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

**Ans:**

**def asser(spam):**

**assert spam >=0, "spam should be positive integer "**

**print(spam\*2)**

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:**

**def test(eggs,bacon):**

**try:**

**assert eggs.lower() != bacon.lower()**

**except AssertionError as msg:**

**print("Both eggs and bacon are same")**

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

**Ans:**

**def test1(x):**

**try:**

**assert 1==2**

**except AssertionError as msg:**

**print("error")**

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

**Ans:** This two below statements are mandatory

**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:**

**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:**

**1. Logging messages provides timestamp**

**2. A log file created we can view log messages later as well.**

**3. With logging we can predict what kind of error messages are occurred (info, warning,error)**

**4. if we want, we can disable log messages.**

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

**Ans:**

**Step over: Each sub-function is executed in one step, that is, sequentially in the order of the given code. Shortcut key F8**

**Step out: When entering the sub-function internal execution, press step out to jump out of the sub-function and return to the previous level. Shortcut SHIFT+F8**

**Step into: Enter the sub-function, each step shows clear shortcuts F7**

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

**Ans:** **After you click Continue, 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?

**Ans: A breakpoint is a setting on a line of code that causes the debugger to pause when the program execution reaches the line.**