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

**integer.**

**Ans.** the assert statement is as follows:

assert spam >= 0, ‘The spam value is negative’

**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.lower() != bacon.lower(), ‘The values are same’

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

**Ans.** assert False, ‘assert thrown everytime’

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

**Ans.** first we must import the logging library and then provide the basic configuration using the basicConfig() method along with the levelname attribute being set to logging.DEBUG.

**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.** This is more specific so

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.** The 5 level of logging are:

* Debug
* Error
* Warning
* Info
* Critical

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

**Ans.** we will call the disable method of the logging.

logging.disable(logging.CRITICAL)

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

**Ans.** The level of control and personalisation might be the best reason we can say here. The logging can be used with different levels based on the programmers need and can be disabled whenever needed without removing the calls. We can include information such are timestamp and custom string containing the location of the problem etc.

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

**Ans.** Step over button is used to skip to the function end of the function by quickly executing everything inside that function. The In is to get inside the function and follow the execution of the statements and out is to get out of the function by executing all the further left statements and stepping out of that function.

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

**Ans.** it will stop when it encounters the breakpoint as it only stops when breakpoint is encountered.

**11. What is the concept of a breakpoint?**

**Ans.** It is an intentional stopping or pausing place in a program, that is put in place for debugging purposed.