1. Create an assert statement that throws an AssertionError if the variable spam is a negative integer.  
Ans. assert spam >= 0, "spam cannot 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. assert eggs.lower() != bacon.lower(), 'The eggs and bacon variables contain the same string (case insensitive).'

3. Create an assert statement that throws an AssertionError every time.  
Ans. assert False, "This assertion always fails"

4. What are the two lines that must be present in your software in order to call logging.debug()?  
Ans. The first line imports the logging module, which provides the logging functionality. The second line configures the logging level to DEBUG, which is the lowest logging level and logs all messages, including debug messages.

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 first line imports the logging module, which provides the logging functionality. The second line configures the logging to write to a file named programLog.txt and sets the logging level to DEBUG.

After these two lines, you can call logging.debug() to log debug messages in your code, and they will be written to the programLog.txt file.

6. What are the five levels of logging?  
Ans. DEBUG, INFO, WARNING, ERROR and 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 has different levels of severity that allows you to display log messages according to the level you want. A print statement does not give you that flexibility.

Logging allows you to direct the log messages to separate files that can then be used for post analysis while the same is not easily available with print statement.

You can set different log levels at individual code file level as well - some files may have INFO level while some may have DEBUG level.

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

The Step Over button is used to execute the current line of code and move to the next line of code.

Step In:

The Step In button is used to step into the current line of code and debug the function call if it exists.

Step Out:

The Step Out button is used to execute the current function and move to the next line of code in the calling function.

10.After you click Continue, when will the debugger stop?  
Ans. After clicking Continue, the debugger will not stop until it reaches a breakpoint or encounters an unhandled exception.

11. What is the concept of a breakpoint?  
Ans. A breakpoint is a specific point in the code where the debugger will pause execution and allow the programmer to inspect the current state of the program.