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

Answer 1: Here: assert spam >= 0, "Variable '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).**

Answer 2: Here: assert eggs.lower() != bacon.lower(), "Variables 'eggs' and 'bacon' should not have matching strings."

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

Answer 3: Here: assert False, "This assertion always triggers an AssertionError."

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

Answer 4:

import logging

logging.basicConfig(level=logging.DEBUG)

**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?**

Answer 5:

import logging

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

**6. What are the five levels of logging?**

Answer 6: The five levels of logging, in increasing order of severity, are: DEBUG, INFO, WARNING, ERROR and CRITICAL

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

Answer 7:

import logging

logging.disable(logging.CRITICAL)

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

Answer 8: Logging allows for different log levels, helping in categorizing the severity of messages. Log messages can be directed to different outputs (console, file, etc.) without modifying the code. Logging can be easily enabled or disabled globally, providing more control over debugging output. It provides timestamp information and log levels, aiding in debugging and monitoring.

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

Answer 9:

Step Over: Executes the current line and stops at the next line in the same function.

Step In: If the current line contains a function call, it will enter the called function and stop at the first line inside that function.

Step Out: Executes the remaining lines of the current function and stops after returning from the current function.

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

Answer 10: After you click Continue, the debugger will stop when it encounters the next breakpoint or when the program finishes execution.

**11. What is the concept of a breakpoint?**

Answer 11: A breakpoint is a designated point in your code where the debugger will pause execution. It allows you to inspect variables, step through code, and analyze the program's state at that specific point. Breakpoints aid in debugging by providing control over when and where the program stops for closer examination. They are essential for interactive debugging and problem identification.