5. Stacking and Concatenating Numpy Arrays:

-> Stacking involves adding a new dimension and combining arrays into a higher-dimensional array.

Using 'np.concatenate()'

```
import numpy as np


# Create two 1D arrays
array1 = np.array([1, 2, 3])
array2 = np.array([4, 5, 6])


# Concatenate arrays
result = np.concatenate((array1, array2))
print("Concatenated array:\n", result)
```

 # Output:

```
# Concatenated array:
# [1 2 3 4 5 6]
```

Using np.hstack() for Horizontal Concatenation

```
import numpy as np


# Create two 2D arrays
array1 = np.array([[1, 2], [3, 4]])
array2 = np.array([[5, 6], [7, 8]])


# Concatenate arrays horizontally
result = np.hstack((array1, array2))
print("Concatenated array horizontally:\n", result)
```

 # Output:

Concatenated array horizontally:

[[1 2 5 6]

[3 4 7 8]]

Using np.vstack() for Vertical Concatenation

```
import numpy as np

# Create two 2D arrays
array1 = np.array([[1, 2], [3, 4]])
array2 = np.array([[5, 6], [7, 8]])

# Concatenate arrays vertically
result = np.vstack((array1, array2))
print("Concatenated array vertically:\n", result)
```

 # Output:

Concatenated array vertically:

[[1 2]

[3 4]

[5 6]

[7 8]]

Using np.dstack() for Depth Concatenation

```
import numpy as np

# Create two 2D arrays
array1 = np.array([[1, 2], [3, 4]])
```

```
array2 = np.array([[5, 6], [7, 8]])


# Concatenate arrays along depth axis
result = np.dstack((array1, array2))
print("Concatenated array along depth:\n", result)


# Output:
# Concatenated array along depth:
# [[[1 5]
#   [2 6]]
#  [[3 7]
#   [4 8]]]
```

➔ • **Stacking ndarrays**

Using np.stack()

The np.stack() function is versatile and can stack arrays along any axis. Here's how you can use it to stack NumPy arrays:

```
import numpy as np


# Create two 1D arrays
array1 = np.array([1, 2, 3])
array2 = np.array([4, 5, 6])


# Stack arrays along rows (axis=0)
result_rows = np.stack((array1, array2), axis=0)
print("Stacked array along rows (axis=0):\n", result_rows)


# Output:
```

```
# Stacked array along rows (axis=0):
# [[1 2 3]
#  [4 5 6]]

# Stack arrays along columns (axis=1)
result_cols = np.stack((array1, array2), axis=1)
print("Stacked array along columns (axis=1):\n", result_cols)

# Output:
# Stacked array along columns (axis=1):
# [[1 4]
#  [2 5]
#  [3 6]]
```

Using np.column_stack() and np.row_stack()

For specific cases where you want to stack arrays column-wise or row-wise, you can use np.column_stack() and np.row_stack(), respectively.

```
import numpy as np

# Create two 1D arrays
array1 = np.array([1, 2, 3])
array2 = np.array([4, 5, 6])

# Stack arrays column-wise
result_cols = np.column_stack((array1, array2))
print("Stacked array column-wise:\n", result_cols)

# Output:
# Stacked array column-wise:
# [[1 4]
#  [2 5]
```

```
#  [3 6]]

# Stack arrays row-wise
result_rows = np.row_stack((array1, array2))
print("Stacked array row-wise:\n", result_rows)

# Output:
# Stacked array row-wise:
# [[1 2 3]
#  [4 5 6]]
```

Stacking Multiple Arrays

You can stack more than two arrays using np.stack() by passing them as a sequence.

```
import numpy as np

# Create three 1D arrays
array1 = np.array([1, 2, 3])
array2 = np.array([4, 5, 6])
array3 = np.array([7, 8, 9])

# Stack arrays along rows (axis=0)
result_rows = np.stack((array1, array2, array3), axis=0)
print("Stacked array along rows (axis=0):\n", result_rows)

# Output:
# Stacked array along rows (axis=0):
# [[1 2 3]
#  [4 5 6]
#  [7 8 9]]
```

Stacking 2D and 3D Arrays

You can also stack 2D and 3D arrays using np.stack(), ensuring that the arrays have compatible shapes for the specified axis.

```
import numpy as np

# Create two 2D arrays

array1 = np.array([[1, 2], [3, 4]])

array2 = np.array([[5, 6], [7, 8]])


# Stack arrays along a new axis (axis=0)

result = np.stack((array1, array2), axis=0)

print("Stacked 2D arrays along a new axis (axis=0):\n", result)

# Output:

# Stacked 2D arrays along a new axis (axis=0):

# [[[1 2]

#   [3 4]]

#  [[5 6]

#   [7 8]]]


# Create two 3D arrays

array1 = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])

array2 = np.array([[[9, 10], [11, 12]], [[13, 14], [15, 16]]])


# Stack arrays along a new axis (axis=0)

result = np.stack((array1, array2), axis=0)

print("Stacked 3D arrays along a new axis (axis=0):\n", result)

# Output:

# Stacked 3D arrays along a new axis (axis=0):

# [[[[ 1  2]

#    [ 3  4]]
```

```
#  [[ 5  6]
#   [ 7  8]]]
#
# [[[ 9 10]
#   [11 12]]
#  [[13 14]
#   [15 16]]]]
```

➔ **Broadcasting in Numpy Arrays**

How Broadcasting Works

1. **Dimension Comparison**: NumPy compares the dimensions of the arrays from right to left (i.e., starting with the trailing dimensions).
2. **Compatibility Rules**: Two dimensions are compatible if they are equal or if one of them is 1. If these conditions are not met, a ValueError is raised.
3. **Broadcasting Process**: The smaller array is broadcasted (replicated) along the larger array's dimensions to match its shape.

Examples of Broadcasting

Example 1: 1D Array and 2D Array

```
import numpy as np

# Create a 1D array
array1 = np.array([1, 2, 3])

# Create a 2D array
array2 = np.array([[4, 5, 6], [7, 8, 9]])

# Perform addition
result = array1 + array2
print(result)

# Output:
# [[5 7 9]
#  [8 10 12]]
```

Example 2: Scalar and Array

```
import numpy as np

# Create a scalar
scalar = 5

# Create a 1D array
array1 = np.array([1, 2, 3])

# Perform addition
result = array1 + scalar

print(result)

# Output:
# [6 7 8]
```

Example 3: 2D Array and 1D Array

```
import numpy as np

# Create a 2D array
array1 = np.array([[1, 2], [3, 4]])

# Create a 1D array
array2 = np.array([5, 6])

# Perform addition
result = array1 + array2

print(result)

# Output:
# [[6 8]
#  [8 10]]
```