**Slicing, Array and lists**

**Slicing:**

powerful technique used to extract a portion of a sequence, such as a string or a list

Example

* **reverse a word using slicing, by using negative step in slicing.**

word = "hello"

reversed\_word = word[::-1]

print(reversed\_word)

# Output: "olleh"

**Array vs list:  
  
1. Type of Elements:**

**Lists:** Lists can contain elements of different types.

**Arrays:** Arrays in Python can only contain elements of the same type.

**2. Memory Efficiency:**

**Lists:** Lists are less memory efficient compared to arrays because they store additional information such as the type of each element.

**Arrays:** Arrays are more memory efficient as they store only the elements without additional metadata.

**3. Functionality and Versatility:**

**Lists:** Lists offer a wide range of built-in methods and are highly versatile for general-purpose programming tasks.

**Arrays:** Arrays provide limited functionality compared to lists, primarily focused on numerical computations.

**4. Performance:**

**Lists:** Lists might offer slightly slower performance compared to arrays, especially for numerical computations on large datasets.

**Arrays:** Arrays can provide better performance for numerical operations due to their memory efficiency.

**5. Use Cases:**

**Lists:** Lists are suitable for a wide range of tasks where flexibility and versatility are required, such as managing heterogeneous data and implementing various data structures.

**Arrays**: Arrays are ideal for scenarios involving large datasets of homogeneous numeric data, such as scientific computing, where memory efficiency and performance are crucial.

**6. Mutability:**

**Lists:** Lists are mutable, allowing elements to be modified, added, or removed after creation.

**Arrays:** Arrays are also mutable, enabling modification of elements, but they are primarily used for numerical computations where modification is often performed through mathematical operations rather than direct manipulation.