**Part B – Exercise 14:**

In the given program, the term "lazy evaluation" refers to the behavior where the evaluation of expressions is deferred until their values are actually needed. Let's break down how lazy evaluation applies to the program:

**Eager evaluation:** In eager evaluation, all expressions are evaluated immediately when they are encountered. In the provided code, the line **values = list(generate\_values())** triggers the execution of **generate\_values()** function, and the generator yields the values 1, 2, and 3. Then, the list comprehension **[square(x) for x in values]** is evaluated, resulting in the execution of the **square()** function for each value in the **values** list. This means that all values are generated and squared eagerly, even if they might not be needed later.

**Lazy evaluation:** In lazy evaluation, expressions are evaluated only when their values are required. In this case, the line **squared\_values = [square(x) for x in generate\_values()]** doesn't immediately trigger the execution of **generate\_values()**. Instead, it creates a generator expression that is not evaluated until it is iterated over or otherwise consumed. The generator will yield values as they are needed, and each value will be squared by the **square()** function at that point. This means that values are generated and squared only as they are needed, which can save computation time and resources, especially if not all values are ultimately required.

In summary, lazy evaluation defers the computation of values until they are actually needed, allowing for more efficient use of resources by avoiding unnecessary computation.