Lecture Plan: 04 - Array Manipulation

1. Transposing the array
2. Reshaping the array
   - `reshape()`
   - `flatten()`
3. Append, Insert & Delete
4. Combining arrays in numpy
   - `vstack()`
   - `hstack()`
5. Splitting arrays in numpy
   - `vsplit()`
   - `hsplit()`

1. Transposing the array

Explain that the transpose of the 2D array is another 2D array with same data but with rows and columns interchanged.

Note: You can pick any example. However, the example mentioned in student notes is as follows:

Consider a scenario where you have collected data on the daily sales of different products over a period of time. You store this data in a 2D NumPy array, where each row represents a specific product, and each column represents the sales for a particular day. Now, let's say you want to analyze the sales by days rather than by products. In other words, you want to transform the array so that each row represents a specific day, and each column represents the sales of different products on that day. The `transpose` function or `.T` attribute rearranges the dimensions of the array, effectively swapping the rows with columns. In this case, it converts the original (3, 5) array into a transposed (5, 3) array. Now, each row represents a specific day, and each column represents the sales of different products on that day. This transformed array allows you to easily analyze and compare the sales across different days for each product.

```
# Array representing the daily sales of three products (A, B, and
C) over five days
sales_data = np.array([[10, 15, 12, 18, 20],
                       [8, 10, 14, 16, 12],
                       [5, 7, 9, 6, 8]])

transposed_data = sales_data.T
print(transposed_data)
```

2. Reshaping the data

Take a 1D array and then using the `reshape()` function, reshape it to 2D array. Then convert it back to 1D array using `flatten()`.

Note: You can pick any example. However, the example mentioned in student notes is as follows:

Consider a scenario where you have collected data on the students' scores in a classroom. You have stored this data in a 1D NumPy array, where each element represents the score of an individual student. Now, let's say you want to analyze the scores in a 2D format, where each row represents a student, and each column represents a specific attribute related to the student (e.g., score, grade, attendance, etc.). Reshaping the array to 2D can provide a structured representation of the data.

```
scores = np.array([82, 75, 90, 88, 92, 78, 85, 80, 87, 95, 86,
91, 84, 89, 93])
reshaped_scores = scores.reshape((5, 3))
print(reshaped_scores)

flattened_scores = reshaped_scores.flatten()
print(flattened_scores)
```

3. Append, Insert, Delete

Explain these functions from numpy. Here is a quick summary for you:

1. **Append:** It allows you to add new data to the end of an array. The append function takes two arguments: the original array and the element(s) you want to append.
2. **Insert:** It allows you to add new data at a specified index while shifting the remaining elements accordingly. The insert function takes three arguments: the original array, the index at which you want to insert the element(s), and the element(s) you want to insert.
3. **Delete:** It allows you to remove unwanted data and resize the array accordingly. The delete function in NumPy takes two arguments: the original array and the index or indices of the element(s) you want to delete.

Note: You can pick any example. However, the example mentioned in student notes is as follows:

```
temperatures = np.array([24.5, 23.8, 26.1, 25.6, 24.9])

updated_temperatures = np.append(temperatures, 27.3)
print(updated_temperatures)

updated_temperatures = np.insert(temperatures, 3, 25.2)
print(updated_temperatures)

updated_temperatures = np.delete(temperatures, 1)
print(updated_temperatures)
```

4. Combining arrays in Numpy

Explain that we need to often combine two arrays into one, either horizontally or vertically.

Note: You can pick any example. However, the example mentioned in student notes is as follows:

Consider a scenario where you have two arrays representing the heights and weights of individuals. Using the vstack function, you can merge the heights and weights arrays to create a new array representing the height and weight measurements of the individuals.

```
heights = np.array([165, 170, 155, 180])
weights = np.array([68, 72, 60, 85])

measurements = np.vstack((heights, weights))
print(measurements)
```

Continuing with the previous example, let's say you have another array representing the ages of the individuals.

```
ages = np.array([25, 32, 28, 36])
data = np.hstack((heights.reshape(-1, 1), weights.reshape(-1, 1),
ages.reshape(-1, 1)))
print(data)
```

Now, the data array has four rows, where the first column represents the heights, the second column represents the weights, and the third column represents the ages of the individuals.

5. Splitting arrays in Numpy

Explain that we need to split one array into subarrays, either horizontally or vertically.

Note: You can pick any example. However, the example mentioned in student notes is as follows:

Consider a scenario where you have a 2D array representing the temperatures recorded at different locations for each month of the year. After vsplitting, `split_rows[0]` represents the temperatures for the first half of the year (first 2 rows) and `split_rows[1]` represents the temperatures for the second half of the year (last 2 rows). Similarly, after hsplitting, `split_columns[0]` represents the temperatures for the first half of the locations (first 2 columns) and `split_columns[1]` represents the temperatures for the second half of the locations (last 2 columns).

```
temperatures = np.array([[25, 28, 30, 26],
                         [22, 24, 26, 20],
                         [28, 30, 32, 29],
                         [20, 23, 25, 22]])

split_rows = np.vsplit(temperatures, 2)
print(split_rows)
```

```
split_columns = np.hsplit(temperatures, 2)
print(split_columns)
```