**Reshaping NumPy Array**

**1. Reshape NumPy Array:**

*Definition:* Reshaping a NumPy array involves modifying its dimensions or structure while preserving the total number of elements. It allows you to transform the shape of an array without changing its data.

*Example:*

import numpy as np

# Original array

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

[4, 5, 6]])

# Reshape the array to a 3x2 matrix

reshaped\_arr = np.reshape(arr, (3, 2))

print("Original Array:")

print(arr)

print("\nReshaped Array:")

print(reshaped\_arr)

*Explanation:*

Here, we reshape a 2x3 array arr into a 3x2 array using the reshape() function. The new shape (3, 2) specifies 3 rows and 2 columns, and the resulting array is printed.

**2. Python | Numpy matrix.resize():**

The resize() method in NumPy modifies the shape and size of an array in-place. It adds or removes elements as necessary to achieve the new shape.

*Example:*

import numpy as np

# Original array

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

[4, 5, 6]])

# Resize the array to 3x3

arr.resize((3, 3))

print("Resized Array:")

print(arr)

*Explanation:*

This example resizes a 2x3 array arr to a 3x3 array using the resize() method. Additional elements are added with zero values to match the new shape.

**3. Python | Numpy matrix.reshape():**

The reshape() method in NumPy returns a new array with a specified shape, without changing the data of the original array.

*Example:*

import numpy as np

# Original array

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

[4, 5, 6]])

# Reshape the array to a 3x2 matrix

reshaped\_arr = arr.reshape((3, 2))

print("Original Array:")

print(arr)

print("\nReshaped Array:")

print(reshaped\_arr)

*Explanation:*

Similar to the previous example, we reshape a 2x3 array arr into a 3x2 array using the reshape() method. The new shape (3, 2) specifies 3 rows and 2 columns, and the resulting array is printed.

**4. NumPy Array Shape:**

The shape of a NumPy array describes its dimensions. It is represented as a tuple of integers indicating the size of the array along each dimension.

*Example:*

import numpy as np

# Original array

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

[4, 5, 6]])

# Get the shape of the array

shape = arr.shape

print("Shape of the Array:", shape)

*Explanation:*

Here, we retrieve the shape of the array arr using the shape attribute. The shape is returned as a tuple (2, 3), indicating 2 rows and 3 columns.

**5. Change the Dimension of a NumPy Array:**

Changing the dimension of a NumPy array involves altering its shape or structure to suit specific requirements or operations.

*Example:*

import numpy as np

# Original array

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

[4, 5, 6]])

# Change the dimension of the array to 1D

new\_arr = arr.ravel()

print("Original Array:")

print(arr)

print("\n1D Array:")

print(new\_arr)

*Explanation:*

Here, we convert a 2D array arr into a 1D array using the ravel() method. The resulting array new\_arr contains all elements of the original array flattened into a single row.

**6. numpy.ndarray.resize() function - Python:**

The resize() function in NumPy modifies the shape and size of an array in-place, allowing for the addition or removal of elements to achieve the new shape.

*Example:*

import numpy as np

# Original array

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

[4, 5, 6]])

# Resize the array to 3x3

arr.resize((3, 3))

print("Resized Array:")

print(arr)

*Explanation:*

Similar to the previous example, this code resizes a 2x3 array arr to a 3x3 array using the resize() function. Additional elements are added with zero values to match the new shape.

**7. Flatten a Matrix in Python using NumPy:**

Flattening a matrix involves converting a multi-dimensional array into a one-dimensional array by combining all its elements.

*Example:*

import numpy as np

# Original array

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

[4, 5, 6]])

# Flatten the array

flattened\_arr = arr.flatten()

print("Original Array:")

print(arr)

print("\nFlattened Array:")

print(flattened\_arr)

*Explanation:*

Here, we flatten a 2D array arr into a one-dimensional array using the flatten() method. All elements are combined into a single row.

**8. numpy.moveaxis() function | Python:**

The moveaxis() function in NumPy rearranges the dimensions of an array, allowing you to specify the new order of axes.

*Example:*

import numpy as np

# Original array with shape (2, 3, 4)

arr = np.zeros((2, 3, 4))

# Move the second axis to the first position

new\_arr = np.moveaxis(arr, 1, 0)

print("Original Array Shape:", arr.shape)

print("New Array Shape:", new\_arr.shape)

*Explanation:*

This example demonstrates moving the second axis of a 3D array arr to the first position using the moveaxis() function. The resulting array new\_arr has a different axis order.

**9. numpy.swapaxes() function | Python:**

The swapaxes() function in NumPy swaps the two specified axes of an array, altering its shape and structure.

*Example:*

import numpy as np

# Original array with shape (2, 3, 4)

arr = np.zeros((2, 3, 4))

# Swap the first and last axes

new\_arr = np.swapaxes(arr, 0, 2)

print("Original Array Shape:", arr.shape)

print("New Array Shape:", new\_arr.shape)

*Explanation:*

Here, we swap the first and last axes of a 3D array arr using the swapaxes() function. The resulting array new\_arr has its axes reordered accordingly.

**Conclusion:**

In this detailed guide, we've explored various techniques for reshaping NumPy arrays, including the reshape() function, the resize() method, flattening arrays, and manipulating axes using functions like moveaxis() and swapaxes().

Understanding these methods is essential for efficient data manipulation and preprocessing tasks in Python, particularly in scientific computing, machine learning, and data analysis applications. By mastering array reshaping techniques, you can optimize your workflows and effectively handle complex data structures with NumPy.