**ASSIGNMENT : 11**

1. Create an assert statement that throws an AssertionError if the variable spam is a negative integer.  
Ans: x = 0

assert x > 0, 'Only positive numbers are allowed'

print('x is a positive number.')

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).  
Ans: assert(eggs. lower() !=

3. Create an assert statement that throws an AssertionError every time.  
Ans: Assertion is a programming concept used while writing a code where the user declares a condition to be true using *assert* statement prior to running the module. If the condition is *True*, the control simply moves to the next line of code. In case if it is *False* the program stops running and returns *AssertionError* Exception.

4. What are the two lines that must be present in your software in order to call logging.debug()?  
Ans:  import logging  
 logging.basicConfig(level=logging.DEBUG, format=' %(asctime)s - %(levelname)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?  
Ans: import logging  
logging.basicConfig (filename='programLog.txt', level=logging.DEBUG, format='%(asctime)s - %(levelname)s - %(message)s')

6. What are the five levels of logging?  
Ans: logging.debug() - variable's state and small details  
logging.info() - general events, confirm a program is working  
logging.warning() - potiental problem to work on in the future  
logging.error() - record an error that caused program to fail to do something  
logging.critical() - fatal error that has caused

7. What line of code would you add to your software to disable all logging messages?  
Ans: logging.disable(logging.DEBUG)

8.Why is using logging messages better than using print() to display the same message?  
Ans: Because with print, when your program is ready for production, you still have to "remove" or comment it out. Verses logging message, you can toggle the setting on/off or write to a file (send to a server). It is more flexible especially with logging level 1-5.

9. What are the differences between the Step Over, Step In, and Step Out buttons in the debugger?  
Ans: Step - one line execution at a time, Over - excecute the next line of code, but if it is a program, it will complete the entire function call. out - execute the lines of code unti it returns from the current function.

10.After you click Continue, when will the debugger stop ?  
Ans: Go runs until the program terminate or reaches a breakpoint set.

11. What is the concept of a breakpoint?  
Ans: When you have Debugger enabled and you can right click on any lines to create a breakpoint. During Go - it will stop there and await your next command.