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

:-

enclosing a list comprehension in square brackets [] creates a list object, whereas enclosing a list comprehension in parentheses () creates a generator object.

# List comprehension

my\_list = [x\*\*2 for x in range(10)] # creates a list of the squares of the numbers 0-9

# Generator comprehension

my\_generator = (x\*\*2 for x in range(10)) # creates a generator that yields the squares of the numbers 0-9

2) What is the relationship between generators and iterators?

:-

An iterator is an object that allows us to traverse through a collection of data one item at a time. A generator is a special type of iterator that is defined using a function instead of a class.

# Iterator implementation

class FibIter:

def \_\_init\_\_(self, n):

self.n = n

self.curr = 0

self.next = 1

def \_\_iter\_\_(self):

return self

def \_\_next\_\_(self):

if self.curr >= self.n:

raise StopIteration

result = self.curr

self.curr, self.next = self.next, self.curr + self.next

return result

# Generator implementation

def fib\_gen(n):

curr, next = 0, 1

for \_ in range(n):

yield curr

curr, next = next, curr + next

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

:-

A generator function is a function that contains at least one yield statement. When the generator function is called, it returns a generator object which can be iterated over using the next() function.

4) What is the purpose of a yield statement?

:-

The yield statement in Python is used in generator functions to produce a value to the caller without terminating the function. When a generator function is called, it returns a generator object that can be iterated over to retrieve successive values produced by the generator.

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

:-

Both map calls and list comprehensions are used to perform operations on iterable objects and create new iterables as a result.

Map is a built-in function in Python that takes a function and one or more iterable objects as arguments, applies the function to each element of the iterables, and returns a map object that contains the results.

squares = map(lambda x: x\*\*2, range(1, 6))

List comprehension, on the other hand, is a syntactic construct that allows us to create a new list by applying an expression to each element of an iterable.

squares = [x\*\*2 for x in range(1, 6)]

One major difference between map and list comprehension is that map returns a map object, which is an iterable that generates the results on demand, whereas list comprehension returns a list object, which is a collection of all the results at once.

map can take any number of iterables as arguments and applies the function to corresponding elements of each iterable, whereas list comprehension can only work with one iterable at a time.