Simple Maths

```
for simple mathematical opearations shape of the array has to match.
```

addition

In [27]:
```
import numpy as np
import warnings
warnings.filterwarnings('ignore')
array = np.array([5,7,4,8,9,5,7,1])
array+2
```

Out[27]:
```
array([ 7,  9,  6, 10, 11,  7,  9,  3])
```

In [2]:
```
import numpy as np
array = np.array([5,7,4,8,9,5,7,1])
array2 = np.array([4,7,2,8,0,2,8,1])

add = array+array2
add
```

Out[2]:
```
array([ 9, 14,  6, 16,  9,  7, 15,  2])
```

In [3]:
```
array = np.array([5,7,4,8,9,5,7,1])
array2 = np.array([4,7,2,8,0,2,8])

add = array+array2
add
```

...

In [4]:
```
array = np.array([5,7,4,8,9,5,7,1])
array2 = np.array([4,7,2,8,0,2,8,1])
array3 = np.array([1,2,2,3,0,2,8,1])
add = array+array2+array3
add
```

Out[4]:
```
array([10, 16,  8, 19,  9,  9, 23,  3])
```

In [8]:
```
# np.add only performs on 2 arrays at a time
array = np.array([5,7,4,8,9,5,7,1])
array2 = np.array([4,7,2,8,0,2,8,1])
array3 = np.array([1,2,2,3,0,2,8,1])
np.add(array,array2,array3)     # array3 is not included in addition
```

Out[8]:
```
array([ 9, 14,  6, 16,  9,  7, 15,  2])
```

In [9]:
```
array = np.array([5,7,4,8,9,5,7,1])
array2 = np.array([4,7,2,8,0,2,8,1])
array3 = np.array([1,2,2,3,0,2,8,1])
np.add(array,array3)
```

Out[9]:
```
array([ 6,  9,  6, 11,  9,  7, 15,  2])
```

In [15]:
```
array = np.array([[5,7,4],
                  [8,9,5],
                  [7,1,8]])
array2 = np.array([[4,7,2],
                   [8,0,2],
                   [8,1,6]])

np.add(array,array2)
```

Out[15]:
```
array([[ 9, 14,  6],
       [16,  9,  7],
       [15,  2, 14]])
```

Subtraction

In [16]:
```
array = np.array([[5,7,4],
                  [8,9,5],
                  [7,1,8]])
array2 = np.array([[4,7,2],
                   [8,0,2],
                   [8,1,6]])

np.subtract(array,array2)
```

Out[16]:
```
array([[ 1,  0,  2],
       [ 0,  9,  3],
       [-1,  0,  2]])
```

In [17]:
```
array = np.array([[5,7,4],
                  [8,9,5],
                  [7,1,8]])
array-2
```

Out[17]:
```
array([[ 3,  5,  2],
       [ 6,  7,  3],
       [ 5, -1,  6]])
```

In [18]:
```
array = np.array([[5,7,4],
                  [8,9,5],
                  [7,1,8]])
array2 = np.array([[4,7,2],
                   [8,0,2],
                   [8,1,6]])

array-array2
```

Out[18]:
```
array([[ 1,  0,  2],
       [ 0,  9,  3],
       [-1,  0,  2]])
```

multiplication

In [19]:
```
array = np.array([[5,7,4],
                  [8,9,5],
                  [7,1,8]])
array2 = np.array([[4,7,2],
                   [8,0,2],
                   [8,1,6]])

np.multiply(array,array2)
```

Out[19]:
```
array([[20, 49,  8],
       [64,  0, 10],
       [56,  1, 48]])
```

In [20]:
```
array = np.array([[5,7,4],
                  [8,9,5],
                  [7,1,8]])
array*10
```

Out[20]:
```
array([[50, 70, 40],
       [80, 90, 50],
       [70, 10, 80]])
```

division

In [31]:
```
array = np.array([[5,7,4],
                  [8,9,5],
                  [7,1,8]])
array2 = np.array([[4,7,2],
                   [8,0,2],
                   [8,1,6]])

np.divide(array,array2)
```

Out[31]:
```
array([[1.25      , 1.        , 2.        ],
       [1.        ,        inf, 2.5       ],
       [0.875     , 1.        , 1.33333333]])
```

In [30]:
```
array/array2
```

Out[30]:
```
array([[1.25      , 1.        , 2.        ],
       [1.        ,        inf, 2.5       ],
       [0.875     , 1.        , 1.33333333]])
```

In [23]:
```
array/2
```

Out[23]:
```
array([[2.5, 3.5, 2. ],
       [4. , 4.5, 2.5],
       [3.5, 0.5, 4. ]])
```

floor_divide

In [29]:
```
array = np.array([[5,7,4],
                  [8,9,5],
                  [7,1,8]])
array2 = np.array([[4,7,2],
                   [8,0,2],
                   [8,1,6]])

np.floor_divide(array,array2)
```

Out[29]:
```
array([[1, 1, 2],
       [1, 0, 2],
       [0, 1, 1]], dtype=int32)
```

In [28]:
```
array = np.array([[5,7,4],
                  [8,9,5],
                  [7,1,8]])
array2 = np.array([[4,7,2],
                   [8,0,2],
                   [8,1,6]])


array//array2
```

Out[28]:
```
array([[1, 1, 2],
       [1, 0, 2],
       [0, 1, 1]], dtype=int32)
```

remainder

In [32]:
```
array = np.array([[5,7,4],
                  [8,9,5],
                  [7,1,8]])
array2 = np.array([[4,7,2],
                   [8,0,2],
                   [8,1,6]])

np.remainder(array,array2)
```

Out[32]:
```
array([[1, 0, 0],
       [0, 0, 1],
       [7, 0, 2]], dtype=int32)
```

In [33]:
```
array%array2
```

Out[33]:
```
array([[1, 0, 0],
       [0, 0, 1],
       [7, 0, 2]], dtype=int32)
```

In [34]:
```
array%2
```

Out[34]:
```
array([[1, 1, 0],
       [0, 1, 1],
       [1, 1, 0]], dtype=int32)
```

square()

In [35]:
```
array = np.array([[5,7,4],
                  [8,9,5],
                  [7,1,8]])

array**2
```

Out[35]:
```
array([[25, 49, 16],
       [64, 81, 25],
       [49,  1, 64]], dtype=int32)
```

In [36]:
```
np.square(array)
```

Out[36]:
```
array([[25, 49, 16],
       [64, 81, 25],
       [49,  1, 64]], dtype=int32)
```

In [39]:
```
np.power(array,2)
```

Out[39]:
```
array([[25, 49, 16],
       [64, 81, 25],
       [49,  1, 64]], dtype=int32)
```

sqrt & cbrt

In [41]:
```
array = np.array([[5,7,4],
                  [8,9,5],
                  [7,1,8]])

array**(1/2)
```

Out[41]:
```
array([[2.23606798, 2.64575131, 2.        ],
       [2.82842712, 3.        , 2.23606798],
       [2.64575131, 1.        , 2.82842712]])
```

In [42]:
```
np.sqrt(array)
```

Out[42]:
```
array([[2.23606798, 2.64575131, 2.        ],
       [2.82842712, 3.        , 2.23606798],
       [2.64575131, 1.        , 2.82842712]])
```

In [43]:
```
np.power(array,0.5)
```

Out[43]:
```
array([[2.23606798, 2.64575131, 2.        ],
       [2.82842712, 3.        , 2.23606798],
       [2.64575131, 1.        , 2.82842712]])
```

In [44]:
```
np.cbrt(array)
```

Out[44]:
```
array([[1.70997595, 1.91293118, 1.58740105],
       [2.        , 2.08008382, 1.70997595],
       [1.91293118, 1.        , 2.        ]])
```

Matrix dot product

```
for matrix dot product no. of columns of 1st matrix has to match the no. of rows of 2nd matrix
```

In [45]:
```
matrix1 = np.array([[3,4],
                    [5,6]])
matrix2 = np.array([[1,2],
                    [4,3]])

# 3*1+4*4     3*2+4*3
# 5*1+6*4     5*2+6*3
```

In [46]:
```
np.dot(matrix1,matrix2)
```

Out[46]:
```
array([[19, 18],
       [29, 28]])
```

In [47]:
```
matrix1 = np.array([[3,4],
                    [5,6]])
matrix2 = np.array([[1,2],
                    [4,3],
                    [3,2]])
np.dot(matrix1,matrix2)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-47-1a448a62a92a> in <module>
      4                    [4,3],
      5                    [3,2]])
----> 6 np.dot(matrix1,matrix2)

<__array_function__ internals> in dot(*args, **kwargs)

ValueError: shapes (2,2) and (3,2) not aligned: 2 (dim 1) != 3 (dim 0)
```

In [48]:
```
matrix1 = np.array([[3,4,2],
                    [5,6,1]])
matrix2 = np.array([[1,2],
                    [4,3],
                    [3,2]])
np.dot(matrix1,matrix2)
```

Out[48]:
```
array([[25, 22],
       [32, 30]])
```

In [ ]:
```

```