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

Ans - Square brackets: When a list comprehension is enclosed in square brackets, it creates and returns a new list containing the generated elements.

For example:

squares = [x\*\*2 for x in range(5)]

print(squares) # Output: [0, 1, 4, 9, 16]

In this case, the list comprehension [x\*\*2 for x in range(5)] generates a list of squares of numbers from 0 to 4 and assigns it to the squares variable.

Parentheses: When a list comprehension is enclosed in parentheses, it creates and returns a generator object. A generator is an iterable that generates values on-the-fly as they are needed, rather than creating the entire sequence in memory upfront

for example

squares\_generator = (x\*\*2 for x in range(5))

print(squares\_generator) # Output: <generator object <genexpr> at 0x00000123456789>

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

Ans - **Iterators**: Iterators are objects that implement the iterator protocol, which consists of the \_\_iter\_\_() and \_\_next\_\_() methods. Iterators represent a sequence of values and allow iteration over those values one at a time. The \_\_iter\_\_() method returns the iterator object itself, and the \_\_next\_\_() method returns the next value in the sequence or raises the exception when there are no more values.

**Generators:** Generators are a specific type of iterator that can be created using a special kind of function called a generator function. Generator functions are defined using the yield keyword instead of the return keyword. When a generator function is called, it returns a generator object, which is both an iterator and an iterable.

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

Ans – presence of yield statement is a sign that a function is generator function.

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

Ans – It returns a generator object to the one who calss the function which contains yield instead of simply returning a value.

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

The relationship between map calls and list comprehensions is that they both provide ways to perform transformations on iterables. Map calls apply a function to each element of an iterable, while list comprehensions allow for more complex transformations and filtering using a concise syntax. List comprehensions are more expressive but may be less efficient for large datasets compared to map calls.