**1) . What is the difference between enclosing a list comprehension in square brackets and parentheses?**

Ans-Enclosing a list comprehension in square brackets creates a list, while enclosing it in parentheses creates a generator expression. A list comprehension creates a new list by evaluating an expression for each item in an iterable and collecting the results, whereas a generator expression creates an iterator that yields each item in the iterable as it is needed. List comprehensions are generally faster than generator expressions for small or medium-sized collections, but generator expressions are more memory-efficient for large collections or infinite sequences.

**2) What is the relationship between generators and iterators?**

Ans-Generators and iterators are closely related concepts in Python. An iterator is an object that provides a sequence of values, one at a time, using the \_\_next\_\_() method. A generator is a type of iterator that is defined using a function that contains one or more yield statements. When a generator function is called, it returns a generator object that can be used to iterate over the values generated by the function. Unlike regular iterators, which must generate all the values in advance, generators generate values on demand, making them more memory-efficient and suitable for working with large or infinite sequences.

**3) What are the signs that a function is a generator function?**

Ans-The most obvious sign that a function is a generator function is the presence of one or more yield statements. A function that contains a yield statement is a generator function, even if it also contains other statements or returns a value using the return statement. Additionally, generator functions are often designed to be used with the for loop or other iterator-consuming constructs, rather than being called directly.

**4) What is the purpose of a yield statement?**

Ans-The purpose of a yield statement in a generator function is to generate a value and pause the function's execution, allowing the caller to consume the value before the function resumes. When a generator function encounters a yield statement, it returns the value to the caller and "remembers" its state so that it can resume where it left off when it is called again. This allows generator functions to generate sequences of values without generating them all at once, making them more memory-efficient and suitable for working with large or infinite sequences.

**5) What is the relationship between map calls and list comprehensions? Make a comparison and contrast between the two.**

Ans-Both map calls and list comprehensions are used to transform and process elements from one iterable to another. They have similarities and differences:

Similarities:

Both map calls and list comprehensions can apply a transformation function to each element of an iterable and generate a new iterable containing the transformed elements.They allow for concise and expressive code to perform element-wise operations.

Differences:

Map calls use the map() function, which takes a function and an iterable as arguments, whereas list comprehensions use a compact syntax within square brackets or parentheses. Map calls return a map object, which is an iterator, while list comprehensions return a new list. List comprehensions can include conditional