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. The key difference is that a list comprehension generates the entire list in memory, whereas a generator expression generates values on-the-fly as they are needed, saving memory.

2. What is the relationship between generators and iterators?

Generators are a type of iterator in Python. An iterator is an object that implements the methods \_\_iter\_\_() and \_\_next\_\_(). A generator is a special type of iterator that is defined using a function with the yield keyword. Generators simplify the process of creating iterators by allowing you to use functions and automatically maintaining the state between function calls.

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

A function is a generator function if it contains the yield keyword. When a generator function is called, it doesn't execute the function body immediately; instead, it returns a generator object that can be used to control the execution and retrieve values using the next() function.

4. What is the purpose of a yield statement?

The yield statement is used in generator functions to produce a value that will be returned to the caller when the generator's next() method is invoked. However, unlike the return statement, the state of the generator function is saved, allowing it to resume execution from where it left off the next time the next() method is called.

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 are used to apply a function to elements in a sequence. However, there are differences:

Map Calls:

* map applies a given function to each item in a sequence (or multiple sequences) and returns an iterator.
* It requires the use of a predefined function.
* It can handle multiple input sequences of the same length.
* It's often used for simple element-wise transformations.

List Comprehensions:

* List comprehensions create a new list by applying an expression to each item in a sequence.
* They allow you to create new sequences by specifying the transformation more directly.
* They can involve conditions and produce more complex results.
* They are often more concise and readable for simple transformations.

In summary, map is more suitable when you have a predefined function to apply, and you want to work with iterators. List comprehensions are better when you want to create a new list using a concise expression, and you may want to involve conditional statements.