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

Enclosing a list comprehension in square brackets creates a list, while enclosing it in parentheses creates a generator expression. A list comprehension evaluates immediately, generating the entire list in memory. In contrast, a generator expression generates elements on the fly as needed, conserving memory. Generators are iterable and can be used in loops, but their elements are not stored in memory all at once like a list.

2) What is the relationship between generators and iterators?

Generators are a type of iterator in Python. They are defined using the `yield` keyword and generate values on the fly when iterated upon. Generators provide a convenient way to create iterators without the need to define a class explicitly. They allow for lazy evaluation, generating values one at a time, conserving memory and enabling efficient processing of large or infinite sequences of data.

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

A function is recognized as a generator function by the presence of the `yield` keyword within its body. When a generator function is called, it doesn't execute the code immediately; instead, it returns a generator object. The generator object can be iterated upon using the `next()` function or in a `for` loop, allowing for the generation of values on demand and pausing and resuming execution using the `yield` statement.

4) What is the purpose of a yield statement?

The `yield` statement in Python is used within a generator function to define a point of suspension and value emission. When the `yield` statement is encountered, the generator function temporarily halts its execution, preserving its internal state. The value provided after the `yield` keyword is emitted as the next value in the sequence when the generator is iterated upon, allowing for lazy evaluation and efficient memory usage.

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

Both map calls and list comprehensions in Python are used to perform operations on iterable objects and generate new sequences.

Map applies a given function to each element of an iterable and returns an iterator of the results. It can be used with any iterable object and is particularly useful when a simple transformation is needed.

List comprehensions, on the other hand, allow for more complex transformations by applying expressions to elements of an iterable and generating a new list. They offer more flexibility and can include conditions and multiple expressions.

Overall, list comprehensions are more expressive and versatile, while map calls are simpler and more concise for basic transformations.