**Assignment 11 Solution**

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

**Answer :-**

**assert spam >= 5 ‘ The spam variable is less than 5. ‘**

Q.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).

**Answer :-**

**assert eggs.lower() != bacon.lower(), ‘ The eggs and bacon variable are same ‘**

**assert eggs.upper() != bacon.upper(), ‘ The eggs and bacon variable are same ‘**

Q.3 => Create an assert statement that throws an AssertionError every time.

**Answer :-**

**assert False, This always throw an AssertionError.**

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

**Answer :-**

**To be able to call logging.debug() you must have these two lines at the start of the program.**

**import logging**

**logging.basicConfig(level = logging.DEBUG, format = ‘ %(asctime)s - %(levelname)s - %(message)s’)**

Q.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?

**Answer :-**

**To be able to send logging message to a file name programLog.txt with logging.debug(), you must have these two lines at the start of the program.**

**import logging**

**logging.basicConfig(filename = “programLog.txt”, level = logging.DEBUG**

**format = ‘ %(asctime)s - %(levelname)s - %(message)s’)**

Q.6 => What are the five levels of logging?

**Answer :-**

**INFO, DEBUG, ERROR, WARNING and CRITICAL.**

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

**Answer :-**

**logging.disable(logging.CRITICAL)**

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

**Answer :-**

**You can disable logging message without removing the function calls.**

**You can selectively disable lower-level logging message.**

**You can create logging message.**

**Logging message provide a timestamp.**

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

**Answer :-**

**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 code until it steps out of the function it currently is in.**

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

**Answer :-**

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

Q.11 => What is the concept of a breakpoint?

**Answer :-**

**A breakpoint setting on line of code that cause the debugger to pause when the program execution reaches the line.**