9.Scalar Math — Element-wise Operations in NumPy

📘 Overview

NumPy allows you to perform **fast, vectorized arithmetic operations** on arrays — without writing loops.
These operations are **element-wise**, meaning each element of one array is combined with the corresponding element of another array.

Scalar math also supports operations between arrays and constants.

Common Arithmetic Functions

Function	Description
`'np.add(a, b)'`	Element-wise addition
`'np.subtract(a, b)'`	Element-wise subtraction
`'np.multiply(a, b)'`	Element-wise multiplication
`'np.divide(a, b)'`	Element-wise division
`'np.power(a, b)'`	Element-wise exponentiation

All operations can also be written using arithmetic symbols:
`'a + b'`, `'a - b'`, `'a * b'`, `'a / b'`, `'a ** b'`

In [70]:
```
import numpy as np

# Create two sample arrays
a = np.array([10, 20, 30, 40])
b = np.array([1, 2, 3, 4])

print("Array a:", a)
print("Array b:", b)

# Element-wise operations
print("\nAddition (a + b):", np.add(a, b))
print("Subtraction (a - b):", np.subtract(a, b))
print("Multiplication (a * b):", np.multiply(a, b))
print("Division (a / b):", np.divide(a, b))
print("Power (a ** b):", np.power(a, b))
```

```
Array a: [10 20 30 40]
Array b: [1 2 3 4]

Addition (a + b): [11 22 33 44]
Subtraction (a - b): [ 9 18 27 36]
Multiplication (a * b): [ 10  40  90 160]
Division (a / b): [10. 10. 10. 10.]
Power (a ** b): [     10     400   27000 2560000]
```

10. Vector Math Operations

Description

Perform element-wise mathematical operations using **NumPy**.
Each operation is vectorized, meaning it applies to every element of the array efficiently without loops.

Example Arrays

Variable	Definition	Example Values
`a`	First NumPy array	`[1, 2, 3]`
`b`	Second NumPy array	`[4, 5, 6]`

Operations and Examples

Operation	NumPy Function	Description	Example Code	Output
Addition	`np.add(a, b)`	Adds corresponding elements	`np.add(a, b)`	`[5 7 9]`
Multiplication	`np.multiply(a, b)`	Multiplies each element	`np.multiply(a, b)`	`[ 4 10 18]`
Square Root	`np.sqrt(a)`	Finds square root of each element	`np.sqrt(a)`	`[1. 1.4142 1.7320]`
Logarithm	`np.log(b)`	Natural log of each element	`np.log(b)`	`[1.386 1.609 1.791]`
Absolute Value	`np.abs()`	Converts negatives to positives	`np.abs([-1, -2, 3])`	`[1 2 3]`
Ceil	`np.ceil()`	Rounds up to nearest integer	`np.ceil([1.2, 2.7])`	`[2. 3.]`

Operation	NumPy Function	Description	Example Code	Output
Floor	`np.floor()`	Rounds down to nearest integer	`np.floor([1.2, 2.7])`	`[1. 2.]`
Round	`np.round()`	Rounds to nearest integer	`np.round([1.49, 2.51])`	`[1. 3.]`

In [72]:
```python
import numpy as np

a = np.array([1, 2, 3])
b = np.array([4, 5, 6])

print("Addition:", np.add(a, b))
print("Multiplication:", np.multiply(a, b))
print("Square Root:", np.sqrt(a))
print("Logarithm:", np.log(b))
print("Absolute:", np.abs([-1, -2, 3]))
print("Ceil:", np.ceil([1.2, 2.7]))
print("Floor:", np.floor([1.2, 2.7]))
print("Round:", np.round([1.49, 2.51]))
```

```
Addition: [5 7 9]
Multiplication: [ 4 10 18]
Square Root: [1.         1.41421356 1.73205081]
Logarithm: [1.38629436 1.60943791 1.79175947]
Absolute: [1 2 3]
Ceil: [2. 3.]
Floor: [1. 2.]
Round: [1. 3.]
```

11. Statistics in NumPy

Description

NumPy provides built-in **statistical functions** to analyze data quickly and efficiently. These functions can compute summary statistics on **1D** and **2D arrays** with ease.

Common Statistical Functions

Function	Description	Example Usage
`np.mean()`	Calculates the average of array elements	`np.mean(a)`
`np.sum()`	Computes the sum of all elements	`np.sum(a)`
`np.min()`	Returns the minimum element	`np.min(a)`
`np.max()`	Returns the maximum element	`np.max(a)`
`np.var()`	Computes variance of array elements	`np.var(a)`

Function	Description	Example Usage
`np.std()`	Computes standard deviation	`np.std(a)`
`np.corrcoef()`	Calculates correlation coefficients between arrays	`np.corrcoef(a, b)`

Example Arrays

Variable	Definition	Example Values
`a`	1D NumPy array	`[1, 2, 3, 4, 5]`
`b`	2D NumPy array	`[[1, 2, 3], [4, 5, 6]]`

Example Code

In [73]:
```
import numpy as np

# 1D array
a = np.array([1, 2, 3, 4, 5])

# 2D array
b = np.array([[1, 2, 3],
              [4, 5, 6]])

print("1D Array Mean:", np.mean(a))
print("1D Array Sum:", np.sum(a))
print("1D Array Min:", np.min(a))
print("1D Array Max:", np.max(a))
print("1D Array Variance:", np.var(a))
print("1D Array Std Dev:", np.std(a))

# 2D Array Operations
print("\n2D Array Mean:", np.mean(b))
print("2D Array Sum:", np.sum(b))
print("2D Array Min:", np.min(b))
print("2D Array Max:", np.max(b))
print("2D Array Variance:", np.var(b))
print("2D Array Std Dev:", np.std(b))
```

```
1D Array Mean: 3.0
1D Array Sum: 15
1D Array Min: 1
1D Array Max: 5
1D Array Variance: 2.0
1D Array Std Dev: 1.4142135623730951

2D Array Mean: 3.5
2D Array Sum: 21
2D Array Min: 1
2D Array Max: 6
2D Array Variance: 2.9166666666666665
2D Array Std Dev: 1.707825127659933
```


Kudum Veerabhadraiah

Data Science and AI Enthusiast