1 mark MCQ questions with answers :

1)Which data type does not support indexing or slicing?

A) String

B) List

C) Set

D) Tuple

Correct Answer: C) Set

2)Which of the following operations can be performed on a range object in Python?

A) Append

B) Modify elements

C) Indexing

D) Remove elements

Correct Answer: C) Indexing

5 marks questions with answers :

1)Explain the difference between mutable and immutable data types in Python. Provide examples of both, and discuss the implications of mutability with respect to operations like appending, modifying, and reassigning elements.

Answer:

In Python, mutable data types are those that can be modified after their creation. These data types allow changes to be made to their content or structure. Immutable data types, on the other hand, cannot be altered after their creation.

Mutable Data Types: Lists, Sets, Dictionaries

Examples:

*List: my_list = [1, 2, 3]

*Set: my_set = {1, 2, 3}

*Dictionary: my_dict = {'a': 1, 'b': 2}

Operations: You can change the contents of mutable types. For instance, you can append to a list (my_list.append(4)), add or remove items from a set (my_set.add(4)), or modify key-value pairs in a dictionary (my_dict['a'] = 10).

Immutable Data Types: Tuples, Strings, Integers, Floats

Examples:

*Tuple: my_tuple = (1, 2, 3)

*String: my_string = "hello"

*Integer: x = 5

Operations: You cannot modify an element in an immutable data type directly. For example, trying to change an element of a tuple like my_tuple[0] = 10 will result in a TypeError.

Mutability allows direct modification, which is efficient when managing data that needs frequent updates. Immutability ensures data integrity. Since the object cannot change, it can be safely shared across different parts of a program without risk of unintended changes**.**

2) You are given a list of strings: fruits = ['apple', 'banana', 'cherry', 'date']

Explain how you would: Append a new string "elderberry" to the list. Remove the string "banana" from the list. Sort the list in alphabetical order. Reverse the list and print the updated list

Answer:

Given the list fruits = ['apple', 'banana', 'cherry', 'date'], here's how you would perform the required operations:

*Append a new string "elderberry" to the list:

We can use the append() method to add an item at the end of the list.

fruits.append('elderberry')

*Remove the string "banana" from the list:

We can use the remove() method to delete a specific item from the list.

fruits.remove('banana')

*Sort the list in alphabetical order:

We can use the sort() method to arrange the elements in ascending order.

fruits.sort()

*Reverse the list and print the updated list:

We can use the reverse() method to reverse the order of elements.

Code implementation:

```
fruits.reverse()
print(fruits)
fruits = ['apple', 'banana', 'cherry', 'date']
fruits.append('elderberry')  # Append 'elderberry'
fruits.remove('banana')  # Remove 'banana'
fruits.sort()  # Sort alphabetically
fruits.reverse()  # Reverse the list
print(fruits)
```

Output:

```
['elderberry', 'date', 'cherry', 'apple']
```