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

Answer:

spam = -5

assert spam>0

print('need to be 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).

Answer:

eggs = input('enter your string')

bacon = input('enter your 2nd string')

assert eggs.lower() != bacon.lower(),'Need to contain different strings'

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

Answer:

n = int(input('Enter your value'))

#n = input('Enter your value')

assert type(n)==str,'Data types donot match'

print('Condition passed')

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

Answer: The following lines should be present in order to use logging.debug()

Import logging

Logging.basicConfig(filename)

Logger = logging.getLogger()

Logger.setLevel(logging.DEBUG)

Logger.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: import logging

Logging.BasicConfig(filename = ‘programLog.txt’)

Logger = logging.getLogger()

Logger.setLevel(logging.DEBUG)

Logger.info(‘Just an information’)

6. What are the five levels of logging?

Answer: There are 6 levels of logging namely NOTSET, DEBUG, INFO, WARNING, ERROR, CRITICAL

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

Answer: We would use logger.propagate = False

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

Answer: Logging messages is better than print because you can never use print statements for large scripts for debugging and logging purposes specially when you are working in modules.

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

Answer:

* 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.  
  http://fourkitchens.com/wp-content/uploads/2017/01/Screen%20Shot%202013-12-04%20at%209.54.54%20PM.png
* 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.  
  http://fourkitchens.com/wp-content/uploads/2017/01/Screen%20Shot%202013-12-04%20at%209.54.58%20PM.png
* 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: In this case , debugger will only stop once it has encountered an break point

11. What is the concept of a breakpoint?

Answer: A breakpoint is an intentional stopping or a pausing place in a program to check the flow of our code for debugging purposes.