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

Solution1:-

try:

spam=int(input('enter a positive number '))

assert spam>0

print(spam, ' is a positive number ')

except AssertionError:

print('it 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).

Solution2:-

try:

eggs=input('enter a srting ')

bacon=input('enter a srting')

e=eggs.capitalize()

b=bacon.capitalize()

assert e==b

print(eggs, " & ", bacon, " are same")

except AssertionError:

print('kindly enter same string')

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

Solution:-

try:

spam=input('enter your data ')

assert False

except AssertionError:

print('kindly enter your data ')

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

Solution4:-

import logging

logging.basicConfig(level=logging.DEBUG, format=' %(asctime)s -%(levelname)s - %(message)s',datefmt='%y-%m-%d %hh:%mm:%ss')

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?

Solution5:-

import logging

logging.basicConfig(filename='programLog.txt', level=logging.DEBUG,format=' %(asctime)s - %(levelname)s - %(message)s',datefmt='%y-%m-%d %hh:%mm:%ss')

1. What are the five levels of logging?

Solution6:-

DEBUG, INFO, WARNING, ERROR, and CRITICAL

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

Solution7:

logging.disable(logging.CRITICAL)

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

Solution8:-

The logging package has a lot of useful features:

Easy to see where and when (even what line no.) a logging call is being made from.

You can log to files, sockets, pretty much anything, all at the same time.

You can differentiate your logging based on severity.

Print doesn't have any of these.

Also, if your project is meant to be imported by other python tools, it's bad practice for your package to print things to stdout, since the user likely won't know where the print messages are coming from. With logging, users of your package can choose whether or not they want to propogate logging messages from your tool or not.

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

Solution9:-

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 ?

Solution10:-

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

11. What is the concept of a breakpoint?

Solution11:-

The Python breakpoint() built-in function is a tool that allows developers to set points in code at which a debugger is called. By default, this function results in an instantiation of Python's native debugger class.