Assignment 11:

## Q1

assert spam >= 0, "spam cannot be a negative integer"

## Q2

assert eggs.lower() != bacon.lower(), "eggs and bacon are the same"

## Q3

assert False, "This assertion always triggers an AssertionError"

## Q4

import logging

logging.basicConfig(level=logging.DEBUG)

## Q5

import logging

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

## Q6

DEBUG: Detailed information for debugging purposes.

INFO: General information about the program's execution.

WARNING: Indication of potential issues or unexpected behaviour that is not necessarily an error.

ERROR: An error has occurred, but the program can still continue.

CRITICAL: A critical error has occurred, and the program may not be able to continue.

## Q7

logging.disable(logging.CRITICAL)

## Q8

There are several reasons why using logging messages is better than using print() to display the same message.

Logging messages are more flexible. we can control the level of logging, which means that we can only see the messages that we are interested in. We can also configure logging to send messages to different destinations, such as a file or a database.

. We can use the same logging configuration in different environments, such as development, staging, and production. Logging messages are more secure. We can configure logging to encrypt messages, which can help to protect sensitive information. print() statements can print a lot of output, which can slow down the code. Logging messages are only printed when they are needed, which can improve the performance of our code.

## Q9

Step Over- Executes the current line of code and then proceeds to the next line. If the current line of code is a function call, the function call will be executed as a single step.

Step In- Executes the current line of code and then steps into any function calls that are made on that line.

Step Out- Executes the current line of code and then steps out of any function calls that are currently active.

## Q10

After clicking the Continue button in a debugger, the debugger will stop if there is a Breakpoint, Exception, End of Program or Interruption.

## Q11

A breakpoint is an essential aspect of debugging. A breakpoint is a designated point in wer code where we want the debugger to pause execution, allowing we to inspect the program's state, analyse variables, and step through the code line by line.Breakpoints are useful for isolating specific sections of code that we want to investigate more closely or for pinpointing the source of an issue.