5,6]],[[7,8,9],
                                  [10,11,12]]])

for x in arr:
    for y in x:
        for z in y:
            print(z)
```
Output
```
1,2 [1 2 3 4 5 6 7 8 9 10 11 12] (horizontally)
```
Iterating Arrays Using nditer()

The function nditer() is a helping function that can be used from very basic to very advanced iterations. It solves some basic issues which we face in iteration, let's go through it with examples.

Iterating on Each Scalar Element
```
import numpy as np
arr = np.array([[[1,2],[3,4]],[[5,6],[7,8]]])
for x in np.nditer(arr):
    print(x)
```
Output
```
[1 2 3 4 5 6 7 8]
```

## Iterating Array With Different Data Type

We can use op-dtypes argument and pass it the expected datatype to change the datatype of elements while iterating.

Numpy does not change the data type of the element in-place (where the element is in array) so it needs some other space to perform this action, that extra space is called buffer, and in order to enable it in nditer() we pass flags = ['buffered'].

e.g Iterate through the array as a string:

```
import numpy as np
arr = np.array([1, 2, 3])
for x in np.nditer(arr, flags=['buffered'],
                   op-dtypes=['S']):
    print(x)
```

Output

```
b'1'
b'2'
b'3'
```

## Iterating With Different Step Size

Iterate through every scalar element of the 2D array skipping 1 element:

e.g: ~~Iterate it~~

```
import numpy as np
arr = np.array([[1,2,3,4],[5,6,7,8]])
for x in np.nditer(arr[:, ::2]):
    print(x)
```

o/p

```
1
3
```