numpy-dsp

August 31, 2024

[1]:
```
import numpy as np
```

[2]:
```
# Creating a 1D array
arr1 = np.array([1, 2, 3, 4, 5])
# Creating a 2D array
arr2 = np.array([[1, 2, 3], [4, 5, 6]])
```

basic operation

[3]:
```
# Addition
arr_sum = arr1 + 10
print(arr_sum,'\n')
# Multiplication
arr_mult = arr1 * 2
print(arr_mult,'\n')
# Dot product (for 2D arrays)
arr_dot = np.dot(arr2, arr1[:3])
print(arr_dot,'\n')
```

```
[11 12 13 14 15]

[ 2  4  6  8 10]

[14 32]
```

Array Properties

[4]:
```
print(arr1.shape,'\n')
print(arr2.shape,'\n')
print(arr1.size,'\n')
print(arr2.dtype,'\n')
```

```
(5,)

(2, 3)

5
```

```
int64
```

2.Data Manipulation:

[5]:
```
# Create a 3x3 array
arr = np.arange(9).reshape(3, 3)
print(arr,'\n')
```

```
[[0 1 2]
 [3 4 5]
 [6 7 8]]
```

[6]:
```
# Indexing
element = arr[1, 2]  # Get element from 2nd row, 3rd column
print(element,'\n')
```

```
5
```

[7]:
```
# Slicing
slice_arr = arr[:2, 1:]  # First two rows and last two columns
print(slice_arr,'\n')
```

```
[[1 2]
 [4 5]]
```

[8]:
```
# Reshaping
reshaped_arr = arr.reshape(1, 9)  # Reshape to 1x9 array
print(reshaped_arr,'\n')
```

```
[[0 1 2 3 4 5 6 7 8]]
```

[9]:
```
# Element-wise square root
sqrt_arr = np.sqrt(arr)
print(sqrt_arr,'\n')
```

```
[[0.         1.         1.41421356]
 [1.73205081 2.         2.23606798]
 [2.44948974 2.64575131 2.82842712]]
```

[10]:
```
# Logarithm
log_arr = np.log(arr + 1)  # Adding 1 to avoid log(0)
print(log_arr,'\n')
```

```
[[0.         0.69314718 1.09861229]
 [1.38629436 1.60943791 1.79175947]
 [1.94591015 2.07944154 2.19722458]]
```

3.Data Aggregation

[11]:
```
# Compute mean
mean_value = np.mean(arr)
print(mean_value,'\n')
```

```
4.0
```

[12]:
```
# Compute median
median_value = np.median(arr)
print(median_value,'\n')
```

```
4.0
```

[13]:
```
# Compute standard deviation
std_dev = np.std(arr)
print(std_dev,'\n')
```

```
2.581988897471611
```

[14]:
```
# Grouping: Separate even and odd numbers
even_numbers = arr[arr % 2 == 0]
print(even_numbers,'\n')
odd_numbers = arr[arr % 2 != 0]
print(odd_numbers,'\n')
```

```
[0 2 4 6 8]

[1 3 5 7]
```

[15]:
```
# Aggregation on groups
sum_even = np.sum(even_numbers)
print(sum_even,'\n')
sum_odd = np.sum(odd_numbers)
print(sum_odd,'\n')
```

```
20

16
```

4.Data Analysis

[17]:
```
#Create two arrays
x = np.array([1, 2, 3, 4, 5])
y = np.array([2, 4, 6, 8, 10])
```

[18]:
```
# Correlation coefficient
correlation = np.corrcoef(x, y)
print(correlation,'\n')
```

```
[[1. 1.]
 [1. 1.]]
```

[19]:
```
data = np.array([1, 2, 3, 4, 100, 6, 7, 8, 9])


# Calculate the z-scores
z_scores = (data - np.mean(data)) / np.std(data)
print(z_scores,'\n')
```

```
[-0.4857185  -0.45234853 -0.41897856 -0.38560858  2.81790887 -0.31886864
 -0.28549866 -0.25212869 -0.21875871]
```

[20]:
```
#Identify outliers
outliers = data[np.abs(z_scores) > 2]
print(outliers,'\n')
```

```
[100]
```

[21]:
```
# Calculate the 25th, 50th, and 75th percentiles
percentiles = np.percentile(data, [25, 50, 75])
print(percentiles,'\n')
```

```
[3. 6. 8.]
```

5.Application in Data Science

Advantages of Using NumPy:

Efficiency

:NumPy is faster than regular Python lists due to its underlying C implementation and efficient memory usage.

Convenience: Provides a broad range of mathematical functions.

Interoperability: Integrates seamlessly with other Python libraries like Pandas, Scikit-learn, and TensorFlow, which are heavily used in data science.

Real-World Examples: Machine Learning: NumPy is the backbone of many machine learning libraries. It's used for data preprocessing, implementing algorithms, and even deep learning (e.g., TensorFlow).

Financial Analysis: NumPy's efficient handling of large datasets is critical for analyzing stock prices, calculating moving averages, and performing Monte Carlo simulations.

Scientific Research: Researchers rely on NumPy for processing large datasets, performing simulations, and analyzing experimental data.

Conclusion: NumPy is a powerful tool for any data science professional. Its ability to handle large datasets efficiently and perform complex numerical operations makes it an essential library in the data science toolkit.

[ ]: