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

l=[i for i in range(10)]

print(type(l))

output:

<class 'list'>

l=(i for i in range(10))

print(type(l))

output:

<class 'generator'>

1. What is the relationship between generators and iterators?

*# Example of iterartor*

iter\_str **=** iter(['iNeuron','Full','Stack','Data Science'])

print(type(iter\_str))

print(next(iter\_str))

print(next(iter\_str))

print(next(iter\_str))

print(next(iter\_str))

print(iter\_str) *# After the iterable object is completed, to use them again we have reassign them to the same object.*

*# Example of Generator*

**def** cube\_numbers(in\_num):

**for** ele **in** range(in\_num**+**1):

**yield** ele**\*\***3

out\_num **=** cube\_numbers(4)

print(next(out\_num))

print(next(out\_num))

print(next(out\_num))

print(next(out\_num))

print(next(out\_num))

<class 'list\_iterator'>

iNeuron

Full

Stack

Data Science

<list\_iterator object at 0x00000193F0A88BE0>

0

1

8

27

64

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

A generator function uses a yield statement instead of a return statement. A generator function will always return a iterable object called generator. where as a normal function can return a string/list/tuple/dict/NoneType ... etc

1. What is the purpose of a yield statement?

The yield statement suspends function’s execution and sends a value back to the caller, but retains enough state to enable function to resume where it is left off. When resumed, the function continues execution immediately after the last yield run. This allows its code to produce a series of values over time, rather than computing them at once and sending them back like a list.

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

List comprehension is more concise and easier to read as compared to map.

List comprehension allows filtering. In map, we have no such facility. For example, to print all odd numbers in range of 50, we can write [n for n in range(50) if n%2 != 0]. There is no alternate for it in map

List comprehension are used when a list of results is required as final output.but map only returns a map object. it needs to be explicitly coverted to desired datatype.

List comprehension is faster than map when we need to evaluate expressions that are too long or complicated to express

Map is faster in case of calling an already defined function on a set of values.