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

Ans: **Single-quoted strings (')** and **double-quoted strings (")** in Python are essentially the same. Both are used to represent string literals, and there is no functional difference between them.

The choice between single or double quotes is often a matter of personal preference or convenience. For example:

single_quoted = 'Hello, World!'

double_quoted = "Hello, World!"

- If your string contains an apostrophe , using double quotes for the string can avoid the need for escaping the apostrophe:

quote_example = "Don't worry!"

- Similarly, if the string contains double quotes, you can use single quotes:

quote_example = 'He said "Hello"'

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

Ans: **Mutable Objects**: These are objects whose state or value can be changed after they are created. Examples include lists, dictionaries, and sets.

my_list = [1, 2, 3]

my_list[0] = 99   # The list is mutable(you can change its contents)

Immutable Objects: These are objects whose state or value cannot be changed after creation. Examples include strings, tuples, and integers. Once you assign a value to an immutable object, you can't modify it.

my_string = "hello"

You cannot modify the string directly.

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

Ans : **Lists**:

- Lists are **mutable**. This means you can modify, add, or remove elements after the list is created.
- Lists are defined using square brackets [].
- Lists are typically used when you need a collection of items that can be changed.

Example:

my_list = [1, 2, 3]

my_list[0] = 10 # Lists can be modified

Tuples:

- Tuples are **immutable**. Once created, you cannot modify their elements.
- Tuples are defined using parentheses ().
- Tuples are used when you need to ensure that the data is not accidentally changed.

Example:

my_tuple = (1, 2, 3)

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

Ans: Lists:

- Lists are **ordered** collections of items. Items in a list have a specific order, and this order is preserved.
- Lists can contain **duplicate elements**.
- Lists allow **indexing**, meaning you can access elements by position.
- **Use cases**: Lists are good when the order of elements matters or when you need to modify the collection (add, remove, change items).

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

print(my_list[0])                    # Access by index
```

- **Sets :**

- Sets are **unordered** collections of items. The order of elements is not guaranteed.
- Sets do **not allow duplicate elements**. Any repeated item will be removed automatically.
- Sets are typically used for operations that involve checking membership, removing duplicates, or performing mathematical operations (like union or intersection.
- **Use cases**: Sets are ideal when you need to check membership quickly, ensure uniqueness, or perform set operations like unions and intersections.

```
my_set = {1, 2, 3, 1}                # The set will automatically remove duplicates

print(my_set)
```

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

Ans : List:

- A list is an ordered collection of elements indexed by integers (0, 1, 2, …).
- Elements in a list are stored sequentially.
- Data is accessed by indexing.
- Lists are ideal for storing collections of items where order matters, and when you need to access elements based on their position.

```
my_list = [10, 20, 30]

print(my_list[1])                    # Access by index
```

Dictionary:

- A dictionary is an unordered collection of key-value pairs.
- Keys in a dictionary are unique, and values are accessed via their keys.
- Dictionaries are ideal for associative mappings, where each key is associated with a value

```
my_dict = {"name": "Alice", "age": 30}

print(my_dict["name"])               # Access by key
```