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

If enclosed in square brackets, it becomes list comprehension.

If enclosed in paranthesis, it becomes generator comprehension.

Example:

[n\*2 for n in range(1,10) if n%2==0] # This is list comprehension

(n\*2 for n in range(1,10) if num%2==0) # This is generator comprehension.

1. What is the relationship between generators and iterators?

If a function has yield statement, then that function becomes a generator function.

Iterator is an object that has a method next() to fetch sequence of values.

A generator is built by calling a function that has one or more yield statements.

A generator is an object, that meets the defintion of iterator.

Every generator is a iterator.

1. What are the signs that a function is a generator function?

If a function has yield statement, then that function becomes a generator function.

generator functions have one or more yield statements in them.

another difference with regular functions and generator functions is that, when you call a function, the code in function is run.

in case of generator function, when you call a generator function, the object from the generator function is returned.

the object of the generator function can be iterated only once, until all the items in it are consumed.

1. What is the purpose of a yield statement?

in a normal function return senda specific value to the caller, whereas yield sends sequence of values.

The purpose of yield statement is to produce a sequence of values.

Yield is used when when we want to iterate over a sequence.

next() is used to fetch 1st and subsequent elements in the iterator

def function1():

yield 1

yield 2

yield 3

yield 4

yield 5

x = function1()

print(next(x)) # prints 1

print(next(x)) # prints 2

print(next(x)) # prints 3

print(next(x)) # prints 4

print(next(x)) # prints 5

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

map() is used if we want to compute the same function for different values in a single line.

syntax: map(functionname, iterables, return type)

List comprehension is a substitute for map(). It provides a concise way to create lists.

It provides extra features than map() such as filtering.

syntax: [expression for item in list if conditions]

comparison between map() and list comprehension

list comprehension is more concise and easier to read than map()

list comprehension allows filtering. map() does not allow filtering

list comprehension is used when list of results is required. map returns only a map object.

list comprehension is faster than map when we need to evaluate expressions which are too long or complicated.

Map is faster in case of calling an already defined function.