Reliance Foundation

N·S·D·C RE IMAGINE FUTURE

Python Arrays

What is a Python Array?

- An array is a collection of elements of the same data type.
- Each element in an array has an index, which is its position in the array.
- Arrays are stored in contiguous memory locations, which makes them efficient for accessing and manipulating data.

There are three ways to create arrays in Python

- Lists
- Tuples
- Arrays provided by Array Module

Types of Arrays

One-dimensional arrays **Two-dimensional** **Multi-dimensional arrays**

Numpy Arrays

One-dimensional arrays, often referred to as vectors, are arrays with a single dimension. They are essentially lists of elements arranged in a linear sequence.

```
one_dimensional_array = [1, 2, 3, 4, 5]
```

Two-dimensional arrays, often referred to as matrices, have two dimensions - rows and columns. They are used to represent tables of data.

```
two_dimensional_array = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9] ]
```

Multi-dimensional Arrays

The term "multi-dimensional arrays" generally refers to arrays with more than two dimensions. In Python, this can be achieved using nested lists or with libraries like NumPy, which supports arrays of any number of dimensions.

```
three_dimensional_array = [
    [
        [1, 2, 3],
        [4, 5, 6],
        [7, 8, 9]
    ],
    [
        [10, 11, 12],
        [13, 14, 15],
        [16, 17, 18]
    ]
]
```

Creating Arrays Using Lists

Arrays can be made using lists. A list is a versatile and commonly used data structure in Python. It is a mutable, ordered sequence that can contain elements of different data types.

Creating a list

```
my_list = [1, 2, 3, 4, 5]
```

Accessing Elements: You can access elements in a list using indexing:

```
first_element = my_list[0]  # Access the first element (index 0)
```

Modifying Elements: Lists are mutable, so you can modify elements:

```
my_list[1] = 10  # Change the second element
```

Adding or Removing Elements from a List

To add or remove elements from a list in Python, you can use various methods and operations. Here are some common ways to achieve this:

Adding Elements:

- append(): Adds an element to the end of the list.
- insert(): Inserts an element at a specified position.

Removing Elements:

- remove(): Removes the first occurrence of a specified value.
- pop(): Removes an element at a specified index (and returns it).
- del: Removes an element at a specified index or a slice of elements.
- clear(): Removes all elements from the list.

Adding or Removing Element

```
my_list.append(6)  # Add an element to the end
my_list.insert(2, 7)  # Insert 7 at index 2
my_list.remove(3)  # Remove the first occurrence of 3
```

Creating Arrays using Tuples

Tuples (also referred as arrays) are similar to lists but are immutable, meaning their elements cannot be changed after creation.

Creating a Tuple:

```
my_tuple = (1, 2, 3, 4, 5)
```

Accessing Elements: You can access elements in a tuple using indexing:

```
first_element = my_tuple[0]  # Access the first element (index 0)
```

Since tuples are immutable, you cannot modify their elements once they are created.

Creating Arrays (from the array module)

The array module provides a fixed-size array that is more efficient than lists when dealing with large datasets of the same data type.

Creating an Array:

```
from array import array

my_array = array('i', [1, 2, 3, 4, 5])
# 'i' denotes the data type (int), and the list contains initial values.
```

The code creates an array named my_array using the 'array' module in Python, specifying the data type 'i' for integers, and initializing it with the values 1, 2, 3, 4, and 5.

Accessing and Modifying Elements of an Array

Accessing Elements:

```
first_element = my_array[0]  # Access the first element)
```

The code assigns the variable first_element the value of the first element in the array my_array by using the index [0]. In Python, array indices start at 0, so my_array[0] retrieves the value at the first position in the array.

Modifying Elements:

```
my_array[1] = 10  # Change the second element
```

The second code modifies the second element in the array my_array by assigning the value 10 to the element at index 1. In Python, array indices start at 0, so my_array[1] refers to the second element in the array, and setting it equal to 10 changes the original value at that position.