**ASSIGNMENT 11** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***1. Create an assert statement that throws an AssertionError if the variable spam is a negative integer.**

**Answer 1:**

spam = -100

assert spam > 100 ,' spam variable is negative hence it is not greater than 100 '

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**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).**

**Answer 2:**

def function ( eggs, bacon):

assert eggs.lower() != bacon.lower(), ' both eggs and bacon are same '

def function1 ( eggs, bacon):

assert eggs.lower() == bacon.upper(), ' both eggs and bacon are same

function ( 'hello', 'hello')

function1 ( 'hello', 'HELLO')

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

**Answer 3:**

assert 0

and

assert False

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

**Answer 4:**

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?**

**Answer 5:**

import logging

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

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

**Answer 6:**

on basis of Severity Level, there are 5 levels of debugging

1. Debug : records simple details
2. Info : records general information
3. Warning : record potential issues which may not cause errors in future
4. Error : records errors which causes a section of code to fail
5. Critical : record Highest level , blockers or errors which fail your entire code / program.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

**Answer 7:**

import logging

logging.disable(level=logging.CRITICAL)

logging.disable(level=logging.DEBUG)

logging.disable(level=logging.INFO)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

**Answer 8:**

Logging is built in a library which stores or records the tracking of events that happens during the execution of code or software.It is helpful to programmers to track error or exception. It is helpful in debugging. It is helpful to programmers to monitor , investigate and diagnose code effectively at any time in code and it is time consuming.

Print () function does not store any information. Print function only prints the output. It is used where output is required. It's time Consuming.

Hence for complex programming , Logging is better than print function.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

**Answer 9:**

It is a feature to debug the code step by step to check whether the code runs smoothly without any syntax or logical errors. In case there is a problem or error occurs during the execution of code, we debug the code and fix the issue.

In Debugging mode, there are three buttons or modes.

Step over – An action to take in the debugger that will step over a given line. If the line contains a function the function will be executed and the result returned without debugging each line.

Step into – An action to take in the debugger. If the line does not contain a function it behaves the same as “step over” but if it does the debugger will enter the called function and continue line-by-line debugging there.

Step out – An action to take in the debugger that returns to the line where the current function was called.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

**Answer 10:**

After clicking the Continue, It will continue the execution of the code till the next breakpoint or exit of the code.

At breakpoint debugger stops execution of code. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

**Answer 11:**

Breakpoint is a feature of debugger. It stops the execution of code at that point.

When breakpoint in code that is marked to a specific line of code. During debugging, it halts the execution at that line . \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*