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

- spam= -2

Assert spam >= 0 ”spam should be a non negative integer”

1. **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).**

**-** eggs="hello"

bacon="Hello"

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

1. **Create an assert statement that throws an AssertionError every time.**

- assert -1==1,"assert statement that throws an AssertionError every time"

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

**-** import loggiging logging.basicConfig(level=logging.DEBUG,format='%(asctime)s - %(levelname)s -%(messgae)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?**

**-** import loggiging logging.basicConfig(filename= “programLog.txt ’level=logging.DEBUG,format='%(asctime)s - %(levelname)s -%(messgae)s')

**6. What are the five levels of logging?**

- DEBUG , INFO , WARNNING, ERROR , CRTICAL

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

- logging.disable(logging.CRITICAL)

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

**-**Once we're done debugging, we'll end up spending a lot of time removing print() calls from our code for each log message. We might even accidentally remove some print() calls that were being used for nonlog messages. The nice thing about log messages is that you're free to fill your program with as many as you like, and you can always disable them later by adding a single logging.disable(logging.CRITICAL) call. Unlike print(), the logging module makes it easy to switch between showing and hiding log messages.

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

**-** STEP = it will cause the debugger to execute the next line of code and then pause again. If the next line of code is a function call, the debugger will "step into" that function and jump to the first line of code of that function.

OVER = it will execute the next line of code, similar to the STEP button. However, if the next line of code is a function call, the OVER button will "step over" the code in the function.

OUT= It will cause the debugger to execute lines of code at full speed until it returns from the current function.

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

**-** Continue execution until the current function returns. Continue execution, only stop when a breakpoint is encountered.

**11. What is the concept of a breakpoint?**

**-** A breakpoint can be set on a specific line of code and forces the debugger to pause whenever the program execution reaches that line.