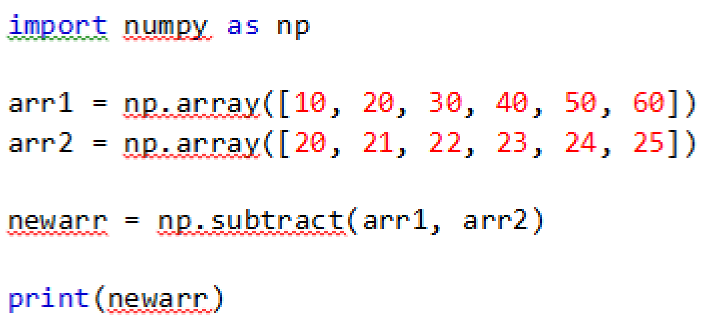
Simple Arithmetic

The subtract() function subtract the content of two arrays, and return the results in a new array. Example:



```
import numpy as np

arr1 = np.array([10, 20, 30, 40, 50, 60])
arr2 = np.array([20, 21, 22, 23, 24, 25])

newarr = np.subtract(arr1, arr2)

print(newarr)
```

Que1:

Create 2 arrays then do the following:

1. Add the values in arr1 to the values in arr2 using np.add()
2. Subtract the values in arr2 from the values in arr1 using np.subtract()
3. Multiply the values in arr1 with the values in arr2 using np.multiply()
4. Divide the values in arr1 with the values in arr2 using np.divide()
5. Raise the values in arr1 to the power of values in arr2 using np.power()
6. Return the quotient and mod considering arr1 and arr2 using np.divmod()
7. Find the absolute value of the array ([-1, -2, 1, 2, 3, -4])- using np.absolute()

In [1]:

```
import numpy as np
```

In [2]:

```
a = np.array([2, 4, 6, 8, 10])
b = np.array([1, 3, 5, 7, 9])
```

In [3]:

```
np.add(a, b), a + b
```

Out[3]:

```
(array([ 3,  7, 11, 15, 19]), array([ 3,  7, 11, 15, 19]))
```

In [4]:

```
np.subtract(a, b), a - b
```

Out[4]:

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

In [6]:

```
np.multiply(a, b), a * b
```

Out[6]:

```
(array([ 2, 12, 30, 56, 90]), array([ 2, 12, 30, 56, 90]))
```

In [7]:

```
np.divide(a, b), a / b
```

Out[7]:

```
(array([2.        , 1.33333333, 1.2       , 1.14285714, 1.11111111]),
 array([2.        , 1.33333333, 1.2       , 1.14285714, 1.11111111]))
```

In [8]:

```
np.power(a, b), a ** b
```

Out[8]:

```
(array([         2,         64,       7776,    2097152, 1000000000],
       dtype=int32),
 array([         2,         64,       7776,    2097152, 1000000000],
       dtype=int32))
```

In [9]:

```
np.divmod(a, b), a // b
```

Out[9]:

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

In [10]:

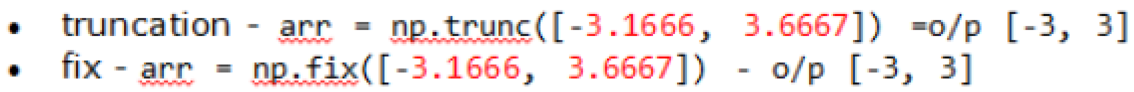
```
a[0] = -9
b[3] = -188
np.absolute(a), np.absolute(b)
```

Out[10]:

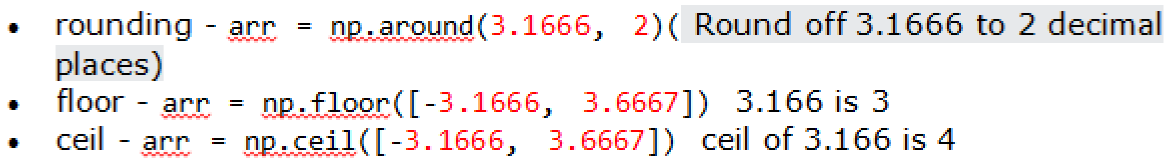
```
(array([ 9,  4,  6,  8, 10]), array([  1,   3,   5, 188,   9]))
```

Rounding Decimals :

There are primarily five ways of rounding off decimals in NumPy:



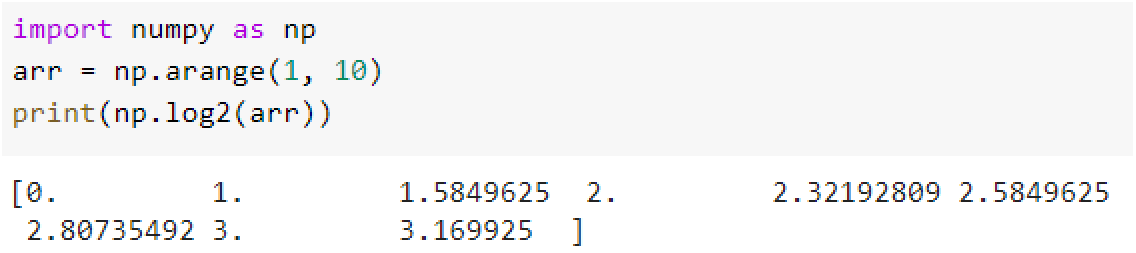
- truncation - arr = np.trunc([-3.1666, 3.6667]) =o/p [-3, 3]
- fix - arr = np.fix([-3.1666, 3.6667]) - o/p [-3, 3]



- rounding - arr = np.around(3.1666, 2)(Round off 3.1666 to 2 decimal places)
- floor - arr = np.floor([-3.1666, 3.6667]) 3.166 is 3
- ceil - arr = np.ceil([-3.1666, 3.6667]) ceil of 3.166 is 4

Logs

NumPy provides functions to perform log at the base 2 and 10



```
import numpy as np
arr = np.arange(1, 10)
print(np.log2(arr))
```

```
[0.         1.         1.5849625  2.         2.32192809 2.5849625
 2.80735492 3.         3.169925  ]
```

Que: Find log at base 10 for all elements from 1 to 10.

In [13]:

```
a = np.arange(1, 10)
print(np.log10(a))
```

```
[0.         0.30103    0.47712125 0.60205999 0.69897    0.77815125
 0.84509804 0.90308999 0.95424251]
```

np.fix - Round to nearest integer towards zero. Remove the decimals, and return the float number closest to zero.

In [19]:

```
a = np.array([-3.166, -3.667, 3.166, 3.667])
np.fix(a), np.trunc(a)
```

Out[19]:

```
(array([-3., -3.,  3.,  3.]), array([-3., -3.,  3.,  3.]))
```

In [20]:

```
np.around(a, 2)
```

Out[20]:

```
array([-3.17, -3.67,  3.17,  3.67])
```

In [22]:

```
np.floor(a)
```

Out[22]:

```
array([-4., -4.,  3.,  3.])
```

In [23]:

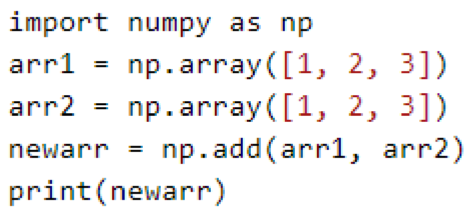
```
np.ceil(a)
```

Out[23]:

```
array([-3., -3.,  4.,  4.])
```

NumPy Summations

Add the values in arr1 to the values in arr2:



```
import numpy as np
arr1 = np.array([1, 2, 3])
arr2 = np.array([1, 2, 3])
newarr = np.add(arr1, arr2)
print(newarr)
```

Que :: Use the same above arrays arr1 and arr2 and find the sum using np.sum() over 1st axis

In [2]:

```
import numpy as np
arr1 = np.array([1, 2, 3])
arr2 = np.array([1, 2, 3])
c = np.sum([arr1, arr2], axis=0)
c
```

Out[2]:

```
array([2, 4, 6])
```

Que:: Perform cummulative summation in the array - [1,2,3,4] using np.cumsum()

In [3]:

```
a = np.array([1, 2, 3, 4])
np.cumsum(a)
```

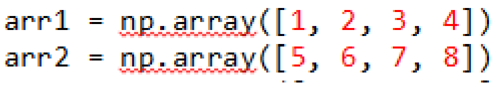
Out[3]:

```
array([ 1,  3,  6, 10], dtype=int32)
```

NumPy Products

QUE

1. Find the product of the elements of array - ([1, 2, 3, 4]) using np.prod()
2. Perform product in the following array over 1st axis



```
arr1 = np.array([1, 2, 3, 4])
arr2 = np.array([5, 6, 7, 8])
```

In [24]:

```
import numpy as np
arr1 = np.array([1, 2, 3, 4])
arr2 = np.array([5, 6, 7, 8])
c = np.prod([arr1, arr2], axis=0)
c
```

Out[24]:

```
array([ 5, 12, 21, 32])
```

QUE:

1. Find the Cummulative product for array ([5, 6, 7, 8]) using np.cumprod()

In [26]:

```
d = np.cumprod(arr2)
d
```

Out[26]:

```
array([   5,   30,  210, 1680], dtype=int32)
```

Differences

A discrete difference means subtracting two successive elements.

- E.g. for [1, 2, 3, 4], the discrete difference would be [2-1, 3-2, 4-3] = [1, 1, 1].
- To find the discrete difference, use the diff() function.

Que:

- Compute discrete difference of the array - ([10, 15, 25, 5]) using np.diff()
- Compute discrete difference of the array ([10, 15, 25, 5]) twice using np.diff(arr, n=2)

In [5]:

```
a = np.array([10, 15, 25, 5])
np.diff(a)
```

Out[5]:

```
array([  5,  10, -20])
```

In [4]:

```
a = np.array([10, 15, 25, 5])
np.diff(a, n=2)
```

Out[4]:

```
array([  5, -30])
```

NumPy GCD Greatest Common Denominator and LCM ¶

Que:

- Find the HCF of the two numbers: 6, 9 using np.gcd()
- Find the LCM of the two numbers: 2,4 using np.lcm()

In [27]:

```
np.gcd(6, 9)
```

Out[27]:

```
3
```

In [28]:

```
np.lcm(2, 4)
```

Out[28]:

```
4
```

In [ ]:

```

```