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

assert spam>=0, 'The spam variable is a negative value'

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(), 'The strings in eggs and bacon represent the same word'

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

assert False, "Assertion Error is Triggered forever"
Since, at any point of time, some condition is always True, **"assert False"** will be always triggered as soon as it comes.

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, 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?

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?

DEBUG, INFO, WARNING, ERROR, and CRITICAL

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

logging.disable(logging.CRITICAL) can be used to disable all the logging messages.

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

Logging is used by the programmer to check how the code is performing. print() need to be used at all the locations wherever the desired information needs to be communicated. But, at the end of the day while releasing the production version of the code, all the print commands need to be removed. This task is tedious and we may end up getting rid of important print messages. You can disable logging messages without removing the logging function calls. You can selectively disable lower-level logging messages. You can create logging messages. Logging messages provides a timestamp.

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

The Step In button will move the debugger into a function call. The Step Over button will quickly execute the function call without stepping into it. The Step Out button will quickly execute the rest of the code until it steps out of the function it currently is in.

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

After you click Continue, the debugger will stop when it has reached the end of the program or a line with a breakpoint.

11. What is the concept of a breakpoint?

A breakpoint is a setting on a line of code that causes the debugger to pause when the program execution reaches the line.