Practical 10

10 Numpy Operations

Aim:

np.array(): Create a NumPy array from a list or tuple.
np.zeros(): Create an array filled with zeros.
np.ones(): Create an array filled with ones.
np.arange(): Create an array with a range of values.
np.linspace(): Create an array with evenly spaced values over a specified interval.
np.reshape(): Change the shape of an array without changing its data.
np.flatten(): Convert a multi-dimensional array into a 1D array.
np.transpose(): Transpose an array (swap rows and columns).
np.concatenate(): Join two or more arrays along an axis.
np.split(): Split an array into multiple sub-arrays.
np.vstack(): Stack arrays vertically (row-wise).
np.hstack(): Stack arrays horizontally (column-wise).
np.append(): Append values to the end of an array.
np.delete(): Delete elements from an array along a specified axis.
np.insert(): Insert values into an array at specified positions.
arr[i]: Access an element of an array at index i.
arr[i:j]: Slice an array from index i to j (exclusive).
arr[start:end:step]: Slice an array with a step between start and end.
arr[:, :]: Access all elements along rows and columns (for 1D, 2D, and 3D arrays).

Code:

```
import numpy as np

# Basic Array Creation
print("=== Basic Array Creation ===")
# Create array from list
arr1 = np.array([1, 2, 3, 4, 5])
print("np.array():", arr1)

# Create arrays with zeros and ones
zeros_arr = np.zeros((3, 4))  # 3x4 array of zeros
ones_arr = np.ones((2, 3))    # 2x3 array of ones
print("\nnp.zeros():\n", zeros_arr)
print("\nnp.ones():\n", ones_arr)

# Create array with range
range_arr = np.arange(0, 10, 2)  # From 0 to 10 (exclusive) with step 2
print("\nnp.arange():", range_arr)

# Create array with evenly spaced values
linspace_arr = np.linspace(0, 1, 5)  # 5 values from 0 to 1 (inclusive)
print("\nnp.linspace():", linspace_arr)

# Array Reshaping
```

```python
print("\n=== Array Reshaping ===")
original = np.arange(12)
reshaped = original.reshape((3, 4))  # Reshape to 3x4
print("Original array:", original)
print("\nReshaped to 3x4:\n", reshaped)

# Flatten a multi-dimensional array
flattened = reshaped.flatten()
print("\nFlattened array:", flattened)

# Transpose (swap rows and columns)
transposed = reshaped.transpose()
print("\nTransposed array:\n", transposed)

# Array Joining and Splitting
print("\n=== Array Joining and Splitting ===")
arr_a = np.array([1, 2, 3])
arr_b = np.array([4, 5, 6])

# Concatenate arrays
concatenated = np.concatenate((arr_a, arr_b))
print("Concatenated:", concatenated)

# Split array
arr_to_split = np.array([1, 2, 3, 4, 5, 6])
split_arrays = np.split(arr_to_split, 3)  # Split into 3 equal parts
print("\nSplit into 3:", [arr for arr in split_arrays])

# Vertical and horizontal stacking
print("\nVertical stack (vstack):")
v_stacked = np.vstack((arr_a, arr_b))
print(v_stacked)

print("\nHorizontal stack (hstack):")
h_stacked = np.hstack((arr_a, arr_b))
print(h_stacked)

# Array Modification
print("\n=== Array Modification ===")
base_arr = np.array([10, 20, 30, 40, 50])

# Append values
appended = np.append(base_arr, [60, 70])
print("Appended:", appended)
```

```
# Delete elements (delete element at index 2)
deleted = np.delete(base_arr, 2)
print("\nAfter deleting index 2:", deleted)

# Insert value
inserted = np.insert(base_arr, 2, 25)  # Insert 25 at index 2
print("\nAfter inserting 25 at index 2:", inserted)

# Array Indexing and Slicing
print("\n=== Array Indexing and Slicing ===")
sample = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
print("Sample array:", sample)

# Access single element
print("\nElement at index 3:", sample[3])

# Slice array
print("Slice [2:6]:", sample[2:6])  # Get elements from index 2 to 5

# Slice with step
print("Slice [1:9:2]:", sample[1:9:2])  # Get every other element from
index 1 to 8

# 2D array slicing
matrix = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print("\n2D array:\n", matrix)
print("\nAll elements in row 1:", matrix[1, :])
print("All elements in column 2:", matrix[:, 2])
print("Submatrix (first 2 rows, last 2 columns):\n", matrix[0:2, 1:3])
```

Output Screenshot:

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS COMMENTS SQL HISTORY TASK MONITOR

```
❯ python3 -u "/Users/debdootmanna/VSCode/Python/10.py"
=== Basic Array Creation ===
np.array(): [1 2 3 4 5]

np.zeros():
 [[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]

np.ones():
 [[1. 1. 1.]
 [1. 1. 1.]]

np.arange(): [0 2 4 6 8]

np.linspace(): [0.   0.25 0.5  0.75 1.  ]

=== Array Reshaping ===
Original array: [ 0  1  2  3  4  5  6  7  8  9 10 11]

Reshaped to 3x4:
 [[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]

Flattened array: [ 0  1  2  3  4  5  6  7  8  9 10 11]

Transposed array:
 [[ 0  4  8]
 [ 1  5  9]
 [ 2  6 10]
 [ 3  7 11]]

=== Array Joining and Splitting ===
Concatenated: [1 2 3 4 5 6]

Split into 3: [array([1, 2]), array([3, 4]), array([5, 6])]

Vertical stack (vstack):
[[1 2 3]
 [4 5 6]]

Horizontal stack (hstack):
[1 2 3 4 5 6]

=== Array Modification ===
Appended: [10 20 30 40 50 60 70]

After deleting index 2: [10 20 40 50]

After inserting 25 at index 2: [10 20 25 30 40 50]

=== Array Indexing and Slicing ===
Sample array: [ 1  2  3  4  5  6  7  8  9 10]

Element at index 3: 4
Slice [2:6]: [3 4 5 6]
Slice [1:9:2]: [2 4 6 8]

2D array:
 [[1 2 3]
 [4 5 6]
 [7 8 9]]

All elements in row 1: [4 5 6]
All elements in column 2: [3 6 9]
Submatrix (first 2 rows, last 2 columns):
 [[2 3]
 [5 6]]

  ~/VSCode/Python  on  main ?12
❯
```

Conclusion/Summary:

In this practical assignment, I explored NumPy, Python's fundamental library for numerical computing. I learned various operations including array creation (using array(), zeros(), ones(), arange(), and linspace()), reshaping techniques, joining and splitting arrays, array modification, and indexing/slicing methods.

NumPy provides efficient data structures and operations for working with arrays, making it significantly faster than Python's built-in lists for numerical computations. The library's capabilities allow for seamless manipulation of multi-dimensional arrays, which is essential for data analysis, scientific computing, and machine learning applications.

Through hands-on practice with these operations, I've gained practical experience with NumPy's core functionality that will serve as a foundation for more advanced data processing tasks. This knowledge is particularly valuable as NumPy forms the basis for many other data science libraries like pandas, matplotlib, and scikit-learn.

Overall, this assignment has enhanced my understanding of numerical computing in Python and prepared me for more complex data manipulation challenges in future projects.

Student Signature & Date	**Marks:**	**Evaluator Signature & Date**