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

Ans>>

2. Create an assert statement that throws an AssertionError if the variables eggs and bacon contain strings that are the same, even if their cases are different (for example, 'hello' and 'hello' are considered the same, as are 'goodbye' and 'GOODbye').

Ans>>

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

Ans>>

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

Ans>>

5. What are the two lines that must be present in your software in order for it to log?

Ans>>

Is it possible for logging.debug() to log a message to a file called programLog.txt?

Ans>> Yes It is possible, for that we have to add an extra argument which is the file name in where we want to add the log message instead of printing them in our output terminal.

6. What are the five stages of logging?

Ans>> There are five stages of logging:-

DEBUG - logging.debug()

INFO – logging.info()

WARNING – logging.warning()

ERROR – logging.error()

CRITICAL – logging.critical()

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

Ans>> logging.disable() will disable all logging message. We often use this function after the completion of debugging and when our program or ready for deployment or ready to use.

8. Why is it easier to use logging messages instead of print() to show the same message?

Ans>> The main issue is for using the print() function for showing the log message is after the completion of our work we have to delete all the print() line otherwise it will keep printing when we will use the program or that software. It will be very irritating and unnecessary, we sure don’t want this to happen. And also this is quite very hard to remove all print() function from our code when it’s a thousand lines of the program. But if we use logging then by adding just one line we can remove all the messages that we were trying to show by print() statement. And then it will not show the message when we will be using the program. So that is the reason using logging is much easier and efficient.

9. What is the difference between the debugger's Step Over, Step In, and Step Out buttons?

Ans>> Step Over - After pressing the debugger’s step Ove button the debugger will step over the following function, and in this time the code will run normally without using the debugging. This can be used for our in-built function like print() or len(), we sure don’t want to check or debugging the line of code which is inside of these function so we can safely step over these function. Here we can use step-over button.

Step In – When we use this step In here debugger will check every line of code in our program, and will not step over any function, the debugger will check every line of code inside that function or any following function.

10.When will the debugger stop after you press Continue?

Ans>> After pressing continue the debugger will stop when it reaches a breakpoint or when the line end.

11. What is the concept of a breakpoint?

Ans>> Breakpoint is a function that we can import at any position of our program. By putting this in our program our debugger will stop at this point.

12. In Mu, how do you place a breakpoint on a line of code?

Ans>>