Discussion-3 How an array could be used for storing information

Discussion Topic:

Identify three real-life scenarios in which an array could be used for storing information. Provide a sample code segment that illustrates how to store data in an array for one of your outlined scenarios. Provide a rationale for your response.

My Post:

Hello Class,

Arrays are used to store collections of data. In Python, they can store objects of the same type, use indexes, are iterable, and are mutable. Thus, they are versatile and commonly used in programming. In real-life scenarios, they are often used to store, iterate, and manipulate data. Below are three real-life examples. Note that arrays are called lists in Python.

1. They are often used to store non-persistent data, such as a player's state (e.g., mana and health levels) during a single video game session or metrics from a sensor that are processed in real-time before being discarded.
2. They are also used to store a small list of items in a shopping cart. Arrays are index-searchable data structures that store elements in sequence and are often used to implement abstract data types, such as Abstract Data Types (ADTs). In computer science, ADTs are data models that define sets of data operations, such as append(), remove(), and add(). Thus, they are perfect for dynamically handling the addition, removal, and retrieval of items in a small collection like a shopping cart.
3. The most common usage of arrays in Python, especially when using its NumPy library, is to store numeric data to perform statistical calculations on the stored numeric values, like finding the average, the highest score, or the lowest score of student exam results for a class.

To implement the class score example in Python, a NumPy array is the ideal choice, as it stores numeric data, is optimized for fast numerical computation, is also optimized for efficient memory management, and comes with built-in statistical functionality:

- The NumPy library backend was built with C, and its name indicates that it handles numeric data types, making the NumPy arrays faster for mathematical and statistical operations than regular/standard Python arrays.
- For large datasets of numbers, NumPy arrays need less memory than Python arrays as they store data in a more compact way, and their backend is built in C. This makes them more efficient for handling computation on large datasets.
- The NumPy library includes numerous optimized and pre-built functions for statistical analysis (e.g., .mean(), .max(), .min(), .std()) for standard deviation, etc.). These abstractions of functionality allow programmers to implement data analysis with simple commands and very little code.

To refine the class score example, let's consider a high school needing to perform a detailed statistical analysis on the ACT scores of thousands of its students. NumPy arrays will be perfect for this example.

Implementation example:

```
import numpy as np

# a standard Python array of students' ACT scores that simulates a database with
thousands of scores
class_scores = [
    88, 92, 75, 95, 68, 85, 78, 99, 72, 89,
    76, 81, 93, 84, 77, 65, 88, 91, 70, 80,
    94, 79, 82, 87, 96, 71, 69, 90, 86, 74
]

# convert the standard array (list in Python) into a NumPy array
scores_array = np.array(class_scores)
num_students = len(scores_array)

print(f"\n--- Class of {num_students} Students ---")

# calculations.
mean_score = np.mean(scores_array)
median_score = np.median(scores_array)
std_dev = np.std(scores_array)
highest_score = np.max(scores_array)
lowest_score = np.min(scores_array)

# how many students scored 90 or higher
high_achievers = scores_array[scores_array >= 90]
num_high_achievers = high_achievers.size

# Display
print(f"Average Score: {mean_score:.2f}")
print(f"Median Score: {median_score}")
print(f"Standard Deviation: {std_dev:.2f}")
print(f"Highest Score: {highest_score}")
print(f"Lowest Score: {lowest_score}")
print(f"Number of Students with Scores >= 90: {num_high_achievers}\n")
```

Outputs:

```
--- Class of 30 Students ---
Average Score: 82.47
Median Score: 83.0
Standard Deviation: 9.23
Highest Score: 99
Lowest Score: 65
```

```
Number of Students with Scores >= 90: 8
```

-Alex