# CS166 Assignment 1
# Elevator simulation

## Introduction

In this group assignment, you complete the implementation of an elevator simulation that we start in class during the first week of the course. You will experiment with different strategies for scheduling an elevator based on passenger demand and measure how well the different strategies work.

**The purpose** of this assignment is to learn how to use object-oriented programming in Python to implement a simulation. You will represent different real-world entities as Python classes that interact with each other through class attributes and methods. As a secondary objective, you gather and interpret efficiency results of different strategies for moving the elevator up and down the building to transport passengers.

**Your task** is to design and implement a simulation of a building containing a single elevator and some number of passengers who want to travel from one floor in the building to another. You should implement at least two strategies (of which one strategy can be the example strategy below) and compare them based on which strategy is most efficient at transporting passengers around the building.

**Example strategy.** The elevator starts on the ground floor and moves all the way to the top floor, stopping at every floor in between. When the elevator reaches the top floor, it changes direction and moves all the way back down to the ground floor, again stopping at every floor in between. At every floor where the elevator stops any passengers who want to get off leave and any passengers who want to get on enter, as long as there is space in the elevator. If the elevator is full, passengers on that floor have to wait. Upon reaching the ground floor, the elevator repeats the cycle of moving all the way up and all the way down the building.

You should attempt to improve on this simple example strategy. You will not get graded on the efficiency of your strategies but on the quality of your code and interpretation of your results. Do finding a more efficient strategy. Just make sure that you have at least two different strategies implemented so you can compare them.

## Simulation specification

- Implement three Python classes — Building, Elevator, Passenger.
- The building has 1 elevator and a user-defined number of floors and passengers.
- The elevator has a user-defined capacity — that is the maximum number of passengers that can be in the elevator simultaneously.
- Each passenger starts on a random floor in the building and wants to go to a different random floor.
- Each passenger uses the elevator only once, and the simulation ends once all passengers have reached their destination floors.
- In your main code block, create the building and run the simulation by repeatedly updating the state of the passengers, elevator, and building in a loop. Check whether all passengers have arrived at their destinations to determine when to stop the loop.
- Display the state of your simulation at regular intervals so you can monitor whether everything is working as you expect. Are the right number of passengers getting on and off the elevator? Does your simulation end or does it get stuck in an infinite loop for some reason? How many passengers have arrived at their destination? Use the answers to these questions to monitor the correctness of your simulation.
- Part of your grade will be for the quality and readability of your code. Be sure to write useful comments that explain what each class method does and what different variables or input arguments represent. Use good variable names! Single letter names are not descriptive so, for example, use “passenger” rather than “p” as a variable, attribute, or method name.

## Group work

- Devise one or more elevator strategies (in addition to the example strategy above).
- Determine how you will quantify the efficiency of a strategy.
- Implement the simulation in Python and help each other understand how to use object-oriented programming to implement this simulation. If you are already a strong Python programmer, help other students understand how to use Python classes and objects to represent the state and interactions of the simulation. If you are relatively new to Python programming, **use this project as an opportunity to improve your Python skills** and ask each other questions so that everyone can learn. You will be able to learn faster as a group during the rest of the course once everyone is comfortable with Python programming.
- **You may not use global variables** to store any part of the state of the building, elevator, or passengers. Global variables lead to all sorts of problems and you should practice working without them from the beginning.
- **Comment your code** to make it more readable to everyone in your group and also to the grader. Use inline comments and doc strings, as appropriate. You will not be able to score more than a 3 on the #pythonimplementation learning outcome without writing readable, documented code. Just in case it is not clear yet, readability matters a lot.
- Test your different elevator strategies, and generate results to determine which strategy is the best.
- All of your code and results must be in a shared Python notebook (for example, on CoCalc) that you will upload as part of your assignment.

## Individual work

- Write a neatly typed report in your own words.
- Include the names of all your group members in your report.
- Explain how your group’s elevator strategies work.
- Explain how you measure the efficiency of a strategy.
- Present your efficiency results.
- Explain what your results mean about the different elevator strategies and how efficient they are at transporting passengers around the building. Are there any assumptions or other considerations that affect the interpretation of these results?
- Write a summary of your main contributions to the project (for example, parts of the code or code documentation you wrote, contributions to designing elevator strategies, organizing the group, helping others learn Python).
- Write a summary of the main things you learned about implementing a simulation in Python.

**Please typeset your work and submit your individual report as a PDF file. Also, submit a Jupyter notebook containing the Python code for your group — including any output (text or plots) generated by running the code.** Make sure you submit a neat, clearly presented, and easy-to-read report. Also remember to comment your code, with the goal of making it easier to understand what the purpose of each part of the code is. You will get a #professionalism HC score on every assignment in this course.