1. Create an assert statement that throws an AssertionError if the variable spam is a negative integer.

**Ans**: assert spam >= 0, "spam should be a non-negative integer"

1. Write an assert statement that triggers an AssertionError if the variables eggs and bacon contain strings that are the same as each other, even if their cases are different (that is, 'hello' and 'hello' are considered the same, and 'goodbye' and 'GOODbye' are also considered the same).

**Ans**: assert eggs.lower() != bacon.lower(), "eggs and bacon should not be the same (case-insensitive)"

1. Create an assert statement that throws an AssertionError every time.

**Ans**: assert False, "This assert statement will always trigger an AssertionError"

1. What are the two lines that must be present in your software in order to call logging.debug()?

**Ans**: import logging

logging.basicConfig(level=logging.DEBUG)

1. What are the two lines that your program must have in order to have logging.debug() send a logging message to a file named programLog.txt?

**Ans**: import logging

logging.basicConfig(filename='programLog.txt', level=logging.DEBUG)

1. What are the five levels of logging?

**Ans**: DEBUG, INFO, WARNING, ERROR, CRITICAL

7. What line of code would you add to your software to disable all logging messages?

**Ans**: logging.disable(logging.CRITICAL)

8.Why is using logging messages better than using print() to display the same message?

**Ans**: Because,

* Logging provides different levels, making it easy to filter messages.
* Logging can output messages to different destinations (file, console, etc.).
* Logging messages can include additional information like timestamps, log levels, etc.

9. What are the differences between the Step Over, Step In, and Step Out buttons in the debugger?

**Ans**: Step Over: Executes the current line of code and stops at the next line. If the current line contains a function call, it won't step into the function.

Step In: Executes the next line of code and, if the line contains a function call, steps into the called function, stopping at the first line of the function.

Step Out: Continues execution until the current function returns, then stops.

10.After you click Continue, when will the debugger stop?

**Ans**: The debugger will stop when it encounters a breakpoint, an exception is raised and not caught, or the program completes its execution.

11. What is the concept of a breakpoint?

**Ans**: A breakpoint is a designated point in your code where the debugger should temporarily pause execution, allowing you to inspect variables, step through code, and analyze the program's state at that specific point.