Assignment-11

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

Ans1- assert spam >= 0, "spam should not be a negative integer"

spam is the variable you want to check.

spam >= 0 is the condition you want to assert. If spam is not greater than or equal to 0 (i.e., it's a negative integer), the assertion will fail.

"spam should not be a negative integer" is an optional error message that will be displayed if the assertion fails. You can customize this message to provide more information about the problem.

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).

Ans2-assert eggs.casefold() != bacon.casefold(), "eggs and bacon should not be the same (case-insensitive)"

eggs.casefold() converts the eggs string to lowercase.

bacon.casefold() converts the bacon string to lowercase.

eggs.casefold() != bacon.casefold() compares the two lowercase strings to check if they are different.

"eggs and bacon should not be the same (case-insensitive)" is an optional error message that will be displayed if the assertion fails.

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

Ans3- assert False, "This assert statement always throws an AssertionError"

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

Ans4- import logging

logging.basicConfig(level=logging.DEBUG)

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?

Ans5-import logging

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

6. What are the five levels of logging?

Ans6- DEBUG: The lowest level of logging.

INFO: Used to provide general information about the program's execution.

WARNING: Indicates potential issues or unexpected events that are not errors but may need attention.

ERROR: Indicates that an error or exception has occurred, but it is not critical enough to terminate the program.

CRITICAL: The highest level of severity.

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

Ans7- logging.disable(logging.CRITICAL)

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

Ans8- a) Logging allows you to configure where and how log messages are handled.

b) With logging, you can set different log levels for different parts of your code.

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

Ans9- Step Over moves to the next line in the current scope, including function or method calls, without diving into them.

Step In dives into the details of a function or method call, allowing you to debug its code.

Step Out exits the current function or method and returns to the calling function's or method's context.

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

Ans10- a) The debugger will stop when it encounters a breakpoint that has been set in your code.

b) If an unhandled exception or error occurs during program execution, the debugger will stop and allow you to examine the exception, its stack trace, and the state of the program at the point where the exception occurred. Debuggers often have an option to automatically break on unhandled exceptions.

c) You can manually pause the execution by clicking a "Pause" or "Stop" button in the debugger's interface.

11. What is the concept of a breakpoint?

Ans11- The debugger will stop when it encounters a breakpoint that has been set in your code. Breakpoints are specific lines or conditions in your code where you've instructed the debugger to pause execution so you can inspect the program's state.