Assignment 25

# Q1

The difference between enclosing a list comprehension in square brackets [ ] and parentheses ( ) lies in the type of object they create.

Square brackets [ ]: When a list comprehension is enclosed in square brackets, it creates a list object. The result of the list comprehension is a new list containing the generated elements.

Parentheses ( ): When a list comprehension is enclosed in parentheses, it creates a generator object. The result of the list comprehension is a generator, which is an iterable object that generates the elements on-the-fly as they are needed

# Q2

In Python, iterators and generators are closely related. An iterator is an object that can be iterated over, and a generator is a function that produces an iterator.

Iterables are objects that can be iterated over, such as lists, strings, and dictionaries. They have an \_\_iter\_\_() method that returns an iterator object.

Generators are functions that produce iterators. They use the yield keyword to return values from the function without actually terminating the function.

In other words, generators are a special type of function that can be used to create iterators. They are a convenient way to create iterators, and they can be used to implement lazy evaluation.

# Q3

A function is recognized as a generator function when it contains the yield statement. The presence of yield indicates that the function will create a generator object when called, allowing it to be iterated over using a loop.

# Q4

The purpose of a yield statement in Python is to define a generator function and control the generation of values in a lazy and memory-efficient manner. When a function contains a yield statement, it becomes a generator function.

When a yield statement is reached during the execution of a generator function, it temporarily suspends the function's execution and returns a value to the caller. This yielded value can be used or processed by the caller, such as storing it in a variable or performing some operation on it.

# Q5

Map calls and list comprehensions are both Python constructs that can be used to iterate over a sequence and apply a function to each element. However, there are some key differences between the two.

Map calls take a function and a sequence as input, and they return a map object. The map object can then be iterated over to get the results of the function applied to each element of the sequence.

List comprehensions are a concise way to write a for loop that creates a new list.