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

parentheses?

Ans:

Square Brackets []: When you enclose a list comprehension in square brackets, you create a new list. The comprehension iterates over an iterable (like a list, tuple, range, etc.), performs an operation on each item, and collects the results in a new list.

Parentheses (): When we enclose a list comprehension in parentheses, we are creating a generator expression. A generator expression is similar to a list comprehension, but it doesn't construct the entire list in memory right away. Instead, it generates each element one at a time as you iterate over it. This can be more memory-efficient for large datasets.

2) What is the relationship between generators and iterators?

The relationship between generators and iterators is that generators are a specific type of iterator. All generators are iterators, but not all iterators are generators.

A generator is a more concise and convenient way to create iterators. It allows you to define an iterator by writing a single function with a special keyword yield. When the function is called, it doesn't execute the entire function body immediately but instead yields values one at a time each time the yield statement is encountered. This makes generators memory-efficient and particularly useful for large datasets.

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

Use of the yield Keyword

Function Contains a Loop

No Return Statements

4) What is the purpose of a yield statement?

The yield statement in Python is used within a generator function to produce a value and temporarily pause the function's execution, while maintaining its internal state. It serves a specific purpose:

Generating Values On-Demand: The primary purpose of the yield statement is to generate values one at a time as requested by the caller. Instead of calculating and storing all values upfront, which could consume a lot of memory, a generator function generates values on-demand, making it memory-efficient for large or infinite sequences of data.

Preserving Function State: When a generator function encounters a yield statement, it saves its current state, including variable values and the position of the code execution. This allows the function to be resumed from the exact point it left off the next time it's called. This is in contrast to normal functions, which start executing from the beginning every time they're called.

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

contrast between the two.

A list comprehension is a more compact and readable way to create a new list by applying an expression to each item in an iterable. It directly generates a list, eliminating the need for an explicit conversion.

The map() function applies a given function to all items in an input iterable and returns a map object, which is an iterator. You typically need to convert the map object to a list or another iterable to work with the results

def num (n) :

return n \* 2

lst = [2, 44, 5.5, 6, -7]

x = map(num, lst)

print(x)

# returns list

print(list(x))