



Simple Arithmetic

[< Previous](#)[Next >](#)

Simple Arithmetic

You could use arithmetic operators `+` `-` `*` `/` directly between NumPy arrays, but this section discusses an extension of the same where we have functions that can take any array-like objects e.g. lists, tuples etc. and perform arithmetic *conditionally*.

Arithmetic Conditionally: means that we can define conditions where the arithmetic operation should happen.

All of the discussed arithmetic functions take a `where` parameter in which we can specify that condition.

Addition

The `add()` function sums the content of two arrays, and return the results in a new array.

Example

[Get your own Python Server](#)



```
arr1 = np.array([10, 11, 12, 13, 14, 15])
arr2 = np.array([20, 21, 22, 23, 24, 25])

newarr = np.add(arr1, arr2)

print(newarr)
```

Try it Yourself »

The example above will return [30 32 34 36 38 40] which is the sums of 10+20, 11+21, 12+22 etc.

Subtraction

The `subtract()` function subtracts the values from one array with the values from another array, and return the results in a new array.

Example

Subtract the values in arr2 from the values in arr1:

```
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)
```



The example above will return [-10 -1 8 17 26 35] which is the result of 10-20, 20-21, 30-22 etc.

Multiplication

The `multiply()` function multiplies the values from one array with the values from another array, and return the results in a new array.

Example

Multiply the values in arr1 with the values in arr2:

```
import numpy as np

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

newarr = np.multiply(arr1, arr2)

print(newarr)
```

Try it Yourself »

The example above will return [200 420 660 920 1200 1500] which is the result of 10*20, 20*21, 30*22 etc.

Division



Example

Divide the values in arr1 with the values in arr2:

```
import numpy as np

arr1 = np.array([10, 20, 30, 40, 50, 60])
arr2 = np.array([3, 5, 10, 8, 2, 33])

newarr = np.divide(arr1, arr2)

print(newarr)
```

Try it Yourself »

The example above will return [3.33333333 4. 3. 5. 25. 1.81818182] which is the result of 10/3, 20/5, 30/10 etc.

Power

The `power()` function rises the values from the first array to the power of the values of the second array, and return the results in a new array.

Example

Raise the values in arr1 to the power of values in arr2:

```
import numpy as np

arr1 = np.array([10, 20, 30, 40, 50, 60])
arr2 = np.array([3, 5, 6, 8, 2, 33])
```

[Try it Yourself »](#)

The example above will return [1000 3200000 729000000 6553600000000 2500 0]
which is the result of $10*10*10$, $20*20*20*20*20$, $30*30*30*30*30*30$ etc.

Remainder

Both the `mod()` and the `remainder()` functions return the remainder of the values in the first array corresponding to the values in the second array, and return the results in a new array.

Example

Return the remainders:

```
import numpy as np

arr1 = np.array([10, 20, 30, 40, 50, 60])
arr2 = np.array([3, 7, 9, 8, 2, 33])

newarr = np.mod(arr1, arr2)

print(newarr)
```

[Try it Yourself »](#)



You get the same result when using the `remainder()` function:

Example

Return the remainders:

```
import numpy as np

arr1 = np.array([10, 20, 30, 40, 50, 60])
arr2 = np.array([3, 7, 9, 8, 2, 33])

newarr = np.remainder(arr1, arr2)

print(newarr)
```

[Try it Yourself »](#)

Quotient and Mod

The `divmod()` function return both the quotient and the mod. The return value is two arrays, the first array contains the quotient and second array contains the mod.

Example

Return the quotient and mod:

```
import numpy as np

arr1 = np.array([10, 20, 30, 40, 50, 60])
arr2 = np.array([3, 7, 9, 8, 2, 33])

newarr = np.divmod(arr1, arr2)
```



The example above will return:

```
(array([3, 2, 3, 5, 25, 1]), array([1, 6, 3, 0, 0, 27]))
```

The first array represents the quotients, (the integer value when you divide 10 with 3, 20 with 7, 30 with 9 etc.

The second array represents the remainders of the same divisions.

Absolute Values

Both the `absolute()` and the `abs()` functions do the same absolute operation element-wise but we should use `absolute()` to avoid confusion with python's inbuilt `math.abs()`

Example

Return the quotient and mod:

```
import numpy as np

arr = np.array([-1, -2, 1, 2, 3, -4])

newarr = np.absolute(arr)

print(newarr)
```

Try it Yourself »

The example above will return `[1 2 1 2 3 4]`.

[Tutorials ▼](#)[Exercises ▼](#)[Services ▼](#)[Sign Up](#)[Log in](#)[SQL](#) [PYTHON](#) [JAVA](#) [PHP](#) [HOW TO](#) [W3.CSS](#) [C](#) [C++](#) [C#](#) [BOOTSTRAP](#)

Exercise [?]

Consider the following code:

```
import numpy as np
x = [2, 5, 5, 1]
y = [1, 4, 3, 1]
z = np.subtract(x, y)
```

What will be the result of z ?

- ☐ [4 2 2 1]
- ☐ [3 9 8 2]
- ☐ [2 1 5 1]
- ☐ [1 1 2 0]

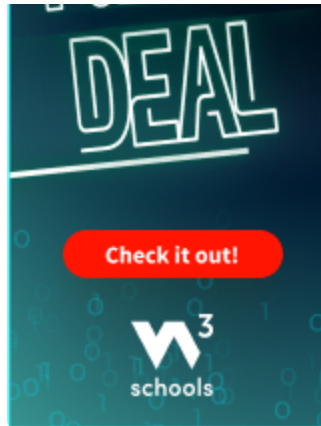
[Submit Answer »](#)

[◀ Previous](#)[Next ▶](#)

Track your progress - it's free!

[Sign Up](#)[Log in](#)

ADVERTISEMENT

[Tutorials ▼](#)[Exercises ▼](#)[Services ▼](#)[Sign Up](#)[Log in](#)[SQL](#)[PYTHON](#)[JAVA](#)[PHP](#)[HOW TO](#)[W3.CSS](#)[C](#)[C++](#)[C#](#)[BOOTSTRA](#)

COLOR PICKER



[Tutorials ▼](#)[Exercises ▼](#)[Services ▼](#)[Sign Up](#)[Log in](#)[SQL](#)[PYTHON](#)[JAVA](#)[PHP](#)[HOW TO](#)[W3.CSS](#)[C](#)[C++](#)[C#](#)[BOOTSTRA](#)[PLUS](#)[SPACES](#)[GET CERTIFIED](#)[FOR TEACHERS](#)[FOR BUSINESS](#)[CONTACT US](#)

Top Tutorials

- [HTML Tutorial](#)
- [CSS Tutorial](#)
- [JavaScript Tutorial](#)
- [How To Tutorial](#)
- [SQL Tutorial](#)
- [Python Tutorial](#)
- [W3.CSS Tutorial](#)
- [Bootstrap Tutorial](#)
- [PHP Tutorial](#)
- [Java Tutorial](#)
- [C++ Tutorial](#)
- [jQuery Tutorial](#)

Top References

- [HTML Reference](#)
- [CSS Reference](#)
- [JavaScript Reference](#)
- [SQL Reference](#)
- [Python Reference](#)
- [W3.CSS Reference](#)

[Tutorials](#) ▼[Exercises](#) ▼[Services](#) ▼[Sign Up](#)[Log in](#)[SQL](#)[PYTHON](#)[JAVA](#)[PHP](#)[HOW TO](#)[W3.CSS](#)[C](#)[C++](#)[C#](#)[BOOTSTRA](#)

Top Examples

- [HTML Examples](#)
- [CSS Examples](#)
- [JavaScript Examples](#)
- [How To Examples](#)
- [SQL Examples](#)
- [Python Examples](#)
- [W3.CSS Examples](#)
- [Bootstrap Examples](#)
- [PHP Examples](#)
- [Java Examples](#)
- [XML Examples](#)
- [jQuery Examples](#)

Get Certified

- [HTML Certificate](#)
- [CSS Certificate](#)
- [JavaScript Certificate](#)
- [Front End Certificate](#)
- [SQL Certificate](#)
- [Python Certificate](#)
- [PHP Certificate](#)
- [jQuery Certificate](#)
- [Java Certificate](#)
- [C++ Certificate](#)
- [C# Certificate](#)
- [XML Certificate](#)

[FORUM](#) [ABOUT](#) [ACADEMY](#)

W3Schools is optimized for learning and training. Examples might be simplified to improve reading and learning.

Tutorials, references, and examples are constantly reviewed to avoid errors, but we cannot warrant full correctness of all content. While using W3Schools, you agree to have read and accepted our [terms of use](#), [cookie and privacy policy](#).

Copyright 1999-2025 by Refsnes Data. All Rights Reserved. [W3Schools is Powered by W3.CSS](#).