1. pythonCopy code

assert spam >= 0, "spam should be a non-negative integer"

1. pythonCopy code

assert eggs.lower() != bacon.lower(), "eggs and bacon should not be the same (case-insensitive)"

1. pythonCopy code

assert False, "This assert statement always throws an AssertionError"

1. The two lines needed to call **logging.debug()** are:

pythonCopy code

import logging logging.debug("Your debug message")

1. To send a **logging.debug()** message to a file named **programLog.txt**, you need these two lines:

pythonCopy code

import logging logging.basicConfig(filename='programLog.txt', level=logging.DEBUG)

1. The five levels of logging, in increasing order of severity, are:
   * DEBUG
   * INFO
   * WARNING
   * ERROR
   * CRITICAL
2. To disable all logging messages, you can add the following line:

pythonCopy code

logging.disable(logging.CRITICAL)

1. Using logging messages is better than using print() for several reasons, including the ability to control log levels, direct log messages to different outputs, and easily disable or enable logging selectively.
2. Differences between Step Over, Step In, and Step Out buttons in the debugger:
   * Step Over: Executes the current line of code and stops at the next line. If the current line contains a function call, it does not enter the function but rather moves to the next line in the current function.
   * Step In: If the current line contains a function call, it enters the function and stops at the first line inside the function.
   * Step Out: Continues execution until the current function is completed and returns to the line where the function was called.
3. After you click Continue, the debugger will stop when it encounters a breakpoint, encounters an unhandled exception, or reaches the end of the program.
4. A breakpoint is a designated point in your code where the debugger should pause or break the execution of the program, allowing you to inspect variables, evaluate expressions, and step through the code interactively.