Python_assignment11

1) spam=-33
```
 assert spam >=0 , "Spam variable can't take -ve integer."
```

```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
<ipython-input-3-442197730030> in <cell line: 2>()
      1 spam=-33
----> 2 assert spam >=0 , "Spam variable can't take -ve integer."

AssertionError: Spam variable can't take -ve integer.
```

2) egg="ineuron"
```
bacon="INeuron"


assert egg==bacon , "Values should be same"
```

```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
<ipython-input-6-4b30c46c182d> in <cell line: 4>()
      2 bacon="INeuron"
      3
----> 4 assert egg==bacon , "Values should be same"
      5

AssertionError: Values should be same
```

3) def assert_always():
```
assert False, 'Always Shows Assertion Error'
assert_always()
```

```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
<ipython-input-7-b25c0da8a33f> in <cell line: 3>()
      1 def assert_always():
      2     assert False, 'Always Shows Assertion Error'
----> 3 assert_always()

<ipython-input-7-b25c0da8a33f> in assert_always()
      1 def assert_always():
----> 2     assert False, 'Always Shows Assertion Error'
      3 assert_always()

AssertionError: Always Shows Assertion Error
```

4) The two line present in our software in order to call logging.debug() are
```
   import logging
logging.basicConfig(filename = 'application_log.txt',level=logging.DEBUG, format='
%(asctime)s - %(levelname)s - %(message)s')
```

5)The two lines that your program must have in order to have logging.debug() send a logging message to a file named programLog.txt are
```
import logging
logging.basicConfig(filename = 'application_log.txt',level=logging.DEBUG, format=' %(asctime)s - %(levelname)s - %
(message)s')
logging.debug("Data Inserted Successfully")
logging.debug('Connection Closed Successfully')
file = open("./application_log.txt","r")
for record in file.readlines():
    print(record)
```

6) The Five levels of Logging provided by Python's Logging Module are CRITICAL(50), ERROR(40), WARNING(30), INFO(20, DEBUG(10), NOTSET(0).

7) The code to disable all login messages is logging.disable== True

8) Post devlopment of our code, we can disable logging messages without removing the logging function, whereas we need to manually remove print() statements, which would become a tedious activity. Also, print is used when we want to display any particular message or help whereas logging is used to record all events such as error, info, debug messages, timestamps.

9) Step in - Step In button will cause the debugger to execute the next line of code and then pause again.

Step Over - Step Over button will execute the next line of code, similar to the Step In button. However, if the next line of code is a function call, the Step Over button will “step over” the code in the function. The function’s code will be executed at full speed, and the debugger will pause as soon as the function call returns.

Step out - Step Out button will cause the debugger to execute lines of code at full speed until it returns from the current function.

10) After you click Continue, This will cause the program to continue running normally, without pausing for debugging untill it terminates or reaches a breakpoint.

11) Breakpoint is a setting on a line of code that causes the debugger to pause when the program execution reaches the line.