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

In Python, the assert statement is used to continue the execute if the given condition evaluates to True. If the assert condition evaluates to False, then it raises the AssertionError exception with the specified error message.

spam= 0

assert spam > 0, 'Only positive numbers are allowed'

print('spam is a positive 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).**

assert eggs.lower() = bacon.lower(), 'The eggs and bacon variables are the same!'

or

assert eggs.upper() = bacon.upper(), 'The eggs and bacon variables are the same!

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

assert False, 'This assertion always triggers.'

**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)

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

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

Logging is an abstract way of printing to console. Using proper logging framework you specify what information you want to log and what is the importance of it. The configuration of a logging framework then decides whether: Put it in /dev/null.

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

The differences are as follows:

Step Over command:

This command lets the application execute the next action. If the action involves a call to an operation, it does not step into its implementation (it steps over it instead).

Step Into command:

This command lets the application execute the next action. If the action involves a call to an operation, it steps into its implementation and breaks the execution on the first action of that implementation.

Step Out command:

This command lets the application execute until the currently executed operation implementation is returned.

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

After you click Go, 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.