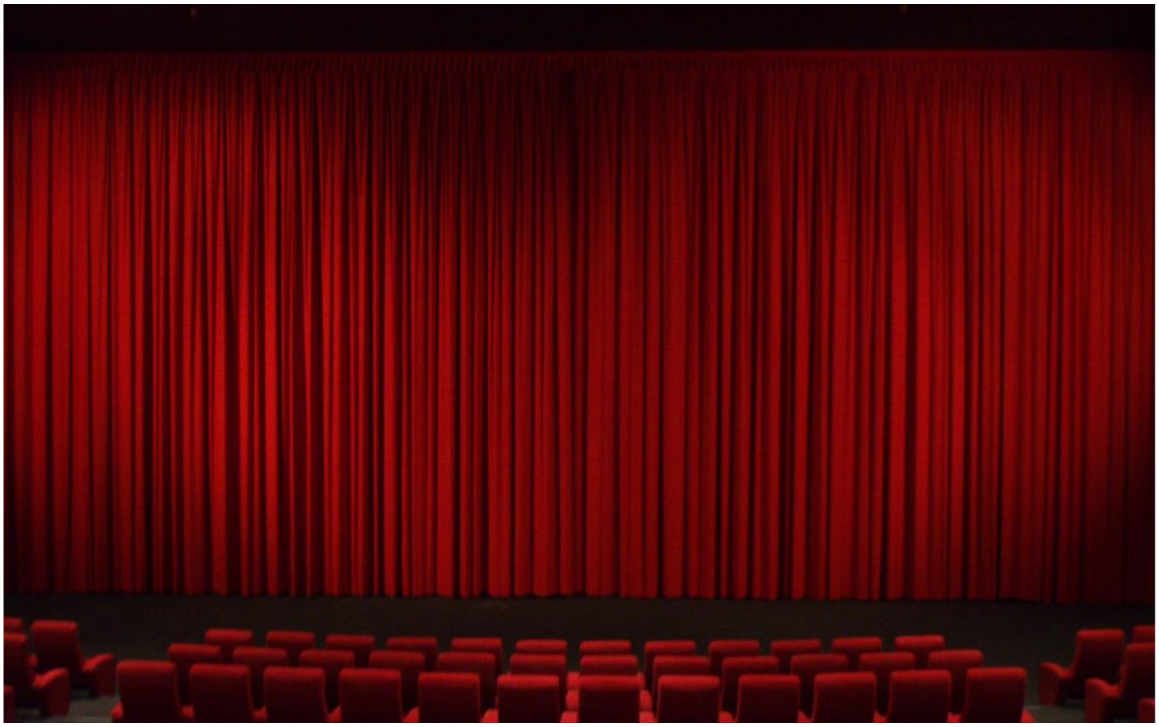
Let’s Go to a Show!



picture from http://www.centuryvillagecondoforsale.com/?page_id=17

Objectives:

- To practice writing classes
- To gain more experience with nested lists
- To gain a better understanding of the python memory model

Background:

This week we’ll be creating a class that will help with selling tickets in a theater. We will represent an empty seat with a ‘-’(hyphen) and one that has been sold with ‘#’. A two dimensional list will be the only instance data that will be needed.

Assignment:

Create a class named Theater. It should be saved in a module named Theater.py. You will need to write the following member methods for it:

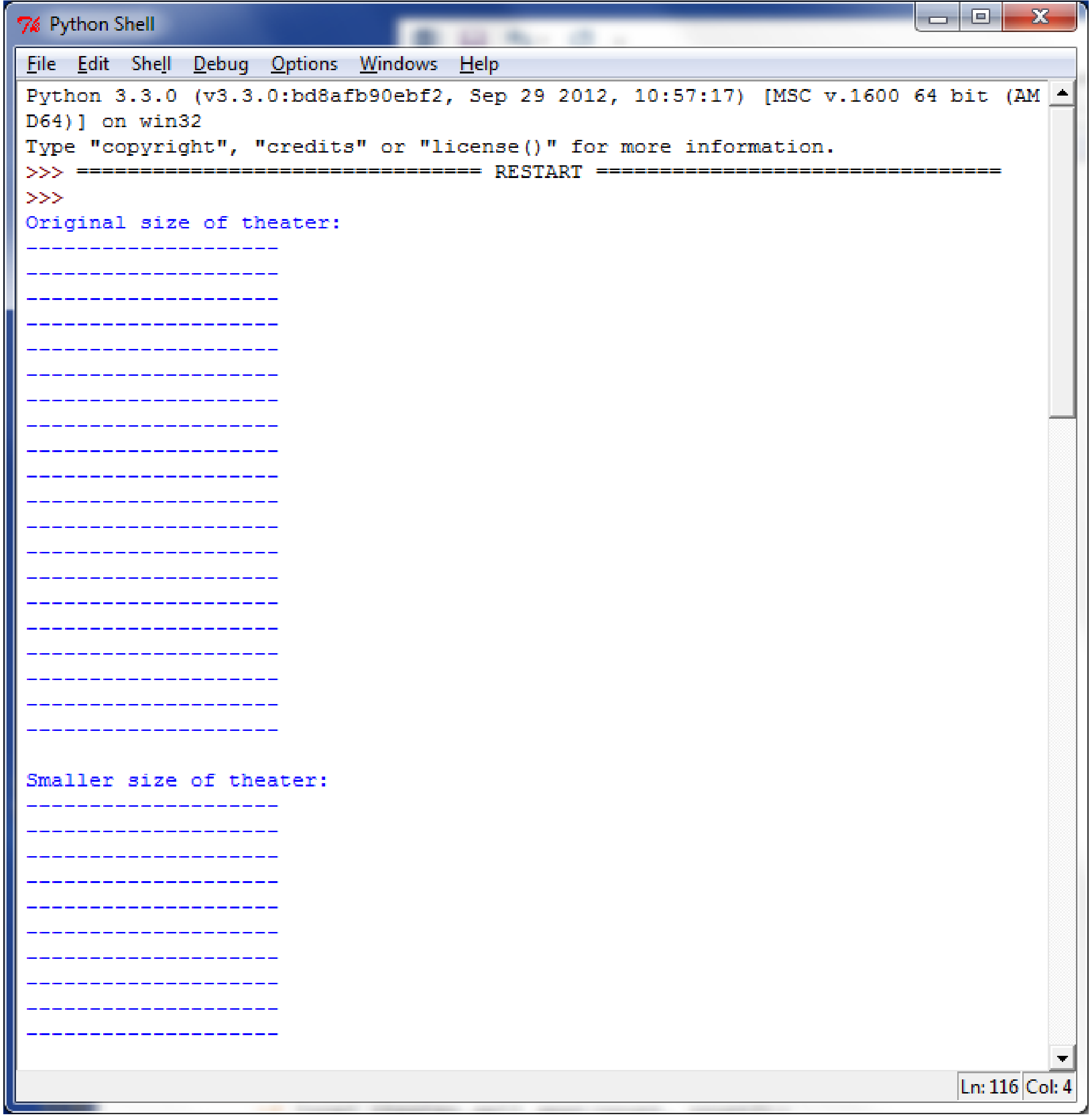
__init__():	This method may take **two** arguments representing the number of rows in the theater and the number of seats in each row. It initializes theater seats, storing a ‘-’(hyphen) in each seat. If it doesn’t get any arguments, then the size is 1 row and 1 seat on the row by default.

__str__():	This method is called to when an object of the class is printed. It should return a string that contains all of the symbols stored in the two dimensional list, starting with the first row. Be sure to add the '\n' characters at the end of each row.
set_size():	This method allows the programmer to change the size of the list, stored in the theater module. All seats already sold will be lost if the size is changed after selling seats. It modifies the two dimensional list representing the theater. Each seat should hold a '-'(hyphen) sign
sell_seat():	This method allows the programmer to sell a seat. It takes two arguments: **the row number** and **seat number**. Remember that users tend to start counting at 1 (not 0). Take that into account when marking a seat as sold. Be sure to check that the seat hasn't already been sold before marking it sold. The method returns **True** if the seat wasn't already sold (and marks it sold with a #) and **False** if the seat had already been sold.

Run the driver program with your class. Once your code works with the class, demonstrate that for your TA.

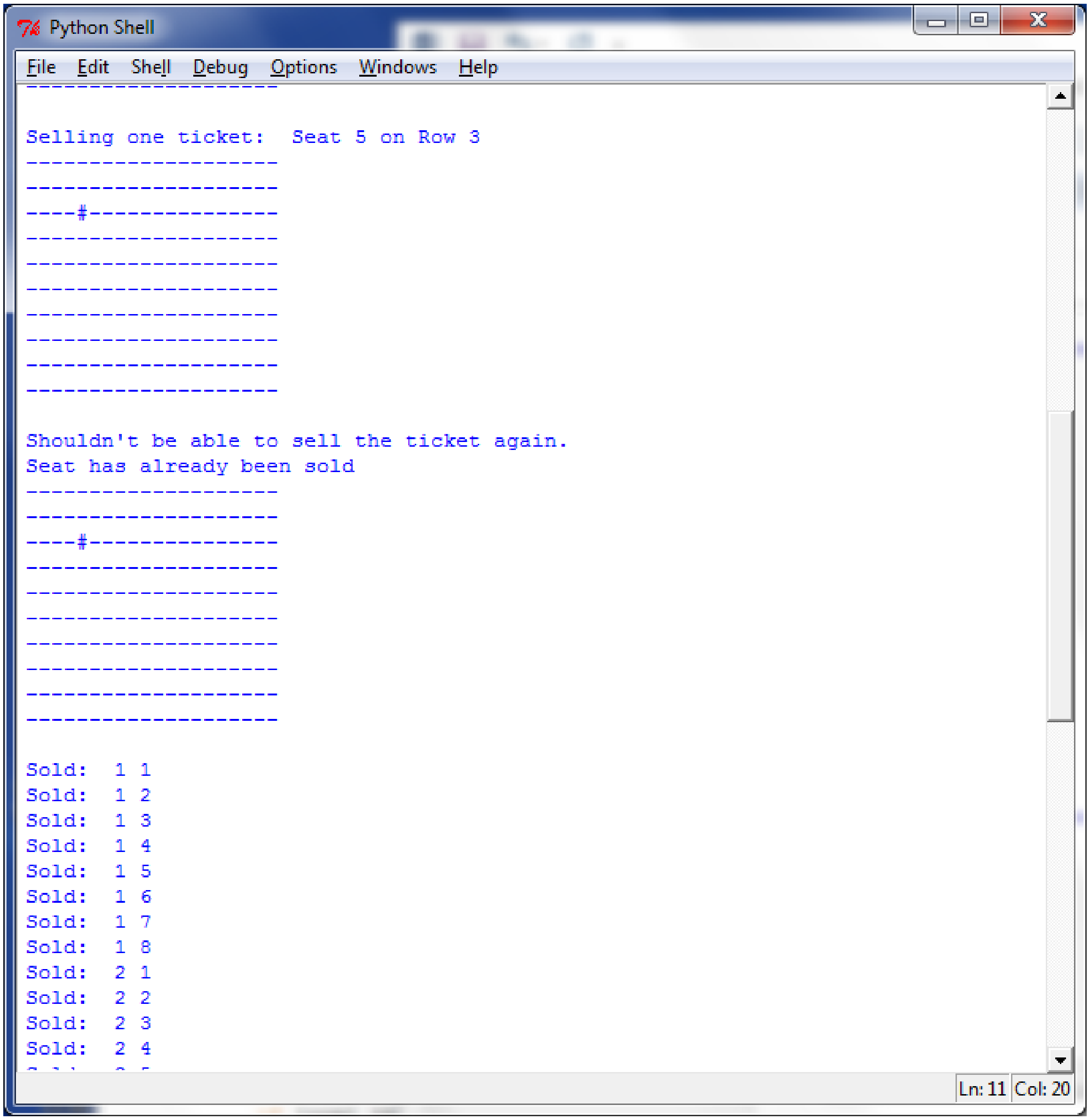
Analysis Questions:

Sample Execution:



```
Python 3.3.0 (v3.3.0:bd8afb90ebf2, Sep 29 2012, 10:57:17) [MSC v.1600 64 bit (AM
D64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>> 
Original size of theater:
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------

Smaller size of theater:
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
```

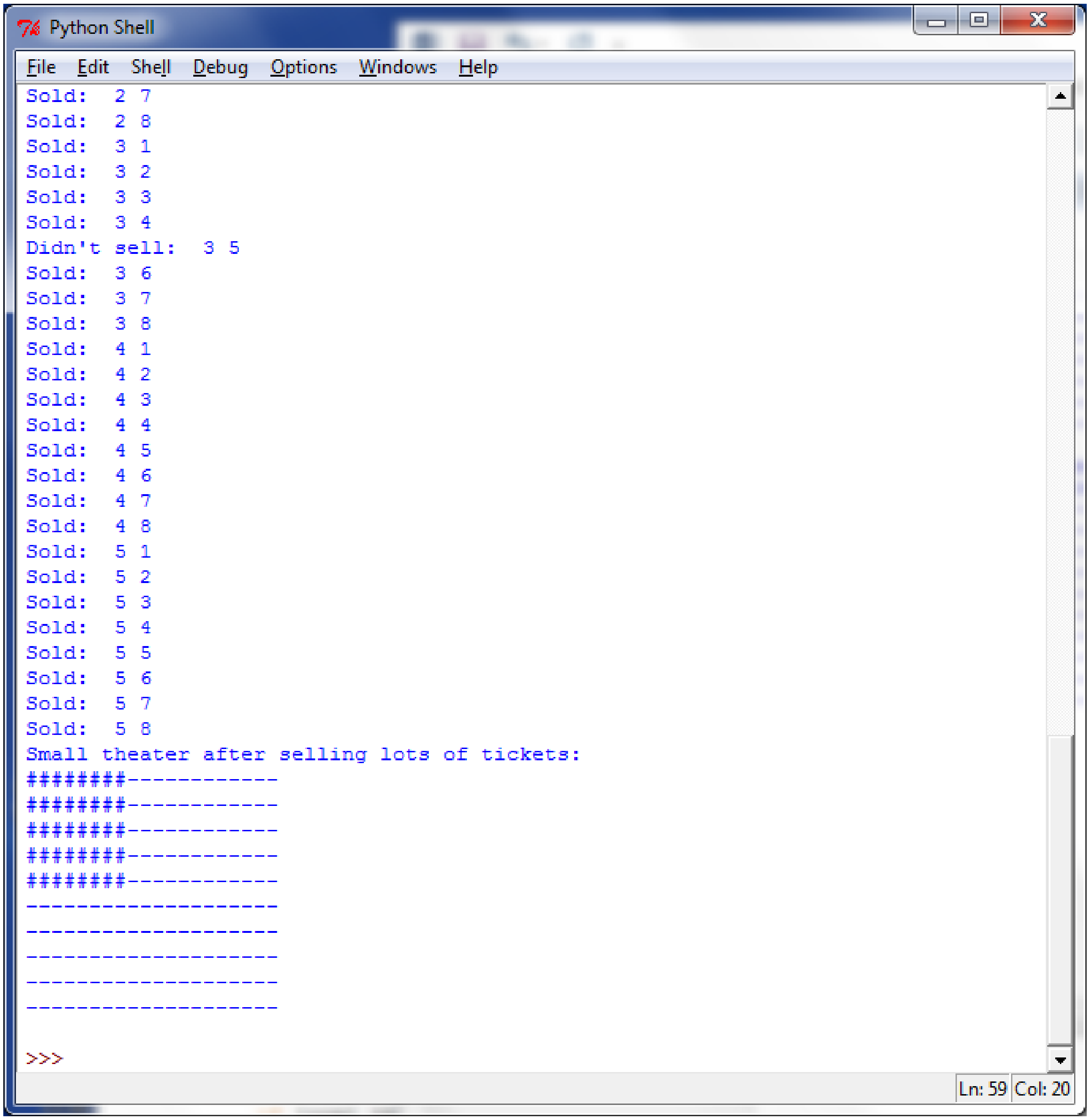


```
--------------------

Selling one ticket:  Seat 5 on Row 3
--------------------
--------------------
----#---------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------

Shouldn't be able to sell the ticket again.
Seat has already been sold
--------------------
--------------------
----#---------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------

Sold:  1 1
Sold:  1 2
Sold:  1 3
Sold:  1 4
Sold:  1 5
Sold:  1 6
Sold:  1 7
Sold:  1 8
Sold:  2 1
Sold:  2 2
Sold:  2 3
Sold:  2 4
Sold:  2 5
```



```
Sold:  2 7
Sold:  2 8
Sold:  3 1
Sold:  3 2
Sold:  3 3
Sold:  3 4
Didn't sell:  3 5
Sold:  3 6
Sold:  3 7
Sold:  3 8
Sold:  4 1
Sold:  4 2
Sold:  4 3
Sold:  4 4
Sold:  4 5
Sold:  4 6
Sold:  4 7
Sold:  4 8
Sold:  5 1
Sold:  5 2
Sold:  5 3
Sold:  5 4
Sold:  5 5
Sold:  5 6
Sold:  5 7
Sold:  5 8
Small theater after selling lots of tickets:
########------------
########------------
########------------
########------------
########------------
--------------------
--------------------
--------------------
--------------------
--------------------

>>>
```

Program Requirements:

For the code:

- Since this is a paired assignment, your program should have the following comment block at the top.

```
#
#Names:        Name of one partner          Name of other partner
#Date Assigned:  Date of lab                 Date Submitted:  Date submitted
#Course:       CSE 1384                      Lab Section:  Your lab section
#
#File name:  lastName.py  (of the person submitting file in myCourses)
#
#Program Description:  A short description of what the program is
#supposed to accomplish.
#
```

- Use appropriate comments throughout the code. Be sure to include comments before each class definition and before each method (member function) explaining the interface (what must be sent, what is returned) and the purpose of the method. Also, don't forget to make appropriate comments within the method definition.

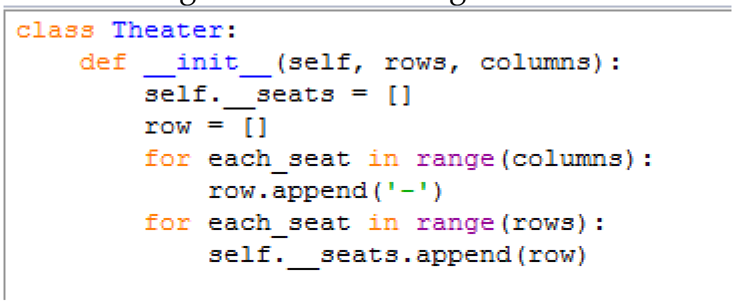
- Make good use of whitespace. Be sure to include a blank line of space before each comment. Two lines of space between functions is nice, but one will suffice – whichever you choose, be consistent. Follow the Python programming style guide: https://www.python.org/dev/peps/pep-0008/#introduction

- Don't forget to use good **descriptive** variable names

For the lab report, follow the Lab Report Format Guide and complete the following sections:

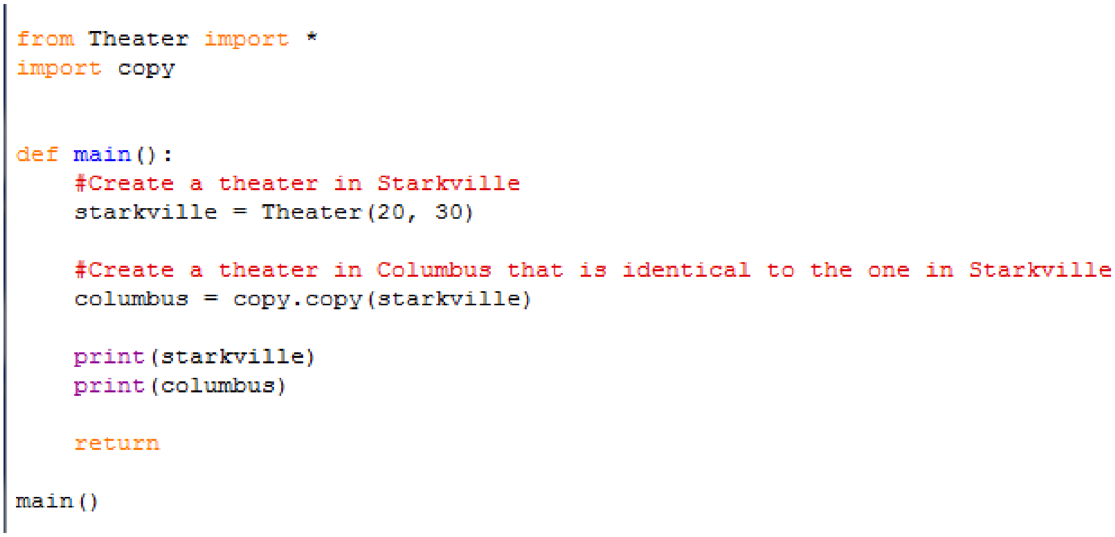
- Title Page
- Design – Create the UML diagram for your Theater class.
- Analysis and Conclusions
- Extra Analysis Questions:
    - If you included a method named backup that returned a copy of the theater, should you use an alias, a shallow copy, or a deep copy? Explain your answer.

- What is wrong with the following code? How would you fix it?



```
class Theater:
    def __init__(self, rows, columns):
        self.__seats = []
        row = []
        for each_seat in range(columns):
            row.append('-')
        for each_seat in range(rows):
            self.__seats.append(row)
```

- What is wrong with the following code? How would you fix it?
  Hint: What happens when you try to sell a seat?



```
from Theater import *
import copy


def main():
    #Create a theater in Starkville
    starkville = Theater(20, 30)

    #Create a theater in Columbus that is identical to the one in Starkville
    columbus = copy.copy(starkville)

    print(starkville)
    print(columbus)

    return

main()
```

- Appendix B - code (Be sure that you copy and paste the code into your Word document, don't take a screen shot of it and paste that – it ends up being too hard to read.) **Be sure that both partners have a copy of the code before leaving lab.**

Deliverables:

Electronic submission in myCourses:

- Code (due by the end of the class period) submitted by **one** of the partners, not both
- Lab Report – to be done individually (due by the start of the next time that your lab meets) -- either a Microsoft Word document or a pdf

Paper submission in class:

- Lab Report (due at the start of lab the next time that it meets)

Grading:

Task	**Points**
Lab Report	60 points
Title Page	5 points
Design (UML diagrams)	5 points
Analysis and Conclusions	10 points
Extra Analysis Questions	30 points
Appendix B – coding style points (see requirements section)	10 points
Program	40 points
__init__() and set_size ()	15 points
sell_seat()	15 points
__str__()	10 points