## L07 Participation Assignment – Simple Class

### **Background**

Implement a simple class

### **Procedure**

Answer the following questions with cut and paste or screenshots.

1. Implement a Counter class with methods to reset value, increment by 1 and a variable to store the values.
2. Create a class called Counter.

Class Counter:

1. An object stores its data in instance variables. Use \_value to store the counter value. If an instance variable starts with an underscore, it should be private which means it should only be accessed by method of the same class however it is not strictly enforced. Define the getValue method that returns the current \_value.

def getValue(self) :

return self.\_value

1. Define a method to increment the counter by 1.

def click(self) :

self.\_value = self.\_value + 1

1. Define a method to reset the value of the counter to 0.

def reset(self) :

self.\_value = 0

1. Save this to a module named counter.py
2. Create a new module named counter\_demo.py
3. Start by importing in the class.

from counter import Counter

1. Create an instance of the class called tally

tally = Counter()

1. Call the reset method of the tally object

tally.reset()

1. Call the click method several times (ie repeat the call)

tally.click()

tally.click()

1. Assign the counter value to result and print.

result = tally.getValue()

print("Value:", result)

1. Run the application counterdemo.py.
2. **Copy and paste the output or screenshot below**:
3. Modify the Counter class by implementing a private limit method to check if clicks exceed the target.

First initialize, then set limit-----

def \_\_init\_\_(self):  
 self.\_limit = **0**

def setLimit(self, maximum) :

self.\_limit = maximum

1. Modify and add to the click method to check if the limit has been exceeded. If so notify the user.

if self.\_value > self.\_limit :

print("Limit Exceeded")

1. Add two more tally.click() method calls to counterdemo.py and getValue() again.

result = tally.getValue()

print("Value:", result)

In counter\_demo.py, pass in the maximim limit value with

tally.setlimit(**2**)

1. **Copy and paste the output or screenshot below**:

A screenshot of a computer program

Description automatically generated

1. Think about what would happen if you didn’t call reset() immediately after constructing the tally object. Would it make a difference?
2. Modify the implementation of the counter and change from an integer counter to adding string characters to keep track of the clicks. For instance:

def reset(self) :

self.\_strokes = ""

def click(self) :

self.\_strokes = self.\_strokes + '|'

1. Also modify getValues to work with the revised implementation.
2. class Counter:
3. def \_\_init\_\_(self):
4. self.\_strokes = ""
5. def getValue(self):
6. return self.\_strokes
7. def click(self):
8. if len(self.\_strokes) > self.\_limit:
9. print("Limit Exceeded")
10. else:
11. self.\_strokes += '|'
12. def reset(self):
13. self.\_strokes = ""
15. def setLimit(self, maximum):
16. self.\_limit = maximum
17. Rerun application and **Copy and paste the output or screenshot below**:

A screen shot of a computer

Description automatically generated

1. Submit this word document to Canvas. Use copy and paste or screenshots for the answers. Questions should be included to use as a study guide.