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

Ans:  
 try :  
 assert spam >= 0, ‘spam variable is negative’  
 except AssertionError as a:  
 print(a)

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(), ‘eggs and bacon variable are same.’

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

Ans: assert False, ‘This statement always throws.’

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

Ans: The two lines that must be present are :  
 import logging as lg  
 lg.basicConfig(filename = ‘xyz.log’, level = lg.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: The two lines that must be present are :  
 import logging as lg  
 lg.basicinfo(filename = ‘programLog.txt’, level = lg.DEBUG, format = ‘%(asctime)s - %(levelname)s - %(message)s’)

6. What are the five levels of logging?

Ans: 1. Debug, 2. Info, 3. warnning, 4. Error, 5. Critical.

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

Ans: lg.disable(level=lg.INFO)

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

Ans: (i) the logging msg can provide timestamp but print() cannot.  
 (ii) we can disable logging without deleting logging functions, with print() we can’t do trhe same.  
 (iii) we can disable lower level logging message while keeping the upper level logging messages using logging.disable(level = \_\_\_\_ )

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: The debugger will stop when it has reached the end of the programme or a line with breakpoint.

11. What is the concept of a breakpoint?

Ans: Breakpoint is a specific line of code that cause the debugger to stop/pause when reached.