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

🡪 assert spam > 0, the spam variable is negative.

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).

🡪 The above mentioned can be done as-

* assert eggs.lower() != bacon.lower(), it means the eggs and bacon variables are same
* assert eggs.upper() != bacon.upper(), it means the eggs and bacon variables are same

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

🡪 Following assert statement will always throw an AssertionError-

assert False

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

🡪Following are the two lines which must be present at the start of our program 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?

🡪 Following are the two lines which must be present at the start of our program in order to have logging.debug() send a logging message to a file named programmLog.txt:

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?

🡪 The five levels of logging are-

* INFO
* DEBUG
* WARNING
* ERROR
* CRITICAL

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

🡪 By following code, we can disable all logging messages-

logging.disable(logging.CRITICAL)

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

🡪You can disable logging messages without removing the logging function calls. You can selectively disable lower-level logging messages. You can create logging messages. Logging messages provides a timestamp.

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

🡪 The Step In button will move the debugger into a function call. The Step Over button will quickly execute the function call without stepping into it. The Step 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 ?

🡪 After clicking Continue the program will start running and will run to the end of program or until a breakpoint is reached.

11. What is the concept of a breakpoint?

🡪 A breakpoint is a kind of signal that tells the debugger to temporarily suspend execution of program at a certain point. And when this happens, that is, when the execution is suspended at certain point, the program is said to be in break mode. Break mode does not mean the complete stop or end of the execution of program, execution can be resumed any time again.