**ADITYA AMIN ASSIGN : 11**

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

spam = -5 # Example value of spam

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

In this example, the assert statement checks if the value of "spam" is greater than or equal to 0. If it's not, it will raise an AssertionError with the given error message "spam cannot be a negative integer". The AssertionError will be raised and the program will terminate with an error message indicating that the assertion has failed, if the value of "spam" is a 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).

assert eggs.lower() != bacon.lower(), "eggs and bacon cannot be the same (case-insensitive)"

Explanation:

eggs.lower() and bacon.lower() convert the strings in the eggs and bacon variables to lowercase, respectively.

!= compares the lowercase versions of the strings to check for inequality, even if their cases are different.

If eggs and bacon contain the same string (considering case-insensitive comparison), the assert statement will trigger an AssertionError with the specified error message: "eggs and bacon cannot be the same (case-insensitive)".

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

assert False, "This is an example of an AssertionError"

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

import logging

logging.basicConfig(level=logging.DEBUG)

1. 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 logging

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

1. What are the five levels of logging?

1.DEBUG: Lowest severity level. Used for detailed diagnostic information that can be useful for debugging and troubleshooting during development. Typically not enabled in production code.

2.INFO: Provides information about normal program operation. Can be used to convey important program milestones or general information about the program's progress.

3.WARNING: Indicates a potential issue or warning that does not necessarily result in an error. Typically used to notify about non-critical issues that may affect the program's functionality.

4.ERROR: Indicates an error that may cause the program to malfunction or fail. Typically used for reporting critical errors that require immediate attention.

5.CRITICAL: Highest severity level. Indicates a critical error that may cause the program to crash or become unusable. Typically used for reporting severe errors that require urgent attention.

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

Import logging

logging.disable(logging.CRITICAL)

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

using logging messages with the logging module in Python provides greater flexibility, configurability, granularity, runtime control, performance, and separation of concerns compared to using print() statements

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

"Step Over" allows you to execute the current line and move to the next line without stepping into function calls, "Step In" allows you to step into function calls and inspect their internal execution, and "Step Out" allows you to quickly return to the caller's context after stepping into a function call. These buttons provide control over the flow of execution during debugging sessions, allowing you to navigate through code and inspect its behavior at different levels of detail.

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

If a breakpoint is set at a particular line of code,If an exception is raised during the execution of the code,If the program reaches its end and completes execution without encountering any breakpoints or exceptions.

1. What is the concept of a breakpoint?

A breakpoint is a debugging feature in software development that allows a programmer to pause the execution of a program at a specific line of code during runtime, allowing them to inspect the state of the program at that point and debug any issues. Breakpoints are commonly used in integrated development environments (IDEs) and debuggers to help developers understand the behavior of their code and identify and fix bugs.