**Title: NumPy Array Manipulation Techniques**

NumPy is a powerful library in Python for numerical computing, and its array manipulation capabilities make it a favourite among data scientists and researchers. In this blog post, we will delve into various array manipulation techniques offered by NumPy.

**1. NumPy Copy and View of Array:** NumPy arrays support both shallow and deep copies. Understanding the difference between copying and viewing arrays is crucial for efficient memory management.

**How to Copy NumPy array into another array?** To make a deep copy of a NumPy array, you can use the copy() method:

import numpy as np

arr1 = np.array([1, 2, 3])

arr2 = arr1.copy().

print(arr2) # Output: [1 2 3]

**2. Appending values at the end of a NumPy array:** To append values to the end of a NumPy array, you can use the append() method:

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

arr = np.append(arr, [4, 5, 6])

print(arr) # Output: [1 2 3 4 5 6]

**3. How to swap columns of a given NumPy array?** To swap columns of a NumPy array, you can directly assign the columns:

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

arr[:, [0, 1]] = arr[:, [1, 0]]

print(arr)

# Output:

# [[2 1 3]

# [5 4 6]]

**3. Insert a new axis within a NumPy array:** You can insert a new axis within a NumPy array using the expand\_dims() method:

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

arr = np.expand\_dims(arr, axis=1)

print(arr)

# Output:

# [[1]

# [2]

# [3]]

**4. numpy.hstack() and numpy.vstack() in Python:** These functions are used to horizontally and vertically stack arrays, respectively.

arr1 = np.array([1, 2, 3])

arr2 = np.array([4, 5, 6])

hstacked = np.hstack((arr1, arr2))

vstacked = np.vstack((arr1, arr2))

print(hstacked) # Output: [1 2 3 4 5 6]

print(vstacked)

# Output:

# [[1 2 3]

# [4 5 6]]

**5. Joining NumPy Arrays:** Joining NumPy arrays is essential when dealing with multidimensional arrays.

**6. Combining a one and a two-dimensional NumPy Array:** To combine a one-dimensional and a two-dimensional NumPy array, you can use np.concatenate():

arr1 = np.array([1, 2, 3])

arr2 = np.array([[4, 5, 6], [7, 8, 9]])

combined = np.concatenate((arr1.reshape(1, -1), arr2), axis=0)

print(combined)

# Output:

# [[1 2 3]

# [4 5 6]

# [7 8 9]]

**Python | Numpy np.ma.concatenate() method:** np.ma.concatenate() is used to concatenate masked arrays along a specified axis.

**Python | Numpy dstack() method:** np.dstack() stacks arrays along the third axis.

**8. Splitting Arrays in NumPy:** You can split arrays in NumPy using split() method.

**9. How to compare two NumPy arrays?** To compare two NumPy arrays, you can use the array\_equal() method.

arr1 = np.array([1, 2, 3])

arr2 = np.array([1, 2, 3])

print(np.array\_equal(arr1, arr2)) # Output: True

**10. Find the union of two NumPy arrays:** To find the union of two NumPy arrays, you can use np.union1d() method.

arr1 = np.array([1, 2, 3])

arr2 = np.array([3, 4, 5])

union = np.union1d(arr1, arr2)

print(union) # Output: [1 2 3 4 5]

**11. Find unique rows in a NumPy array:** To find unique rows in a NumPy array, you can use np.unique() method.

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

unique\_rows = np.unique(arr, axis=0)

print(unique\_rows)

# Output:

# [[1 2]

# [2 3]

# [3 4]]

**12. numpy.trim\_zeros() in Python:** np.trim\_zeros() is used to trim the leading and/or trailing zeros from a NumPy array.

These array manipulation techniques provide powerful tools for handling data efficiently in Python using NumPy. Whether you're comparing arrays, joining them, or splitting them, NumPy offers a comprehensive set of functions to meet your needs.