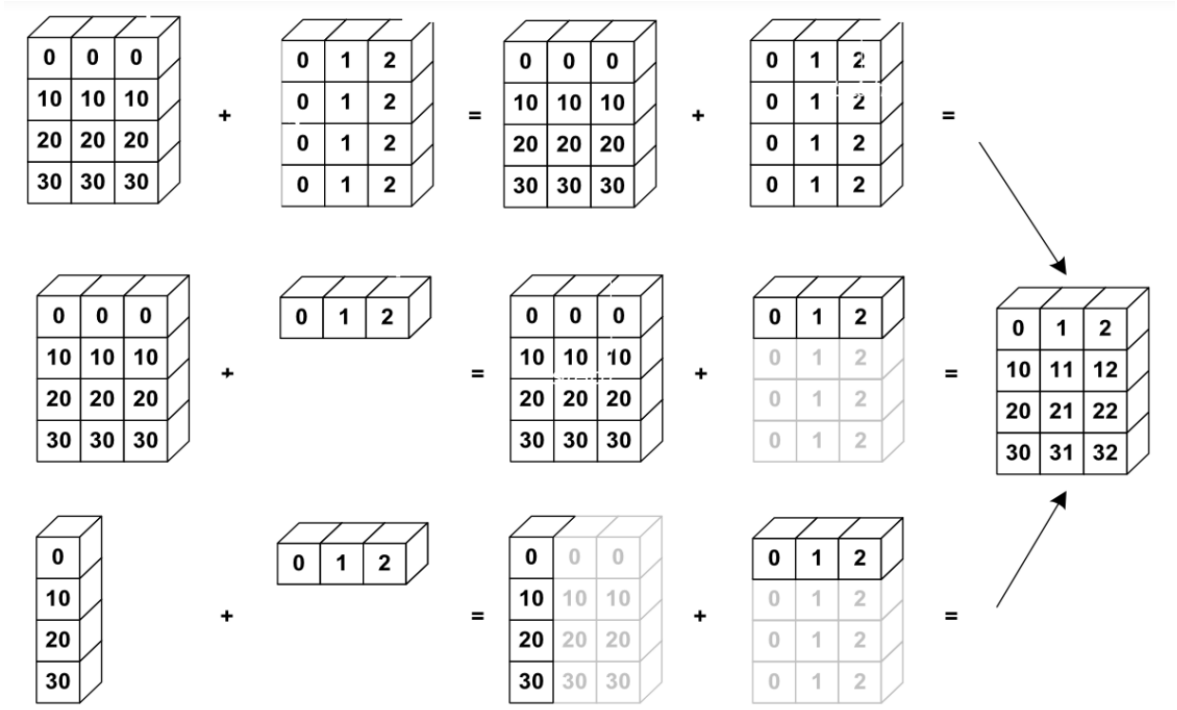
Broadcasting



Case1:

You are given two 2D array

```
[[0,   0,  0],      [[0, 1, 2],
 [10, 10, 10], and   [0, 1, 2],
 [20, 20, 20],       [0, 1, 2],
 [30, 30, 30]]       [0, 1, 2]]
```

Shape of **first array** is **4x3**

Shape of **second array** is **4x3**.

Will addtion of these array be possible? Yes as the shape of these two array matches.

In [ ]:
```
a = np.tile(np.arange(0,40,10), (3,1))
a
```

Out[5]:
```
array([[ 0, 10, 20, 30],
       [ 0, 10, 20, 30],
       [ 0, 10, 20, 30]])
```

`np.tile` function is used to repeat the given array multiple times

In [ ]:
```
1 np.tile(np.arange(0,40,10), (3,2))
```

Out[6]:
```
array([[ 0, 10, 20, 30,  0, 10, 20, 30],
       [ 0, 10, 20, 30,  0, 10, 20, 30],
       [ 0, 10, 20, 30,  0, 10, 20, 30]])
```

Now, let's get back to example:

In [ ]:
```
1 a
```

Out[7]:
```
array([[ 0, 10, 20, 30],
       [ 0, 10, 20, 30],
       [ 0, 10, 20, 30]])
```

In [ ]:
```
1 a = a.T
```

In [ ]:
```
1 a
```

Out[13]:
```
array([[ 0,  0,  0],
       [10, 10, 10],
       [20, 20, 20],
       [30, 30, 30]])
```

In [ ]:
```
1 b = np.tile(np.arange(0,3), (4,1))
```

In [ ]:
```
1 b
```

Out[15]:
```
array([[0, 1, 2],
       [0, 1, 2],
       [0, 1, 2],
       [0, 1, 2]])
```

Let's add these two arrays:

In [ ]:
```
1 a + b
```

Out[16]:
```
array([[ 0,  1,  2],
       [10, 11, 12],
       [20, 21, 22],
       [30, 31, 32]])
```

Text book case of element wise addition of two 2D arrays.

Case2 :

Imagine a array like this:

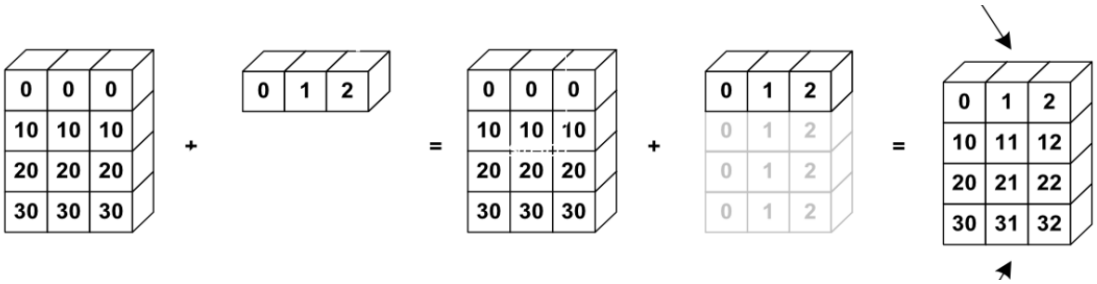
```
[[0,  0,  0],
 [10, 10, 10],
 [20, 20, 20],
 [30, 30, 30]]
```

I want to add the following array to it:

```
[[0, 1, 2]]
```

Is it possible? **Yes!**

What broadcasting does is replicate the second array row wise 4 times to fit the size of first array.



In [ ]:
```
a
```

Out[17]:
```
array([[ 0,  0,  0],
       [10, 10, 10],
       [20, 20, 20],
       [30, 30, 30]])
```

In [ ]:
```
b = np.arange(0,3)
b
```

Out[19]:
```
array([0, 1, 2])
```

In [ ]:
```
a + b
```

Out[20]:
```
array([[ 0,  1,  2],
       [10, 11, 12],
       [20, 21, 22],
       [30, 31, 32]])
```

The **smaller array is broadcast across the larger array** so that they have **compatible shapes**.

Case 3:

Imagine I have two array like this:

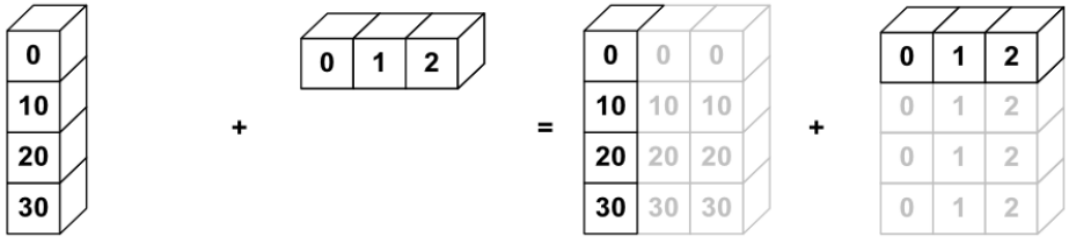
```
[[0],
[10],
[20],
[30]]
```

and

```
[[0, 1, 2]]
```

i.e. one column matrix and one row matrix.

When we try to add these array up, broadcasting will replicate first array column wise 3 time



In [ ]:
```
a = np.arange(0,40,10)
a
```

Out[22]:
```
array([ 0, 10, 20, 30])
```

This is a 1D row wise array, But we want this array colum wise? How do we do it ? Reshape?

In [ ]:
```
a = a.reshape(4,1)
a
```

Out[24]:
```
array([[ 0],
       [10],
       [20],
       [30]])
```

In [ ]:
```
b = np.arange(0,3)
b
```

Out[31]:
```
array([0, 1, 2])
```

In [ ]:
```
a + b
```

Out[32]:
```
array([[ 0,  1,  2],
       [10, 11, 12],
       [20, 21, 22],
       [30, 31, 32]])
```

Question: (for general broadcasting rules)

What will be the output of the following?

```
a = np.arange(8).reshape(2,4)
b = np.arange(16).reshape(4,4)

print(a*b)
```

In [ ]:
```
a = np.arange(8).reshape(2,4)
a
```

Out[3]:
```
array([[0, 1, 2, 3],
       [4, 5, 6, 7]])
```

In [ ]:
```
b = np.arange(16).reshape(4,4)
b
```

Out[4]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15]])
```

In [ ]:
```
a + b
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-6-bd58363a63fc> in <module>()
----> 1 a + b

ValueError: operands could not be broadcast together with shapes (2,4) (4,4)
```

Why didn't it work?

To understand this, let's learn about some **General Broadcasting Rules**

```
For each dimension ( going from right side)
    1. The size of each dimension should be same OR
    2. The size of one dimension should be 1
```

Rule 1 : If two array differ in the number of dimensions, the shape of one with fewer dimensions is padded with ones on its leading(Left Side).

Rule 2 : If the shape of two arrays doesnt match in any dimensions, the array with shape equal to 1 is stretched to match the other shape.

Rule 3 : If in any dimesion the sizes disagree and neither equal to 1 , then Error is raised.

In the above example, the shapes were (2,4) and (4,4).

Let's compare the dimension from right to left

- First, it will compare the right most dimension (4) which are equal.
- Next, it will compare the left dimension i.e. 2 and 4.
  - Both conditions fail here. They are neither equal nor one of them is 1.

Hence, it threw an error while broadcasting.

Now, Let's take a look at few more examples

Question : Will broadcasting work in this case ?

```
A = np.arange(1,10).reshape(3,3)
B = np.array([-1, 0, 1])
A * B
```

In [ ]:
```
A = np.arange(1,10).reshape(3,3)
A
```

Out[4]:
```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

In [ ]:
```
B = np.array([-1, 0, 1])
B
```

Out[5]:
```
array([-1,  0,  1])
```

In [ ]:
```
A * B
```

Out[6]:
```
array([[-1,  0,  3],
       [-4,  0,  6],
       [-7,  0,  9]])
```

Why did `A * B` work in this case?

- `A` has 3 rows and 3 columns i.e. (3,3)
- `B` is a 1-D vector with 3 elements (3,)

Now, if you look at **rule 1**

```
Rule 1 : If two array differ in the number of dimensions,
the shape of one with fewer dimensions is padded with ones on its le
ading( Left Side).
```

What is the shape of A and B ?

- A has a shape of (3,3)
- B has a shape of (3,)

As per the rule 1,

- the shape of array with fewer dimensions will be prefixed with ones on its leading side.

Here, shape of B will be prefixed with 1

- So, it's shape will become (1,3)

Can we add a (3,3) and (1,3) array ?

We check the validity of broadcasting. i.e. if broadcasting is possible or not.

Checking the dimension from right to left.

- It will compare the right most dimension (3); which are equal
- Now, it compares the leading dimension.
  - The size of one dimension is 1.

Hence, broadcasting condition is satisfied

How will it broadcast?

As per rule 2:

```
Rule 2 :
If the shape of two arrays doesnt match in any dimensions,
the array with shape equal to 1 is stretched to match the other shap
e.
```

Here, array B (**1**,3) will replicate/stretch its row 3 times to match shape of B

So , **`B` gets broadcasted over `A` for each row of `A`**

Question: Will broadcasting work in following case ?

```
A = np.arange(1,10).reshape(3,3)
B = np.arange(3, 10, 3).reshape(3,1)
C = A + B
```

In [ ]:
```
A = np.arange(1,10).reshape(3,3)
A
```

Out[3]:
```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

In [ ]:
```
B = np.arange(3, 10, 3).reshape(3,1)
B
```

Out[4]:
```
array([[3],
       [6],
       [9]])
```

How did this `A + B` work?

- `A` has 3 rows and 3 columns i.e. shape (3,3)
- `B` has 3 rows and 1 column -i.e. shape (3,1)

Do we need to check rule 1 ?

Since, both arrays have same number of dimensions, we can ignore Rule 1.

Let's check whether broadcasting is possible or not

Now, for each dimension from right to left

- Right most dimension is 1.
- Leading dimension are matching (3)

So, conditions for broadcasting are met.

How will broadcasting happen?

As per rule 2, dimension with value 1 will be streched.

- A.shape => (3,3)
- B.shape => (3,1)

Hence, columns of B will be replicated/streched to match dimensions of A.

- So, **`B` gets broadcasted on every column of `A`**

In [ ]:
```
C = A + B
np.round(C, 1)
```

Out[5]:
```
array([[ 4,  5,  6],
       [10, 11, 12],
       [16, 17, 18]])
```