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

Ans: assert spam >= 10, '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: The below two lines should have to be available in order to call logging.debug():

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:

* 1. DEBUG
  2. INFO
  3. WARNING
  4. ERROR and
  5. 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: below are the reasons to use logging message over print():

* 1. Logging message provides a timestamp.
  2. It can be disabled without removing the logging function call.
  3. Lower-level logging message can only be disabled.
  4. Logging message can be created as well.

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

Ans:

* + Step Over - It will execute the function call without stepping into it.
  + Step In - This button moves the debugger into the function call.
  + Step Out - This button will quickly execute rest of the lines within the function and will come out of it.

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

Ans: It will stop at the next break point.

11. What is the concept of a breakpoint?

Ans: Breakpoint is used while we debug a program. We use breakpoint to stop/pause a program at a certain point.