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

Ans.

Spam = -5

assert spam >=0, 'The variable spam is a negative integer'

1. 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.

eggs = 'hello'

bacon = 'HellO'

assert eggs.upper() != bacon.upper(), 'The eggs and bacon variables are the same!'

# OR

assert eggs.lower() != bacon.lower(), 'The eggs and bacon variables are the same!'

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

Ans.

assert False, 'This assertion triggers every time.'

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

Ans. start of your program with import logging followed by logging.basicConfig as

import logging

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

1. 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')

1. What are the five levels of logging?

Ans.

* 1. DUBUG
  2. INFO
  3. WARNING
  4. ERROR
  5. CRITICAL

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

Ans.

logging.disable(logging.CRITICAL)

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

Ans. It’s simple to debug in a complex project, logs are configurable, easy to categorize message, filter them and send them to files with proper format and timestamps. Easy to disable logs without removing logging function for any level of logging messages.

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

Ans.

**Step over:** Quickly execute the function without stepping into it (without debugging each line of code in function.

**Step In:** If the line does not contain a function, it behaves the same as “step over” but if it does, the debugger will enter the called function and continue line-by-line debugging there.

**Step out:** If debugger is inside a function, then quickly execute rest of the code in same function.

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

Ans. Continue execution, only stop when a breakpoint is encountered.

1. What is the concept of a breakpoint?

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