Lecture Plan: 01 - Introduction to Numpy

Need to cover following topics:

1. Introduction to Numpy
    - What is Numpy?
    - Why should we use it?
2. How to use numpy
    - Installing numpy using pip
    - Importing it in script to use it
3. The Numpy Arrays
    - 1D, 2D and 3D arrays
    - Concept of axes
    - Creating numpy arrays using `np.array()`
    - Creating numpy arrays with initial placeholder values using `np.zeros()`, `np.ones()`, `np.arange()`, `np.linspace()`, `np.full()` and `np.empty()`
4. I/O with Numpy Arrays
    - Why and when to use I/O functions?
    - `np.load()` and `np.save()` functions
    - `np.loadtxt()` and `np.savetxt()` functions
5. Inspecting your Numpy Array
    - `a.shape`, `a.ndim`, `a.size` and `a.dtype`

1. Introduction to Numpy

Cover the following points:

- What is numpy?
- How and why numpy provides fast operations on large arrays?
- Why it is superior to Python lists?
- Why should we use it?

2. How to use numpy?

- Install it first: `pip install numpy`
- Import it to use it: `import numpy as np`

3. The Numpy Arrays

Explain students that the soul of numpy is - numpy arrays. 1D Arrays are just a linear collection of items of same type. You can nest them to make 2D and 3D arrays.

- Example of 1D array: You have a list of temperatures recorded each day for a week: [25, 26, 24, 28, 27, 25, 23].

- Example of 2D array: You can think of them as having rows and columns, just like a spreadsheet. For instance, consider a list of students and their corresponding marks in three subjects: Math, English, and Science. You can represent this data in a 2D array, where each row represents a student and each column represents a subject.
- Example of 3D array: They can be visualized as a stack of 2D arrays. For example, imagine you have a collection of grayscale images, where each image is a 2D array representing pixel intensities. If you want to store multiple images, you can use a 3D array.

Explain the concept of axes. Mainly use the example of 2D data. Tell students, that `axis = 0` represents 'along columns' and `axis=1` represents 'along rows'.

Then move on to creation of arrays. Use `np.array()` to create the arrays.

```
### 1D Array Example
temperatures = [25, 26, 24, 28, 27, 25, 23]
temperatures_array = np.array(temperatures)
print(temperatures_array)

### 2D Array Example
marks = [
    [80, 75, 85],  # Marks of the first student [Math, English,
Science]
    [90, 88, 92],  # Marks of the second student [Math, English,
Science]
    [77, 80, 85],  # Marks of the third student [Math, English,
Science]
]
marks_array = np.array(marks)
print(marks_array)

### 3D Array Example
image1 = [
    [100, 120, 105],
    [115, 110, 90],
    [90, 95, 100]
]
image2 = [
    [80, 70, 85],
    [95, 100, 110],
    [105, 100, 95]
]
images = [image1, image2]
images_array = np.array(images)
print(images_array)
```

Touch the idea of array creation with initial placeholders. Introduce `np.zeros()`, `np.ones()`, `np.arange()`, `np.linspace()`, `np.full()` and `np.empty()`.

1. **np.zeros()**: This function creates an array filled with zeros.
   Example: Suppose you want to create an array of size 5, representing the number of seats available in a row in a movie theater. You can use np.zeros() to

create the array with all seats initially set to zero.

2. **np.ones():** This function creates an array filled with ones.
   Example: Let's say you want to create an array to represent the number of goals scored by a football team in five matches. You can use np.ones() to create the array with all elements initially set to one.

3. **np.arange():** This function creates an array with a sequence of numbers.
   Example: Imagine you need an array containing numbers from 0 to 9. You can utilize np.arange() to create the array with the desired sequence.

4. **np.linspace():** This function creates an array with evenly spaced values between a specified range.
   Example: Suppose you want to create an array of 6 values ranging from 0 to 1, representing intervals of time. You can use np.linspace() to achieve this.

5. **np.full():** This function creates an array with all elements set to a specific value.
   Example: Let's say you want to create an array representing the brightness levels of pixels in a grayscale image, where all pixels have an initial brightness value of 128. You can utilize np.full() to create the array with the desired value.

6. **np.empty():** This function creates an array without initializing its elements to any particular value. The values in the array can be arbitrary and vary each time the function is called.
   Example: Suppose you need an array of size 4, representing the scores of participants in a competition. You can use np.empty() to create the array without initializing the values.

```
### Examples

seats = np.zeros(5)
print(seats)

goals = np.ones(5)
print(goals)

numbers = np.arange(10)
print(numbers)

time_intervals = np.linspace(0, 1, 6)
print(time_intervals)

brightness_levels = np.full((3, 3), 128)
print(brightness_levels)

scores = np.empty(4)
print(scores)
```

4. I/O with Numpy Arrays

Explain students that NumPy provides functions like np.load(), np.save(), np.loadtxt(), and np.savetxt() to facilitate input/output operations with external files.

```
student_names = np.array(['Alice', 'Bob', 'Charlie', 'David'])
np.save('student_names.npy', student_names)

loaded_names = np.load('student_names.npy')
print(loaded_names)
```

In this example, we use np.save() to save the student_names array to a file. Later, we use np.load() to load the saved array from the file and assign it to the loaded_names array.

Suppose you have a text file named 'data.txt' containing comma-separated values (CSV) representing student scores:

```
    Alice,95
    Bob,87
    Charlie,92
    David,88
```

```
data = np.loadtxt('data.txt', delimiter=',', dtype=str)
print(data)

scores = np.array([95, 87, 92, 88])
np.savetxt('scores.txt', scores, delimiter=',')
```

In this example, we use np.loadtxt() to load the data from the text file 'data.txt'. We specify the delimiter as , to indicate that the values are comma-separated. The resulting NumPy array, data, contains the loaded data.

We then save the scores array to a text file named 'scores.txt', with values separated by commas.

5. Inspecting your Numpy Array

Introduce `a.shape`, `a.ndim`, `a.size` and `a.dtype`.

```
a = np.array([[1, 2, 3], [4, 5, 6]])
print(a.shape)
print(a.ndim)
print(a.size)
print(a.dtype)
```