Mathematic Operation Using numpy

```
import numpy as np
```

```
arr1=np.arange(1,10).reshape(3,3) # creates an array with numbers 1 to
9 and converts into a 3x3 array
arr2=np.arange(1,10).reshape(3,3)
print(arr1)
print(arr2)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

```
arr1+arr2 # element wise addition of two array
```

```
array([[ 2,  4,  6],
       [ 8, 10, 12],
       [14, 16, 18]])
```

```
np.add(arr1,arr2) # element wise addition of two array with add
function
```

```
array([[ 2,  4,  6],
       [ 8, 10, 12],
       [14, 16, 18]])
```

```
arr1-arr2 # element wise subtraction of two array
```

```
array([[0, 0, 0],
       [0, 0, 0],
       [0, 0, 0]])
```

```
np.subtract(arr1,arr2) # element wise subtraction of two array with
subtract function
```

```
array([[0, 0, 0],
       [0, 0, 0],
       [0, 0, 0]])
```

```
arr1/arr2 # division(element wise), default dtype float
```

```
array([[1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.]])
```

```
arr1//arr2 # return floor value after dividion, dtype int
```

```
array([[1, 1, 1],
       [1, 1, 1],
       [1, 1, 1]], dtype=int32)
```

```
np.divide(arr1,arr2) # division(element wise) with divide function
```

```
array([[1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.]])
```

```
#true_divide returns division in dtype int
np.true_divide(arr1,arr2)
# ...................................wanted...........................
..........
```

```
array([[1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.]])
```

```
arr1*arr2 # element wise multiplication
```

```
array([[ 1,  4,  9],
       [16, 25, 36],
       [49, 64, 81]])
```

```
np.multiply(arr1,arr2) # element wise multiplication with multiply
function
```

```
array([[ 1,  4,  9],
       [16, 25, 36],
       [49, 64, 81]])
```

```
arr1@arr2 # matrix multiplication
```

```
array([[ 30,  36,  42],
       [ 66,  81,  96],
       [102, 126, 150]])
```

```
arr1.dot(arr2) # matrix multiplication using .dot function
```

```
array([[ 30,  36,  42],
       [ 66,  81,  96],
       [102, 126, 150]])
```

```
print(arr1)
print(arr1.max()) # shows maximum value present in the array
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
9
```

```
# index of the max value
arr1.argmax()
# .....................................wanted.........................
..........
```

```
8
```

```
print(arr1.max(axis=0)) # max value column wise
print(arr1.max(axis=1)) # max value row wise
```

```
[7 8 9]
[3 6 9]
```

```
arr1.min() # returns minimum value
```

```
1
```

```
arr1.argmin()
```

```
0
```

```
arr1.min(axis=0)
```

```
array([1, 2, 3])
```

```
arr1.min(axis=1)
```

```
array([1, 4, 7])
```

```
np.sum(arr1) # sum of all the elements
```

```
45
```

```
np.sum(arr1,axis=0) # sum of elements column wise
```

```
array([12, 15, 18])
```

```
np.sum(arr1,axis=1)
```

```
array([ 6, 15, 24])
```

```
arr1.mean() # or we can write np.mean(arr1). This returns the average
of the numbers in that array
```

```
5.0
```

```
arr1.mean(axis=0,dtype=int)
```

```
array([4, 5, 6])
```

```
# returns square root of a numbers present in the array
np.sqrt(arr1) #......................................
wanted....................................................
```

```
array([[1.        , 1.41421356, 1.73205081],
       [2.        , 2.23606798, 2.44948974],
       [2.64575131, 2.82842712, 3.        ]])
```

```
np.std(arr1) # ops... I don't even know what is standard deviation
```

```
2.581988897471611
```

```
np.exp(arr1) # returns the exponential of the numbers
```

```
array([[2.71828183e+00, 7.38905610e+00, 2.00855369e+01],
       [5.45981500e+01, 1.48413159e+02, 4.03428793e+02],
       [1.09663316e+03, 2.98095799e+03, 8.10308393e+03]])
```

```
np.log(arr1) # returns normal log means ln or log base(e)X
```

```
array([[0.        , 0.69314718, 1.09861229],
       [1.38629436, 1.60943791, 1.79175947],
       [1.94591015, 2.07944154, 2.19722458]])
```

```
np.log10(arr1) # returns simple logarithm means log base(10)X
```

```
array([[0.        , 0.30103   , 0.47712125],
       [0.60205999, 0.69897   , 0.77815125],
       [0.84509804, 0.90308999, 0.95424251]])
```