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

**Ans:**

**assert spam >= 0, 'The variable spam is a negative number.'**

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.upper() != bacon.upper(), 'Variables in Eggs and Bacons are same '**

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

**Ans:**

**assert 1==2,’Something is Wrong’**

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

**Ans:**

**import logging**

**logging.basicConfig(filename=’test.log’, 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:**

**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: Debug, Info, Error, Warning and Critical**

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

**Ans: logging.shutdown()**

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

**Ans: Easy to see where and when (even what line no.) a logging call is being made from. You can log to files, sockets, pretty much anything, all at the same time. You can differentiate your logging based on severity.**

**You can use it for debugging the error in the code or system even after many days.**

**Printing is not the recommended idea since it will last only till your code is running. Secondly, you do not want to show error messages or other sensitive messages to your users/clients.**

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

**Ans: If the line does not contain a function it behaves the same as “step over” but if it does the debugger will enter the called function and continue line-by-line debugging there, it is ‘Step In’. Step out – An action to take in the debugger that returns to the line where the current function was called.**

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

**Ans: Debugger will stop when it will reach the end of the program or when it will hit the next breakpoint.**

11. What is the concept of a breakpoint?

**Ans: A breakpoint is an intentional stopping or pausing place in a program, put in place for debugging purposes**