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

2) What is the relationship between generators and iterators?

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

4) What is the purpose of a yield statement?

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

**ANS:**1) When a list comprehension is enclosed in square brackets (`[]`), it produces a list as the result.

- When a list comprehension is enclosed in parentheses (`()`), it creates a generator expression. Generator expressions are lazy evaluated, meaning they produce values on-the-fly as requested and do not store all the values in memory at once like lists do.

2)

- Generators are a type of iterator in Python. All generators are iterators, but not all iterators are generators.

- Generators are created using generator functions or generator expressions. They use the `yield` statement to produce a series of values lazily.

- Iterators, on the other hand, are objects that implement the iterator protocol, which consists of the `\_\_iter\_\_()` and `\_\_next\_\_()` methods. Iterators can be created from iterables using the `iter()` function.

3)

- A function is a generator function if it contains the `yield` statement.

- Generator functions use the `yield` statement to yield values one at a time, instead of returning a single result like regular functions.

4)

- The `yield` statement is used in generator functions to produce a series of values lazily.

- When a generator function encounters a `yield` statement, it temporarily suspends its execution and returns the yielded value to the caller. The state of the function is saved, allowing it to resume execution from the same point later when the next value is requested.

- This allows generator functions to generate values one at a time, consuming less memory and often improving performance compared to returning a full list of values.

5)

- Both map calls and list comprehensions are used to apply a function to each element of an iterable and produce a new iterable with the results.

- The main difference is in syntax and readability:

- Map calls use a functional approach, where you provide a function and an iterable as arguments. It applies the function to each element of the iterable and returns a map object, which is an iterator.

- List comprehensions provide a more concise and readable syntax for creating lists by applying an expression to each item in an iterable and optionally filtering the items based on a condition.

- List comprehensions are often preferred for their readability and flexibility, especially for simple transformations or filtering operations. However, map calls can be more memory-efficient for large datasets since they produce an iterator instead of a full list.