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.

**Code**:

x = -1

assert x >= 0, 'Only positive numbers are allowed'

print('x is a zero or 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).

eggs='goodbye'

bacon ='GOODbye'

assert eggs.lower() != bacon.lower() or eggs.upper() != bacon.upper()

print('The eggs and bacon variables are not the same!')

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

assert False

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

import logging as lg

lg.basicConfig(level=lg.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 as lg

lg.basicConfig(filename='programLog.txt', level=lg.DEBUG, format=' %(asctime)s - %(levelname)s - %(message)s')

6. What are the five levels of logging?

There are five built-in levels of the log message.

Debug : These are used to give Detailed information, typically of interest only when diagnosing problems.

Info : These are used to confirm that things are working as expected

Warning : These are used an indication that something unexpected happened, or is indicative of some problem in the near future

Error : This tells that due to a more serious problem, the software has not been able to perform some function

Critical : This tells serious error, indicating that the program itself may be unable to continue running

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

import logging as lg

lg.disable(lg.DEBUG)

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

Logging is better than print() because:

* The print() statement fails if your code does not have access to the console.
* To define basic logging needs, several lines of code are needed.
* Including additional logging information is not easy.
* The print() statement only displays messages on the console. Recording logging data inside a file or sending it over the internet needs additional works.
* You can selectively enable/disable a level of 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 Over button will quickly execute the function call without stepping into it.

The Step In button will move the debugger into a function call.

The 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 ?

It will stops at next breakpoint, if there are no further breakpoints program will be fully executed.

11. What is the concept of a breakpoint?

A breakpoint is an intentional stopping point or pause put into a program for debugging purposes.