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.

1. Enclosing a list comprehension in square brackets creates a new list, whereas enclosing it in parentheses creates a generator expression, which is an iterable that generates values on the fly and does not construct the entire sequence in memory at once. For example, `[x for x in range(10)]` would create a list containing the integers from 0 to 9, whereas `(x for x in range(10))` would create a generator expression that produces the same values one at a time as you iterate over it.
2. Generators are a type of iterable that are defined using a function and the `yield` statement. When a generator function is called, it returns a generator object that can be iterated over using a for loop or other iteration tools. In contrast, iterators are objects that represent a stream of data that can be iterated over. A generator is a specific type of iterator that is defined using a generator function.
3. A function is a generator function if it contains at least one `yield` statement. The presence of a `yield` statement indicates that the function will return a generator object rather than a normal value.
4. The `yield` statement is used in a generator function to produce a value that can be iterated over one at a time. When the `yield` statement is executed, the current state of the generator function is saved, and the yielded value is returned to the caller. The next time the generator is iterated over, the function resumes execution from the point where it left off, and continues until the next `yield` statement is encountered.
5. Both map calls and list comprehensions are used to apply a function to each element of an iterable and produce a new iterable as a result. However, there are some differences between the two:

- Map calls apply a function to each element of an iterable and return an iterator that produces the results. List comprehensions create a new list by applying an expression to each element of an iterable.
- Map calls can be more memory-efficient than list comprehensions when dealing with large iterables, since they produce an iterator that generates the results on the fly rather than constructing a list in memory. However, list comprehensions can be more readable and easier to understand for simple operations.
- Map calls can also be combined with lambda functions to create simple one-line expressions. List comprehensions can also be combined with conditionals to filter the input iterable or modify the output in more complex ways.