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

ANS) assert spam >= 10, 'The spam variable is less than 10.'

**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 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.**

ANS)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()?**

ANS)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?**

ANS)To be able to send logging messages to a file named programLog.txt with logging.debug(), you must have these two lines at the start of your program:

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) DEBUG, INFO, WARNING, ERROR, and CRITICAL

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

ANS)logging.disable(logging.CRITICAL)

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

ANS) A **logger** is an object that records events. You use it like you would use the **print** function, but it is much more powerful **than print** . Every **logging** event has a message, but also records other information like line number and module name.

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

ANS) The Step button will move the debugger into a function call. The Over button will quickly execute the function call without stepping into it. 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 ?**

ANS)  This will cause your program to continue running normally, without pausing for **debugging**. It may run to completion.

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

ANS)When a **programmer** creates code they can add what is known as a **breakpoint**. A **breakpoint** is a point in the **program** where the code will stop executing.