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

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).

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

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

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()?

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?

import logging

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

6. What are the five levels of logging?

 The five levels of logging, in increasing order of severity, are:

* DEBUG
* INFO
* WARNING
* ERROR
* CRITICAL

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

import logging

logging.basicConfig(level=logging.CRITICAL)

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

 Using logging messages is better than using print() for several reasons:

* Logging allows you to control the verbosity of messages with different log levels.
* Log messages can be directed to different outputs (e.g., console, files) without changing the code.
* Logging provides timestamp information and can be customized for different parts of your code.
* In a production environment, logs are essential for debugging and monitoring, and you can easily adjust log levels to get more or less detail.

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

 The differences between the Step Over, Step In, and Step Out buttons in the debugger are as follows:

* **Step Over**: Executes the current line of code and stops at the next line. If the next line is a function call, it will execute the entire function and stop at the next line in the current function.
* **Step In**: If the current line contains a function call, it will step into that function and pause at the first line of the called function. If the current line does not contain a function call, it behaves the same as Step Over.
* **Step Out**: Continues execution until the current function returns, and then it stops at the line that called the function. It effectively "steps out" of the current function.

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

After clicking "Continue," the debugger will stop when it encounters the next breakpoint or when the program finishes executing.

11. What is the concept of a breakpoint?

A breakpoint is a designated point in your code where you want the debugger to pause execution. It allows you to inspect variables, step through code, and troubleshoot issues. You can set breakpoints in your code to stop at specific locations during debugging.