**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.**

***SOLUTIONS***

1. *The difference between enclosing a list comprehension in square brackets* ***[]*** *and parentheses* ***()*** *is the type of object that is created. Enclosing a list comprehension in square brackets creates a list object, which is a mutable, ordered collection of elements. On the other hand, enclosing a list comprehension in parentheses creates a generator expression, which is an object that generates values on-the-fly, rather than creating a list of all the values at once.*
2. *Generators and iterators are closely related concepts in Python. A generator is a type of iterator, specifically a special kind of function that generates values one at a time, rather than creating a list of all the values at once. An iterator is any object that implements the* ***\_\_iter\_\_*** *method and has a* ***next*** *method that can be used to retrieve the next value in the sequence.*
3. *A function is a generator function if it contains a* ***yield*** *statement. The* ***yield*** *statement is used to return a value from the function and pause its execution, allowing the function to be resumed later on and continue executing from the point where it was paused. Generator functions return a generator object, which can be used to generate the values produced by the function one at a time.*
4. *The purpose of a* ***yield*** *statement is to return a value from a generator function and pause its execution, allowing the generator to be resumed later and continue executing from the point where it was paused. The* ***yield*** *statement allows generator functions to produce a sequence of values over time, rather than generating the entire sequence of values at once.*
5. *Map calls and list comprehensions are similar in that they allow you to apply a function to each element of a sequence and create a new sequence from the results. However, there are some differences between the two. Map calls return a map object, which is an iterable that generates the results of the function one at a time, rather than creating a list of all the results at once. List comprehensions, on the other hand, create a list object that contains all the results of the function. List comprehensions are generally more concise and easier to read, while map calls are more flexible and can be more efficient for large data sets.*