1. **What is the difference between Single quoted string and double quoted string in python?**

**Ans:**

| Single quoted | Double quoted |
|---|---|
| Single quote is denoted by ‘ ’ | Double quote is denoted by “ “ |
| Single quotes (' ') are convenient when your string contains double quotes: 'She said, "Hello"'. | Double quotes (" ") are convenient when your string contains single quotes: "It's a sunny day". |
| Single quote is used for single line strings. | double quote is used for single line strings. |
| It is define as a string. | Double quote is also define as a string. |

2. **What is the difference between immutable and mutable objects?**

**Ans:**

- **Definition:**
  - Immutable objects: Once created, the object's state (its data) cannot be changed. Examples include integers, strings, and tuples.
  - Mutable objects: The object's state can be modified after it is created. Examples include lists, dictionaries, and sets.
- **Modification:**
  - Immutable: If you try to change the value of an immutable object, a new object is created with the modified value, while the original object remains unchanged.
  - Mutable: The same object can be altered directly without creating a new object.
- **Memory Efficiency:**
  - Immutable: Since immutable objects do not change, they can be stored more efficiently and even shared across different parts of the program**.** For example, two variables referencing the same string actually point to the same memory location.
  - Mutable: Mutability allows in-place modification, but this can sometimes lead to inefficiency if many modifications are performed.

3. **What is the difference between list and tuple in python?**

**Ans: Mutability:**

- List: Mutable, meaning you can modify, add, or remove elements after the list is created.
- Tuple: Immutable, meaning once a tuple is created, you cannot change, add, or remove elements from it.

**Syntax:**

- List: Defined using square brackets: my_list = [1, 2, 3]
- Tuple: Defined using parentheses: my_tuple = (1, 2, 3)

**Performance:**

- List: Since lists are mutable, they are generally slower than tuples when performing operations like iteration. This is because Python has to allocate extra memory to accommodate potential changes.
- Tuple: Being immutable, tuples are generally faster and more memory-efficient than lists, especially when iterating over large datasets.

**Use Cases:**

- List: Used when you need a collection that can be modified, e.g., when you want to add or remove items dynamically.
- Tuple: Used when you need a fixed collection of items, such as coordinates or records that should not change throughout the program.

4. **What are the difference between a set and list in terms of Functionality and use cases?**

**Ans**: Duplicates:

- List: Allows duplicate elements. You can have multiple occurrences of the same value.
    - Example: [1, 2, 2, 3] is a valid list.
- Set: Does not allow duplicates. If you try to add a duplicate value, it will be ignored.
    - Example: {1, 2, 2, 3} becomes {1, 2, 3}.

**Order:**

- List: Maintains the order of elements as they are inserted. Elements can be accessed by their index.

    - Example: [1, 2, 3] will always maintain this order.
- Set: Does not maintain any order. The elements are stored in an unpredictable order, and you cannot access elements by index.
    - Example: {1, 2, 3} may internally be stored as {3, 1, 2}, but ordering doesn't matter.

**Speed:**

- List: Slower for lookups (in checks) because it has to search through the entire list.
    - Example: Checking membership in a large list can be slow.
- Set: Faster for lookups (in checks) because sets are implemented as hash tables, providing average O(1) time complexity for membership tests.
    - Example: Checking if an element exists in a large set is faster than in a list.

**Operations:**

- List: Has more functionality related to managing ordered data, such as slicing, indexing, and allowing duplicate elements.
    - Methods: append(), remove(), sort(), pop(), etc.
- Set: Provides functionality for set operations like union, intersection, difference, and symmetric difference.
    - Methods: union(), intersection(), difference(), add(), etc.

**5. How does a dictionary differ from a list in term of data storage and retrieval?**

**Ans**: A list is an ordered collection of items, whereas a dictionary is an unordered data collection in a key: value pair. Elements from the list can be accessed using the index, while the elements of the dictionary can be accessed using keys.