\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**CSC121 PYTHON Programming**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

LAB 13 **OBJECT-ORIENTD PROGRAMING**

# Objectives

In this lab assignment, students will learn:

- How to design objects

- How to define classes

- How to create and use objects

- How to control accessibility to class members

# Goals

In this lab assignment, students will demonstrate the abilities to:

- design objects

- define classes

- create and use objects

- control accessibility to class members

# Instruction and Problems

Create a Python project for each of the following problems. Zip each Python project into a zip file. Submit the zip files to Blackboard for credit.

## Problem 1

Create a Python project fly\_drone. Add a Python file drone.py to this project. Define a class Drone in this file. This class has the following five methods:

1. \_\_init\_\_ : Create two instance variables to store the speed and the height of the drone. Initialize them to 0.0. This method has no parameters other than self and no return value.
2. accelerate : Increase the speed of the drone by 10. The new speed cannot be negative. This method has no parameters other than self and no return value.
3. decelerate : Decrease the speed of the drone by 10. This method has no parameters other than self and no return value.
4. ascend : Increase the height of the drone by 10. This method has no parameters other than self and no return value.
5. descend : Decrease the height of the drone by 10. The new height cannot be negative. This method has no parameters other than self and no return value.

The following class diagram shows the design of this class:

|  |
| --- |
| **Drone** |
| +speed: Float  +height: Float |
| +create()  +accelerate()  +decelerate()  +ascend()  +descend() |

Add a file fly\_drone\_main.py to this project. In this main module, create an instance of Drone. Write a loop to control the speed and height of the drone. In the loop, ask the user to enter 1 for acceleration, 2 for deceleration, 3 for ascending, 4 for descending, or 0 for exit. Call the appropriate method of the Drone object to change the speed or height of the drone. Every time the speed or height of the drone changes, display the speed and height.

The following shows a sample test run:

Enter 1 for accelerate, 2 for decelerate, 3 for ascend, 4 for descend, 0 for exit: 1

Speed: 10 Height: 0

Enter 1 for accelerate, 2 for decelerate, 3 for ascend, 4 for descend, 0 for exit: 1

Speed: 20 Height: 0

Enter 1 for accelerate, 2 for decelerate, 3 for ascend, 4 for descend, 0 for exit: 3

Speed: 20 Height: 10

Enter 1 for accelerate, 2 for decelerate, 3 for ascend, 4 for descend, 0 for exit: 3

Speed: 20 Height: 20

Enter 1 for accelerate, 2 for decelerate, 3 for ascend, 4 for descend, 0 for exit: 3

Speed: 20 Height: 30

Enter 1 for accelerate, 2 for decelerate, 3 for ascend, 4 for descend, 0 for exit: 2

Speed: 10 Height: 30

Enter 1 for accelerate, 2 for decelerate, 3 for ascend, 4 for descend, 0 for exit: 0

Speed: 10 Height: 30

Zip your Python project and submit it to Blackboard for credit.

## Problem 2

Create a Python project manage\_course. Add a Python file course.py to this project. Define a class Course in this file. This class has three publicly accessible instance variables to store course code, quota and enrollment. Define the following methods:

1. An \_\_init\_\_ method that accepts the course code and quota as arguments. Write statements in \_\_init\_\_ to store them in instance variables. Also create an instance variable to store enrollment and initialize it to 0.
2. An add\_student method to increase enrollment by one and display “One student added” if the course is not full. If the course is already full, make no change to enrollment and display “Course already full”. This method has no argument other than self and has no return value.
3. A drop\_student method to decrease enrollment by one and display “One student dropped” if the course is not empty. If the course is empty, make no change to enrollment and display “Course is empty”. This method has no argument other than self and has no return value.

The following class diagram shows the design of this class:

|  |
| --- |
| **Course** |
| +course code: String  +quota: Integer  +enrollment: Integer |
| +create(code:String, quota:Integer)  +add\_student()  +drop\_student() |

Add a file manage\_course\_main.py to this project. This is the main module. Write code to ask the user to enter course code and quota of a course. Use the data provided by the user to create an instance of Course. Write a loop to manage this course. In the loop, ask the user to enter 1 for adding a student, 2 for dropping a student, 3 for displaying course info, or 0 for exit. If 1 is chosen, call the add\_student method of the Course object and use a print statement to display updated enrollment. If 2 is chosen, call the drop\_student method of the Course object and use a print statement to display updated enrollment. If 3 is chosen, display values stored in the Course object to show course code, quota and enrollment.

The following shows a sample test run:

Enter course code: CSC121

Enter course quota: 3

Enter 1 for add student, 2 for drop student, 3 for course info, 0 for exit: 2

Course is empty.

Enrollment: 0

Enter 1 for add student, 2 for drop student, 3 for course info, 0 for exit: 1

One student added.

Enrollment: 1

Enter 1 for add student, 2 for drop student, 3 for course info, 0 for exit: 1

One student added.

Enrollment: 2

Enter 1 for add student, 2 for drop student, 3 for course info, 0 for exit: 1

One student added.

Enrollment: 3

Enter 1 for add student, 2 for drop student, 3 for course info, 0 for exit: 1

Course already full.

Enrollment: 3

Enter 1 for add student, 2 for drop student, 3 for course info, 0 for exit: 2

One student dropped

Enrollment: 2

Enter 1 for add student, 2 for drop student, 3 for course info, 0 for exit: 3

Course code: CSC121

Quota: 3

Enrollment: 2

Enter 1 for add student, 2 for drop student, 3 for course info, 0 for exit: 0

Zip your Python project and submit it to Blackboard for credit.

## Problem 3

Write a new version of the fly\_drone project you wrote in Problem 1. Make a copy of the fly\_drone project folder and rename it to fly\_drone\_new. Make the following changes to class Drone:

1. Change all instance variables to private
2. Define a \_\_str\_\_ method to display the drone’s speed and height.

In the main module, you also need to modify the print statement that displays the speed and height of the drone because those two attributes are private now.

The following shows a sample test run:

Enter 1 for accelerate, 2 for decelerate, 3 for ascend, 4 for descend, 0 for exit: 1

Speed: 10 Height: 0

Enter 1 for accelerate, 2 for decelerate, 3 for ascend, 4 for descend, 0 for exit: 1

Speed: 20 Height: 0

Enter 1 for accelerate, 2 for decelerate, 3 for ascend, 4 for descend, 0 for exit: 3

Speed: 20 Height: 10

Enter 1 for accelerate, 2 for decelerate, 3 for ascend, 4 for descend, 0 for exit: 3

Speed: 20 Height: 20

Enter 1 for accelerate, 2 for decelerate, 3 for ascend, 4 for descend, 0 for exit: 3

Speed: 20 Height: 30

Enter 1 for accelerate, 2 for decelerate, 3 for ascend, 4 for descend, 0 for exit: 4

Speed: 20 Height: 20

Enter 1 for accelerate, 2 for decelerate, 3 for ascend, 4 for descend, 0 for exit: 4

Speed: 20 Height: 10

Enter 1 for accelerate, 2 for decelerate, 3 for ascend, 4 for descend, 0 for exit: 2

Speed: 10 Height: 10

Enter 1 for accelerate, 2 for decelerate, 3 for ascend, 4 for descend, 0 for exit: 0

Speed: 10 Height: 10

Zip your Python project and submit it to Blackboard for credit.

## Problem 4

Write a new version of the manage\_course project you wrote in Problem 2. Make a copy of the manage\_course project folder and rename it to manage\_course\_new. Make the following changes to class Course:

1. Change all instance variables to private
2. Write getter methods for course code, quota and enrollment
3. Write setter method for quota. This method accepts new quota as an argument. If new quota is negative, display “Quota cannot be negative”. Otherwise, set quota to new quota and display “Quota changed”. This method has no return value.

In the main module, you need to call getter methods to retrieve data stored in instance variables of the course objects because they are private now. In addition, add a choice of changing quota to the menu. If this choice is chosen, ask the user to enter new quota and call the setter method of quota to set a new quota.

The following shows a sample test run:

Enter course code: CSC121

Enter course quota: 2

Enter 1 for add student, 2 for drop student, 3 for course info, 4 for change quoita, 0 for exit: 1

One student added.

Enrollment: 1

Enter 1 for add student, 2 for drop student, 3 for course info, 4 for change quoita, 0 for exit: 1

One student added.

Enrollment: 2

Enter 1 for add student, 2 for drop student, 3 for course info, 4 for change quoita, 0 for exit: 1

Sorry! Course already full.

Enrollment: 2

Enter 1 for add student, 2 for drop student, 3 for course info, 4 for change quoita, 0 for exit: 4

Enter new quota: 3

Quota changed.

Enter 1 for add student, 2 for drop student, 3 for course info, 4 for change quoita, 0 for exit: 3

Course code: CSC121

Quota: 3

Enrollment: 2

Enter 1 for add student, 2 for drop student, 3 for course info, 4 for change quoita, 0 for exit: 1

One student added.

Enrollment: 3

Enter 1 for add student, 2 for drop student, 3 for course info, 4 for change quoita, 0 for exit: 0

Zip your Python project and submit it to Blackboard for credit.

# Grading rubric for problem 1

Writing the \_\_init\_\_ method [3 points]

Writing the accelerate method [3 points]

Writing the decelerate method [3 points]

Writing the ascend method [3 points]

Writing the descend method [3 points]

Writing main module [10 points]

# Grading rubric for problem 2

Writing the \_\_init\_\_ method [4 points]

Writing the add\_student method [4 points]

Writing the drop\_student method [4 points]

Writing main module [13 points]

# Grading rubric for problem 3

Making instance variables private [5 points]

Writing \_\_str\_\_ method [5 points]

Displaying object’s attributes in main module [5 points]

Other statements [5 points]

# Grading rubric for problem 4

Making instance variables private [5 points]

Writing getter method [10 points]

Writing setter method [5 points]

Displaying object’s attributes in main module [5 points]

Other statements [5 points]