Theater Seating

LAB # 5

By

Corey Henry

And

Natalie Morrison

***“On my honor, as a Mississippi State University student, I have neither***

***given nor received unauthorized assistance on this academic work.”***

Signature:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

CSE-1384-06-201430 Intermediate Computer Programming

Class Section # 6

Jesse Farek

2/17/2015

**Analysis and Conclusions**

The point of this lab was to create a empty theater and be able to sell a seat, you had to keep track of what was sold and not be able to sell a sold seat. You had to take the size of the theater by a variable provided for you in a driver. This lab was very Similar to the turtle class, it worked almost the same except you had to modify your list editing. The printing of the theater seats was very different though and the most difficult part of the program. We used row and seat as row and column to keep track of all the seats. First starting off with a empty theater and writing the ‘-‘ to all the seats, to reset the theater we just copy and pasted the init so it would rewrite everything without carrying over the sold seats. To print we created a empty string and used the inner list function to add to the string so it could be returned to be printed as a string.

Analysis Questions

1. I would use a deep copy to return a list of the theater, that way if you started to sell seats and there was confusion and you needed a old copy deep copy would ensure nothing had been changed. Also It would allow you to do two theaters at the same time without having to create a new object or the class. It would also allow customers to change their minds and not want their seats.

Source Code:

#Corey Henry & Natalie Morrison #Date Assigned: 17Feb15

# #

#Course CSE 1384 Sec 06 #Date Due: 24 Feb 15

#File name: Theater.py

#

#Program description- creates a theater seat selling systen

#Initialize row and seat to default of 1 and create a intial 2D list

class Theater():

def \_\_init\_\_(self,row =1,seat= 1):

self.row = row

self.seat = seat

self.size = []

for each in range(row):

self.size +=[['-']\* seat]

#Create a 2D list that overrides the old list

def set\_size(self, row, seat):

self.row = row

self.seat = seat

self.size = []

for each in range(row):

self.size += [['-']\* seat]

#This function takes the row and seat and replaces the '-' with '#' if open

def sell\_seat(self,row,seat):

row\_number = row -1

seat\_number = seat -1

#if seat is not available, return false

if self.size[row\_number][seat\_number] == '#':

return(False)

#seat is sold, return true

else:

self.size[row\_number][seat\_number] = '#'

return(True)

#function that returns the grid to print

def \_\_str\_\_(self):

#create a empty string to return

|  |
| --- |
|  |

new = ''

#double loop to take the index of the row and seat and add it to the string

for row in self.size:

for seat in row:

new += seat

new += '\n'

return new

|  |
| --- |
| Theater |
| Self.row  Self.size  Self.seat |
| \_\_init\_\_(self, row = 1, seat = 1)  set\_size(self, row, seat)  sell\_seat(self, row, seat)  \_\_str\_\_(self) |