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

Answer :

spam=int(input("Enter a Positive Integer))

assert spam > 0, "Only Positive Integers are allowed"

print("Integer is positive", spam)

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:

eggs= input("Enter a string")

bacon = input("Enter a string")

assert eggs.lower() =bacon.lower() , "Same string is inputted both times"

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

Answer : assert False, "AssertionError everytime".

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

Answer :

import logging

logging.basicConfig(level=logging.DEBUG, format='%(asctime)s-%(levelname)s-%(message)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?

Answer:

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?

Answer: DEBUG, INFO, WARNING, ERROR and CRITICAL are five levels of logging.

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

Answer: logging.disable(logging.CRITICAL)

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

Answer: print() can be used for debugging but it will take lot of time removing print() statements, one might also end up deleting a useful print () statement. Instead logging message can be used many number of times and can be disabled by using a single logging.disable(logging.CRITICAL) critical.

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

Answer : Step Over runs into the next statement without stepping into function or methods.

Step In runs the next statement. If the current line contains a function call, **Step Into** steps into the most deeply nested function.

Step Out advances the debugger all the way through the current function. **Step Out** continues running code and suspends execution when the current function returns.

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

Answer: Continue execution only stops when breakpoint is encountered.

11. What is the concept of a breakpoint?

Answer : Breakpoint is a specific point in line of code that forces the debugger to pause whenever that program execution reaches that line.