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

1. **Use** :
    - Use single quotes (') when the string contains double quotes.
    - Use double quotes (") when the string contains single quotes.

Examples:

single_quoted = 'She said, "Hello!"'

double_quoted = "It's a beautiful day."

2. **Escaping Quotes**: If your string contains both single and double quotes, you'll need to escape one type of quote using a backslash (\).

Example:

mixed_quotes = 'It\'s "Python" programming!'

mixed_quotes2 = "It's \"Python\" programming!"

3. **Docstrings**: Triple-quoted strings (''' or """) are often used for multi-line strings or docstrings in Python. While either can be used, the convention is to use triple double quotes (""") for docstrings.

Example:

"""This is a docstring.

It can span multiple lines."""

What is the difference between immutable and mutable objects?

Mutable Objects

- **Definition**: Objects whose values or state can be changed after creation.
- **Examples**:
  - Lists (list)
  - Dictionaries (dict)
  - Sets (set)
  - Custom objects (depending on implementation)
- **Behavior**:
  - Modifications, such as adding, removing, or changing elements, affect the original object.
  - Multiple variables referencing the same mutable object will reflect changes made through any reference.

Example:

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

another_list = my_list

my_list.append(4)


print(my_list)      # Output: [1, 2, 3, 4]

print(another_list)  # Output: [1, 2, 3, 4] (same object modified)
```

Immutable Objects

- **Definition**: Objects whose values or state cannot be changed after creation.
- **Examples**:
  - Strings (str)
  - Tuples (tuple)
  - Numbers (int, float, complex)
  - Frozensets (frozenset)
- **Behavior**:
  - Any "modification" creates a new object, leaving the original unchanged.
  - Variables referencing the same immutable object do not affect each other when reassigned.

Example:

```
my_string = "hello"
another_string = my_string
my_string += " world"

print(my_string)      # Output: "hello world" (new object created)
print(another_string)  # Output: "hello" (original object unchanged)
```

What is the difference between list and tuple in python?

ASPECT	LIST	TUPLE
MUTABILITY	Mutable: Items can be added, removed, or changed.	Immutable: Items cannot be modified after creation.
SYNTAX	Defined using square brackets: [ ].	Defined using parentheses: ( ).
PERFORMANCE	Slower: Overhead for mutability.	Faster: Optimized for immutability.
USE CASE	Best for dynamic collections that change over time.	Best for static data or read-only collections.
HASHABLITY	Not hashable: Cannot be used as keys in dictionaries.	Hashable (if containing hashable elements): Can be used as dictionary keys or set elements.
MEMMORY USAGE	Consumes more memory due to extra functionality for mutability.	Consumes less memory than lists.

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

ASPECT	SET	LIST
MUTABILITY	Mutable (can add/remove elements)	Mutable (can add/remove elements)
ORDERIING	Unordered (no guaranteed order of elements).	Ordered (preserves the order of elements).
DUPLICATES	Does not allow duplicate elements.	Allows duplicate elements.
ACCESS	No indexing or slicing; elements must be accessed via iteration.	Supports indexing and slicing.
DATA STRUCTURE	Based on a hash table for fast membership testing.	Sequential structure for ordered data.
MEMBERSHIP TESTING	Fast (O(1) for in checks).	Slow (O(n) for in checks).
PERFORMANCE	Faster for operations like union, intersection, and difference.	Optimized for ordered data manipulation.
HASHABILITY	Elements must be hashable (e.g., int, str, tuple).	No such restriction; can include any object.

Use Cases

Set

- When you need to store **unique items** (e.g., tracking unique visitors on a website).
- For **fast membership testing** (e.g., checking if an element exists in a large collection).
- When performing **set-based operations** like union, intersection, or difference.

List

- When you need an **ordered collection** of items.
- If **duplicates are allowed or required**.
- For operations requiring **indexing, slicing, or positional manipulation**.
- When working with **sequential data** like a queue or stack.

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

ASPECT	DICTIONARY	LIST
STRUCTURE	Stores data as key-value pairs.	Stores data as an ordered sequence of elements.
ACCESS METHOD	Retrieve values using unique keys.	Retrieve elements using integer indices.
MUTABILITY	Mutable (keys and values can be added, removed, or updated).	Mutable (elements can be added, removed, or updated).
UNIQUENESS	Keys must be unique (values can be duplicated).	No uniqueness constraint on elements.
ORDER	Maintains insertion order (since Python 3.7+).	Maintains insertion order.
PERFORMANCE	Fast (O(1)) for key-based lookups.	Slow (O(n)) for lookups if searching by value.
MEMORY USAGE	Higher memory overhead due to hash table implementation.	Lower memory usage compared to dictionaries.
USE CASE	Ideal for data that needs to be associated with unique identifiers (keys).	Ideal for ordered collections where elements are accessed by position.