Write an assert statement that triggers an AssertionError if the variable spam is negative.

assert spam >= 0, 'The spam variable is less than 0.'

Write an assert statement that triggers and 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).

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!'

Write an assert statement that always triggers and AssertionError.

assert False, 'The assertion always triggers.'

What are the two lines that your program must have in order to be able to call logging.debug()?

To be able to call logging.debug(), you must have these two lines at the start ofyour program:  
  
import logging  
logging.basicConfig(level=logging.DEBUG, format=' %(asctime)s -  
%(levelname)s - %(message)s')

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?

To be able to send logging message to a file named programLog.txt with logging.debug(), you must have these two lines at the start of your:  
  
import logging  
>>> logging.basicConfig(filename='programLog.txt', level=logging.DEBUG,  
format=' %(asctime)s - %(levelname)s - %(message)s')

What are the five logging levels?

DEBUG, INFO, WARNING, ERROR, and CRITICAL

What line of code can you add to disable all logging messages in your program?

logging.disable(logging.CRITICAL)

Why is using logging messages better than upsing 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.

What are the differences between the Step, Over, and Out buttons in the Debug Control window?

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.

After you click Go in the Debug Control window, when will the debugger stop?

After you click Go, the debugger will stop when it has reached the end of the program or a line with a breakpoint.

What is a breakpoint?

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