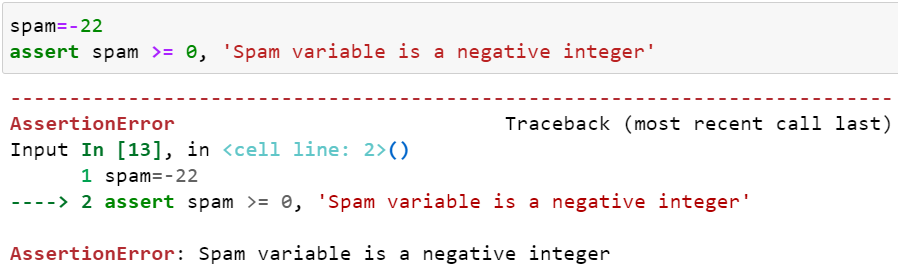
1. Create an assert statement that throws an AssertionError if the variable spam is a negative

integer.

**Answer :**

assert spam>=0, ‘Spam variable is a negative integer’





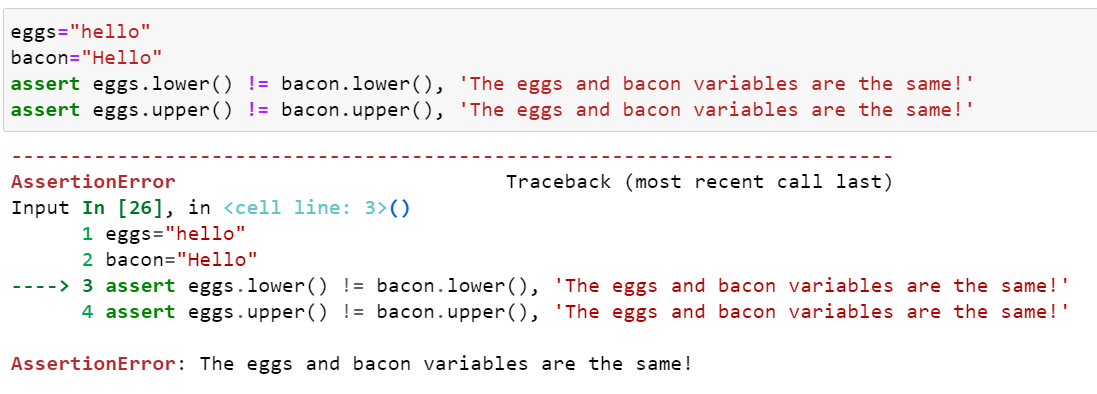
2. Write an assert statement that triggers an AssertionError if the variables eggs and bacon contain

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).

**Answer :**

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

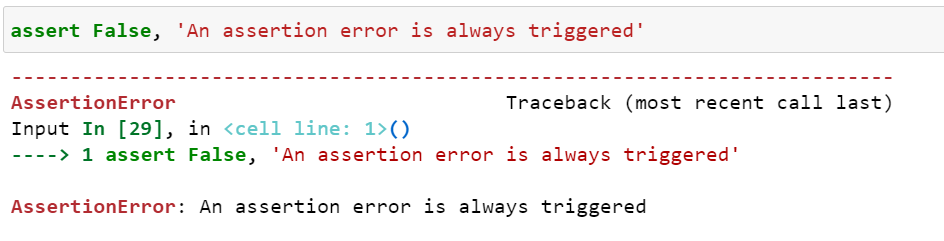
assert eggs.upper() != bacon.upper(), 'The eggs and bacon variables are the same!'



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

**Answer :**

Assert False, ‘An assertion error is always triggered’

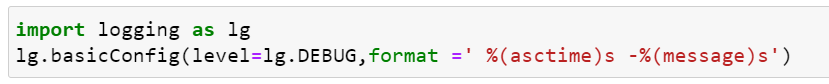


4. What are the two lines that must be present in your software in order to call logging.debug()?

**Answer :**

import logging as lg

lg.basicConfig(level=lg.DEBUG,format =' %(asctime)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?

**Answer :**

import logging

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

6. What are the five levels of logging?

**Answer :**

Following are the five levels of logging:

1. DEBUG
2. INFO
3. WARNING
4. ERROR
5. CRITICAL

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

**Answer :**

logging.disable(logging.CRITICAL)

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

**Answer :**

Print() only displays message on console. One of the biggest use of logging is that you can categorize the messages and turn them on and off depending on what you need. You cannot do this with print() statement. Logs are configurable. You can easily filter them, send them to files, format them, add timestamps and any other things that needs to be done on the of global basis. print() statements are not easily managed.

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

**Answer :**

* The step over button will quickly execute the function call without stepping into it.
* The step in button will move the debugger into function call.
* 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 ?

**Answer :**

Continuing means resuming program execution until program completes normally. So continue execution will only stop when a breakpoint is encountered.

11. What is the concept of a breakpoint?

**Answer :**

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