1. Create an assert statement that throws an AssertionError if the variable spam 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).

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

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

6. What are the five levels of logging?

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

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

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

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

11. What is the concept of a breakpoint?

Answers:

1. assert spam >= 0, "spam cannot be negative"
2. assert eggs.lower() != bacon.lower(), "eggs and bacon cannot be the same"
3. assert False, "This assertion always fails"
4. The two lines that must be present are:

import logging logging.basicConfig()

1. The two lines that the program must have are:

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

1. The five levels of logging are: DEBUG, INFO, WARNING, ERROR, and CRITICAL.
2. logging.disable(logging.CRITICAL)
3. Using logging messages is better than using print() to display the same message because logging messages can be selectively turned on or off at different levels, and they can be written to different outputs such as a file or a console. This makes it easier to control and manage the amount of information that is being displayed, and it allows for more flexibility in debugging and troubleshooting.
4. The Step Over button allows the debugger to execute the current line and move to the next line in the program. The Step In button allows the debugger to step into a function call and examine its contents. The Step Out button allows the debugger to finish executing the current function and return to the calling function.
5. The debugger will stop again when it encounters another breakpoint, or when the program terminates.
6. A breakpoint is a point in the code where the debugger will pause execution so that the developer can examine the program's state and debug any issues. A breakpoint can be set on a specific line of code, and the debugger will halt execution when that line is reached.