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

Ans : assert spam >= 0, "spam should not be a negative integer."

2. 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:

eggs="hello"

bacon ="Hello"

assert eggs.lower() != bacon.lower(),"eggs and bacon are the same (case-insensitive)"

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

Ans**: assert False**, "This assert statement always throws an AssertionError.

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

Ans :

import logging

logging.debug("Your debug message here")

5. 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)

6. What are the five levels of logging?

Ans:

1.DEBUG

2.INFO

3.WARNIG

4.ERROR

5.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 :logging provides more flexibility, control, and features compared to simple print() statements. It's a more professional and effective way to handle log messages in software development, especially for long-term projects and production environments.

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 and moves to the next line in the same function.

"Step In" allows you to enter and debug the function called on the current line.

"Step Out" finishes executing the current function and returns to the calling function.

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

Ans : It allows the program to continue its execution without any further interruption by the debugger until a stopping condition is encountered.

11. What is the concept of a breakpoint?

Ans : Breakpoints are invaluable tools for debugging complex programs and identifying the root cause of issues.