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

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).

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

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

6. What are the five levels of logging?

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

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

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

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

11. What is the concept of a breakpoint?

**ANS:  
1. ```python**

**assert spam >= 0, "spam must be a non-negative integer"**

**```**

**2. ```python**

**assert eggs.lower() != bacon.lower(), "eggs and bacon must be different"**

**```**

**3. ```python**

**assert False, "This assert statement always triggers an AssertionError"**

**```**

**4. The two lines that must be present to call `logging.debug()` are:**

**```python**

**import logging**

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

**```**

**5. To send `logging.debug()` messages to a file named `programLog.txt`, the two lines should be:**

**```python**

**import logging**

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

**```**

**6. The five levels of logging are: DEBUG, INFO, WARNING, ERROR, and CRITICAL.**

**7. To disable all logging messages, you can set the logging level to `logging.CRITICAL`:**

**```python**

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

**```**

**8. Using logging messages is better than using print() because:**

**- Logging messages can be easily configured to output to different destinations (console, file, etc.).**

**- Logging messages can be filtered based on severity levels, making it easier to manage and troubleshoot different parts of the code.**

**- Logging messages can provide additional information like timestamps and severity levels, aiding in debugging and monitoring.**

**9. In the debugger:**

**- Step Over executes the current line of code and stops at the next line, treating function calls as a single step.**

**- Step In executes the current line of code and stops at the next line, but steps into any function calls on that line, allowing you to follow the execution into function definitions.**

**- Step Out continues execution until the current function returns, then stops at the line where the function was called.**

**10. After clicking Continue, the debugger will stop when it encounters another breakpoint or when the program completes execution.**

**11. A breakpoint is a point in the program's execution where the debugger will pause execution, allowing you to inspect variables, evaluate expressions, and step through the code interactively. Breakpoints are typically set at specific lines of code or triggered based on conditions.**