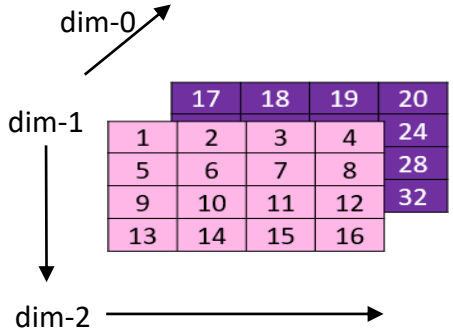
Objective: Python supports broadcasting on arrays. This worksheet explains the concept taking few examples. Broadcasting is a useful feature that is used for vectorization.



3-dimensional Array

- The above figure shows a 3-dimensional array of size 2x4x4 and it is named as ***arr3d*** in this worksheet. This array can be added to another array of the same size. But Python supports addition of arrays which are of different sizes. There are few rules of it:
- The dimensions are compared backwards. So if two arrays are of dimensions (m1 x m2 x m3) and (n1 x n2 x n3). First m3 and n3 are compared, then m2 and n2 and in the last m1 and n1 are compared.
- When a dimension in an array is missing, it scaled to that specific dimension of other the array. So if two arrays are of dimensions (m1 x m2 x m3) and (m2 x m3), the second arrays will be assumed to have m1 as the first dimension. So in this case (m2 x m3) will be replicated m1 times.
- Based on this rule, we have an arr3d of dimension (2 x 4 x 4) as shown above, so can it be added to an array of dimension (4 x 4). If yes, what the answer would be? Let us attempt it in the code.

```
arr3d = np.array([
    [
        [1, 2, 3, 4],
        [5, 6, 7, 8],
        [9, 10, 11, 12],
        [13, 14, 15, 16]
    ],
    [
        [17, 18, 19, 20],
        [21, 22, 23, 24],
        [25, 26, 27, 28],
        [29, 30, 31, 32]
    ]
])
```

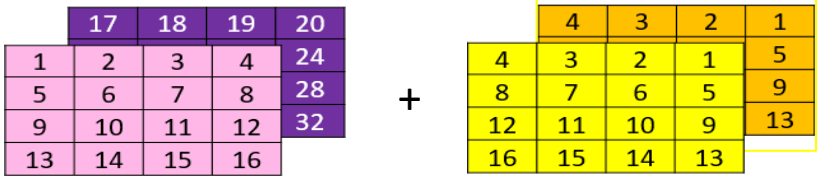
```
arr2d = np.array(
    [
        [4, 3, 2, 1],
        [8, 7, 6, 5],
        [12, 11, 10, 9],
        [16, 15, 14, 13]
    ]
)
```

```
arr3d+arr2d
```

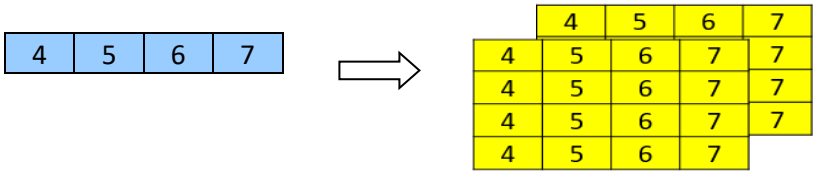
```
array([[[ 5,  5,  5,  5],
        [13, 13, 13, 13],
        [21, 21, 21, 21],
        [29, 29, 29, 29]],

       [[21, 21, 21, 21],
        [29, 29, 29, 29],
        [37, 37, 37, 37],
        [45, 45, 45, 45]]])
```

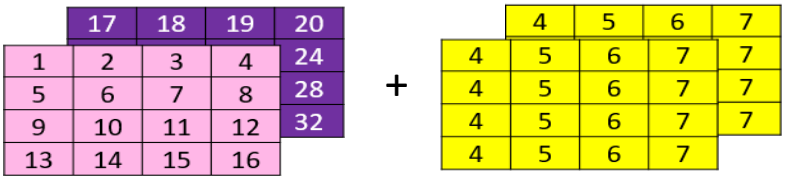
- Notice that the second array arr2d is of dimension 4 x 4, so it is assumed to be in the dimension of 2 x 4 x 4 borrowing the first dimension from arr3d and then the addition is performed.
- Yellow matrix replicated to match the first dimension of arr3d.



- When a dimension in an array is 1, it is also scaled to that specific dimension of the other array. So if two arrays are of dimensions (m1 x m2 x m3) and (1 x m3), the second arrays will be assumed to have m1 as the first dimension and m2 as the second dimension.
- Let us understand it from the code when an array arr1d of dimension (1 x 4) is added to arr3d. The arr1d is transformed to the dimension of 2 x 4 x 4 as following before it is added:



- And then the addition takes place as shown below:




```
arr1d = np.array([4, 5, 6, 7])
```

```
arr3d+arr1d
```

```
array([[[ 5,  7,  9, 11],
        [ 9, 11, 13, 15],
        [13, 15, 17, 19],
        [17, 19, 21, 23]],

       [[21, 23, 25, 27],
        [25, 27, 29, 31],
        [29, 31, 33, 35],
        [33, 35, 37, 39]]])
```

- If the dimension is not missing and it is not 1, then it means it is different. In that case, broadcast will not happen. For example, arr3d (2 x 4 x 4) cannot be added to an array of dimension (2 x 3 x 4) or an array of dimension of (2 x 2).
- ***reshape ()*** is a useful function which can help to reshape the arrays in NumPy. For example an array of 6 elements can be reshaped to an array of 2x3 as shown below:

```
arr1 = np.array([1, 2, 3, 4, 5, 6])
arr2 = arr1.reshape((2, 3))
arr2
```

```
array([[1, 2, 3],
       [4, 5, 6]])
```

- Functions ***reshape ()*** and ***arange ()*** can be used together to quickly create and shape the arrays. Let us say there is a need to create an array having element values from 0 to 23 arranged in a (2 x 3 x 4) dimension. It can be done as following:

```
newArray = np.arange(24).reshape((2, 3, 4))
newArray
```

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

       [[12, 13, 14, 15],
        [16, 17, 18, 19],
        [20, 21, 22, 23]]])
```

- An array along with T attribute can be used to get the transpose of the array.

```
myArr
```

```
array([[0, 1, 2],
       [3, 4, 5]])
```

```
myArr.T
```

```
array([[0, 3],
       [1, 4],
       [2, 5]])
```

Exercise:

1. Can arr3d be added to a scalar (just one single number)?
2. Can arr3d be added to an array of size 1 x 1 array?
3. The following lines of code is executed. Can newArray1 and newArray2 be added?

```
newArray1 = np.arange (24).reshape ((2, 3, 4))
newArray2 = np.ones ((1, 4))
```

4. Can these two arrays be added? If the first array (yellow) transposed? Try with the code also.

1	2	3	4

1	2	3	4	5

5. Which of the following three arrays can be added to arr3d? Also verify writing the code. In case of error, what is the error displayed?

2
3
5

2	3	5

2
3
5
7