CSE 1321L: Programming and Problem Solving I Lab

Fall 2025

Lab 11

Module 5

What students will learn

- Design classes including attributes, constructors, and methods
- Creating objects of classes

Content

- Overview
- Assignment 11A: The Architect

Overview

For this assignment, you are going to practice implementing and designing classes defining its attributes and behaviors by implementing a constructor function and other member functions.

Final note: *Do not cheat*
If your temptation is to look online, don't. Come see us instead and ask questions – we are here to help. Remember, you are going to have to write code in your future job interviews, so learn it now to secure a high-paying job later.

Lastly, make sure you review the sample output and make sure the output of your program follows the exact same format including the input statements, print statement, etc. As always, user input is shown in **red** and **bold**.

Lab 11A: The Architect

Buildings can be built in many ways. Usually, the architect of the building draws up maps and schematics of a building specifying the building's characteristics such as how tall it is, how many stories it has, etc. Then the actual building itself is built based on schematics (also known as blueprints).

Now it is safe to assume that the actual building is based off the blueprint but is not the blueprint itself (and vice versa).

The idea of classes and objects follows a similar ideology. The class can be considered the Blueprint and the objects are the building following the analogy mentioned above. The class contains the details of the object, i.e., the object's attributes (variables) and behavior (methods).

Please keep in mind that a class is a template of an eventual object. Although the class has variables, these variables lack an assigned value since each object will have a unique value for that variable. Think of a form that you may fill out, for example, for a bank; this form has many boxes such as ones for your first name, last name, etc. That form is analogous to a class; it's generic and does not have any unique information. Once someone picks up the form and fills it out it becomes unique to that person and is no longer a generic form; this is analogous to an object.

For the one and only exercise in Lab 13, you will need to design a class called **BuildingBlueprint** and create objects from this class in your main program. Please read and follow the instructions below carefully

Requirements:

- **BuildingBlueprint** class:
    - The constructor function should take in 3 parameters, a parameter for the number of stories, number of apartments, and the occupancy rate.
    - The constructor function should also define optional values for these parameters:
        - Number of stories: 10
        - Number of apartments: 20
        - Occupancy rate: 1.0 (100%)
    - The constructor should set the instance attributes for number of stories, number of apartments, and occupancy rate accordingly.
    - Moreover, the constructor should also define another instance attribute which will be a Boolean that tracks whether the building is fully occupied.
        - The value of this Boolean will be based on the occupancy rate: If the occupancy rate is 1.0 (100%) then this value should be True, while if the occupancy rate is less than 1.0 (> 100%) then will be set as False.
    - Lastly, the class should have a function called **update_occupancy()** which takes one parameter, which will be used to change the object's occupancy rate.
        - This function should also update the Boolean attribute that tracks if the building is fully occupied or not.
- **Main** Program:
    - In your program, create two **BuildingBlueprint** objects, **buildingOne** and **buildingTwo**.

- **buildingOne** will be created using the default parameter values.
- **buildingTwo** will be created with these arguments:
  - 30 Stories
  - 30 Apartments
  - 0.75 Occupancy rate
- The program should then print out the information of **buildingOne** and **buildingTwo** (view sample output).
- Call the **update_occupancy()** function to change occupancy rate of **buildingOne** to 0.0
- Call the **update_occupancy()** function to change occupancy rate of **buildingTwo** to 1.0
- Print out the information of **buildingOne** and **buildingTwo** (view sample output).

Sample Output #1:

```
Year 2025:
Building 1 has 10 floors, 20 apartments, and is 100% occupied. Full? True
Building 2 has 30 floors, 30 apartments, and is 75% occupied. Full? False

Many years passed

Building 1 has 10 floors, 20 apartments, and is 0% occupied. Full? False
Building 2 has 30 floors, 30 apartments, and is 100% occupied. Full? True
Looks like people prefer taller buildings.
```

Submission Instructions:

- Programs must follow the output format provided. This includes each blank line, colons (:), and other symbols.
- Programs must be working correctly.
- Programs must be written in Python.
- Programs must be submitted with the correct **.py** format.
- Programs must be saved in files with the correct file name:
  - Lab11A.py
- Programs (source code files) must be uploaded to Gradescope by the due date.