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

The difference between enclosing a list comprehension in square brackets [] and parentheses () in Python is that the former creates a list, while the latter creates a **generator object.**

1. **What is the relationship between generators and iterators?**

In Python, a generator is a function that produces a sequence of values using the yield keyword, instead of returning a single value. When you call a generator function, it returns an iterator that allows you to access the generated values. This means that generators are a type of iterator, but not all iterators are generators.

Generators are a type of iterator that produce their elements on-the-fly, while regular iterators store their elements in memory. Generators are created using the yield keyword, while regular iterators implement the \_\_iter\_\_() and \_\_next\_\_() methods.

1. **What are the signs that a function is a generator function?**

There are several signs that a function is a generator function in Python:

* The function uses the **yield** keyword to produce a value, rather than the return keyword.
* The function contains at least one yield statement.
* When you call the function, it returns a generator object that implements the iterator protocol.
* You can loop over the elements of the generator using a for loop or convert it to a list using the list() function.

1. **What is the purpose of a yield statement?**

The purpose of a yield statement in Python is to return a value from a generator function. A generator function is a special type of function that produces a sequence of values using the yield keyword, rather than returning a single value.

A yield statement is similar to a return statement, but it does not terminate the function. Instead, it suspends the function's execution and saves its state, so that the next time the generator is resumed, it continues from the point where it left off. This allows the generator to produce a sequence of values, rather than just a single value.

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

In Python, the map() function and list comprehensions are similar in that they both allow you to apply a function to a sequence of elements and produce a new sequence of transformed elements. The main difference between the two is that map() returns an iterator, while a list comprehension returns a list.