1. Create an assert statement that throws an AssertionError if the variable spam is a negative integer.
Ans:- You can create an assert statement that throws an AssertionError if the variable spam is a negative integer like this:
assert spam >= 0, "Spam variable is 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:- 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 can be written as
assert eggs.lower() != bacon.lower(), "The eggs and bacon variables are the same!"

3. Create an assert statement that throws an AssertionError every time.
Ans:- An assert statement that always triggers an AssertionError can be written as:-
assert False, "This assertion always triggers."

4. What are the two lines that must be present in your software in order to call logging.debug()?
Ans:- The two lines that must be present in your software in order to call logging.debug() are:-
import logging
logging.basicConfig(level=logging.DEBUG, format='%(asctime)s - %(levelname)s - %(message)s')

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:- The two lines that your program must have in order to have logging.debug() send a logging message to a file named programLog.txt are:-
import logging
logging.basicConfig(filename='programLog.txt', level=logging.DEBUG, format='%(asctime)s - %(levelname)s - %(message)s')

6. What are the five levels of logging?
Ans:- The five levels of logging are: DEBUG, INFO, WARNING, ERROR, and CRITICAL

7. What line of code would you add to your software to disable all logging messages?
Ans:- To disable all logging messages in your program, you would add the following line of code:-
logging.disable(logging.CRITICAL)

8.Why is using logging messages better than using print() to display the same message?
Ans:- Using logging messages is better than using print() to display the same message because5:

- Logging provides different levels of severity that allows you to display log messages according to the level you want.
- Logging allows you to direct the log messages to separate files that can then be used for post analysis.
- You can disable logging messages without removing the logging function calls.
- Logging messages provide a timestamp.

9. What are the differences between the Step Over, Step In, and Step Out buttons in the debugger?
Ans:- The differences between the Step Over, Step In, and Step Out buttons in the debugger are6:

- Step Into: Goes inside the next function and lets you see how the function is executing line by line till it returns.
- Step Over: Executes the next function like a black box and returns the result, but you cannot see how the function was executed.
- Step Out: If you have stepped into a function and now want to skip seeing how the rest of the function is going to execute, you step out and the function returns.

10.After you click Continue, when will the debugger stop ?
Ans:- After you click Continue in the debugger, it will stop when it has reached the end of the program or a line with a breakpoint.

11. What is the concept of a breakpoint?
Ans:- A breakpoint is a setting on a line of code that causes the debugger to pause when the program execution reaches that line.