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

**Difference between square brackets and parentheses in list comprehensions**:

* When you enclose a list comprehension in square brackets, it produces a list as the result.
* When you enclose a list comprehension in parentheses, it produces a generator expression. This is more memory-efficient because it generates items on-the-fly when needed, as opposed to creating the entire list in memory.

2) What is the relationship between generators and iterators?

**Relationship between generators and iterators**:

* Generators are a type of iterator. In Python, an iterator is an object that can be iterated (looped) over, and it typically has two methods, \_\_iter\_\_() and \_\_next\_\_(). Generators are a way to create iterators in a more concise and readable manner.
* While iterators can be created using classes and implementing the necessary methods, generators use functions with yield statements to create iterators.

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

**Signs that a function is a generator function**:

* A generator function contains one or more yield statements.
* It typically has a def statement defining the function.
* When the function is called, it doesn't execute immediately; it returns a generator object.
* The execution of the function is paused at yield statements and can be resumed later when items are requested from the generator.

4) What is the purpose of a yield statement?

**Purpose of a yield statement**:

* The yield statement is used in generator functions to yield a value to the caller while maintaining the function's state.
* It allows a generator function to produce a sequence of values one at a time without consuming a lot of memory. The function's state is saved between yield calls.
* It also allows the generator function to be paused and resumed, making it suitable for generating large sequences of data.

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

**Relationship between map calls and list comprehensions**:

* Both map calls and list comprehensions are used for applying a given function to each element of an iterable, producing a new iterable with the results.
* However, there are differences:
  + map returns a map object in Python 3, which is an iterable. To get a list, you usually need to convert it using list().
  + List comprehensions return a list directly, which is more convenient when you want a list as the result.
  + List comprehensions often provide more readable and concise syntax for simple transformations and filtering operations.
  + map can be useful when you need to apply a function to multiple iterables simultaneously.
  + List comprehensions are more flexible and versatile, allowing more complex expressions and filtering conditions.