**1. What is a Python library? Why do we use Python libraries?**

**Answer:- 1 What is a Python Library?**

A **Python library** is a collection of pre-written code that you can import into your Python programs to reuse functionality, simplifying the development process. Libraries can include functions, classes, and methods designed to perform specific tasks such as data manipulation, web development, machine learning, and more.

**2. Why Do We Use Python Libraries?**

We use Python libraries for several reasons:

1. **Code Reusability**: Libraries save time and effort by providing ready-made functions and modules, so you don't need to reinvent the wheel for common tasks.
2. **Efficiency**: Libraries are optimized for performance, helping to implement complex functionality with minimal effort.
3. **Focus on Core Logic**: By using libraries, you can focus on writing the core logic of your application while relying on external libraries for supporting functionality.
4. **Community Support**: Popular libraries have extensive community support, frequent updates, and documentation, making development easier and faster.

**2. What is the difference between Numpy array and List?**

**Answer**:- Difference Between NumPy Array and List

| **Feature** | **NumPy Array** | **Python List** |
| --- | --- | --- |
| **Data Type** | **Homogeneous (all elements of the same type).** | **Heterogeneous (can contain elements of different types).** |
| **Memory Efficiency** | **More memory-efficient due to fixed-size elements.** | **Less memory-efficient, as elements are dynamically typed.** |
| **Performance** | **Faster for numerical and array-based operations.** | **Slower for such operations due to the dynamic typing and overhead.** |
| **Functionality** | **Supports advanced mathematical, statistical, and vectorized operations.** | **Does not support vectorized operations natively.** |
| **Size Flexibility** | **Fixed size after creation. Resizing requires a new array.** | **Lists can dynamically grow or shrink in size.** |
| **Operations** | **Allows element-wise operations and broadcasting.** | **Does not support element-wise operations directly.** |
| **Methods** | **Comes with built-in methods for numerical computations, such as mean(), sum().** | **Lacks built-in mathematical functions. You need to manually loop through elements.** |
| **Multidimensional** | **Supports multidimensional arrays (e.g., 2D, 3D).** | **Lists require nested structures for multidimensional arrays.** |

**3. Find the shape, size and dimension of the following array? [[1, 2, 3, 4] [5, 6, 7, 8], [9, 10, 11, 12]]**

**Answer:- The array has:**

* **Shape: (3, 4) (3 rows and 4 columns)**
* **Size: 12 (total number of elements)**
* **Dimension: 2 (it's a 2-dimensional array)**

import numpy as np

# Define the array

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

[5, 6, 7, 8],

[9, 10, 11, 12]])

# Get the shape, size, and dimension

array\_shape = array.shape

array\_size = array.size

array\_dimension = array.ndim

array\_shape, array\_size, array\_dimension

1. **Write python co^e to access the first row of the following array? [[1, 2, 3, 4] [5, 6, 7, 8], [9, 10, 11, 12]]**

**Answer:-**

import numpy as np

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

[5, 6, 7, 8],

[9, 10, 11, 12]])

first\_row = arr[0]

print(first\_row)

1. **How do you access the element at the third row and fourth column from the given numpy array? [[1, 2, 3, 4] [5, 6, 7, 8], [9, 10, 11, 12]]**

**Answer:-**

import numpy as np

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

[5, 6, 7, 8],

[9, 10, 11, 12]])

element = arr[2, 3]

print(element)

1. **How can you generate a random 3x3 matrix with values between 0 and 1?**

**Answer:-**

import numpy as np

# Generate a 3x3 matrix with random values between 0 and 1

random\_matrix = np.random.rand(3, 3)

print(random\_matrix)

1. **Describe the difference between np.random.rand anddnp.random.randn?**

**Answer:-**

**np.random.rand():**

* Generates random numbers from a uniform distribution over the range [0, 1).
* The values are always between 0 and 1.

**np.random.randn():**

* Generates random numbers from a standard normal distribution (mean = 0, standard deviation = 1).
* The values can be both positive and negative, centered around 0.

1. **Write code to increase the dimension of the following array? [[1, 2, 3, 4] [5, 6, 7, 8], [9, 10, 11, 12]]**

**Answer:-**

You can increase the dimension of the given array using np.expand\_dims() or by reshaping it.

Here's the code to add an extra dimension:

**Using np.expand\_dims():**

**import numpy as np**

**arr = np.array([[1, 2, 3, 4],**

**[5, 6, 7, 8],**

**[9, 10, 11, 12]])**

**# Adding an extra dimension (making it 3D)**

**new\_arr = np.expand\_dims(arr, axis=0) # Adds a new dimension at the front**

**print(new\_arr)**

**print(new\_arr.shape)**

Using reshape():

**# Reshaping to add an extra dimension**

**reshaped\_arr = arr.reshape(1, 3, 4) # Reshape to 3D (1x3x4)**

**print(reshaped\_arr)**

**print(reshaped\_arr.shape)**

1. **How to transpose the following array in NumPy? [[1, 2, 3, 4] [5, 6, 7, 8], [9, 10, 11, 12]]**

**Answer:-**

import numpy as np

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

[5, 6, 7, 8],

[9, 10, 11, 12]])

# Using transpose() function

transposed\_arr = np.transpose(arr)

print(transposed\_arr)

# Alternatively, using .T attribute

transposed\_arr\_T = arr.T

print(transposed\_arr\_T)