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

**CSC121 PYTHON Programming**

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

Programming Project 2

# Objectives

In this project, students will learn:

- How to apply object oriented design

- How to create modules and functions

- How to create and use objects

- How to store data in lists

- How to create and use selection control structures

- How to create and use iterative control structures

- How to add comments to Python code

# Goals

In this project, students will demonstrate the abilities to:

- Apply object-oriented design

- Create modules and functions

- Create and use objects

- Store data in lists

- Create and use selection control structures

- Create and use iterative control structures

- Add comments to Python code

# Project Descrpiton

In project 1 we wrote a program for class registration system. This time we are going to rewrite that program with object oriented design.

This program creates a class registration system. It allows users to log in as students or administrators. A student user can add courses, drop courses and list courses he/she has registered for. An administrator user can show class rosters and change maximum class sizes.

There will be four classes in the object oriented design. The first one is the Course class.

| **Course** |
| --- |
| - course\_code: String  - max\_size: Integer  - roster: List |
| + Course(c\_code: String, m\_size: Integer)  + add\_student(id: String)  + drop\_student(id: String)  + display\_roster()  + change\_max\_size()  + get\_course\_code(): String  + student\_in\_course(id): Bool |

The Course class has three private instance variables. The constructor takes two arguments: c\_code is course code; m\_size is maximum class size. Use c\_code and m\_size to initialize the instance variables course\_code and max\_size. Initialize roster to an empty list.

The add\_student method adds a student to the roster. It has one parameter: id, which is the ID of the student to be added. If the course is already full, display error message and stop. If the student is already enrolled, display error message and stop. Otherwise, add student to roster and display a message showing which student added to which course. It has no return value.

The drop\_student method removes a student from roster. It has one parameter: id, which is the ID of the student to be dropped. If the student is not enrolled, display error message and stop. Otherwise, remove student from roster and display a message showing which student dropped from which course. It has no return value.

The display\_roster method sorts and displays ID of the students enrolled in this course and the enrollment count. It has no parameter and no return value.

The change\_max\_size method changes the maximum class size. It has no parameter. It displays current enrollment count and current maximum class size. It asks user to enter new max size. If new max size is smaller than current enrollment count, display error message and ask for a new max size repeatedly until it is not smaller than current enrollment count. It has no return value.

The get\_course\_code method has no parameter and returns the value of the instance variable course\_code.

The student\_in\_course method tests whether a student is enrolled in this course. It has one parameter: id, which is the ID of the student to be tested. If student is enrolled in this course, return true. Otherwise, return false.

The second class is User, which is the base class of Student and Admin.

| **User** |
| --- |
| #id: String  #pin: String |
| + User(id: String, pin: String)  + get\_id(): String  + get\_pin(): String |

| **Student** |
| --- |
|  |
| + Student(id: String, pin: String)  + add\_course(c\_list: List)  + drop\_course(c\_list: List)  + list\_courses(c\_list: List) |

| **Admin** |
| --- |
|  |
| + Admin(id: String, pin: String)  + show\_roster(c\_list: List)  + change\_max\_size(c\_list: List) |

The User class has two protected instance variables: id and pin. The constructor takes two arguments: id and pin. Use these arguments to initialize corresponding instance variables.

The get\_id method has no parameter and returns the value of the instance variable id.

The get\_pin method has no parameter and returns the value of the instance variable pin.

The Student class has no instance variables other than the ones inherited from User. The constructor takes two arguments: id and pin. It calls the constructor of the base class.

The add\_course method adds a student to a course. It has one parameter: c\_list, which is the list of Course objects. It asks user to enter the course he/she wants to add. If the course is not offered, display error message and stop. Otherwise, call the add\_student method of the course to add the student. This method has no return value.

The drop\_course method drops a student from a course. It has one parameter: c\_list, which is the list of Course objects. It asks user to enter the course he/she wants to drop. If the course is not offered, display error message and stop. Otherwise, call the drop\_student method of the course to drop the student. This method has no return value.

The list\_courses method displays and counts courses a student has registered for. It has one parameter: c\_list, which is the list of Course objects. It iterates over c\_list to display and count courses the student is in the roster. This method has no return value.

The Admin class has no instance variables other than the ones inherited from User. The constructor takes two arguments: id and pin. It calls the constructor of the base class.

The show\_roster method displays the roster of a course. It has one parameter: c\_list, which is the list of Course objects. It asks user to enter the course he/she wants to see the roster. If the course is not offered, display error message and stop. Otherwise, call the display\_roster method of the course to display the roster. This method has no return value.

The change\_max\_size method changes the maximum size of a course. It has one parameter: c\_list, which is the list of Course objects. It asks user to enter the course he/she wants to change max size. If the course is not offered, display error message and stop. Otherwise, call the change\_max\_size method of the course to maximum size. This method has no return value.

You must define the following functions in the main module.

| Function | Specification |
| --- | --- |
| login(u\_list) | This function allows a student or an administrator to log in. It has one parameter: u\_list, which is a list of User objects (Student objects and Admin objects are User objects). This function asks user to enter ID and PIN. If they match the data of an element in u\_list, display message of verification and return the index of that element (e.g. return 0 if it is the first element of the list, 1 if it is the second element, and so on). Otherwise, display error message and return -1. |
| student\_session(c\_list, s\_list) | This function creates a student session. It has two parameters: c\_list is the list of Course objects; s\_list is the list of Student objects. It calls the login function for the student user to log in. If login is successful, use a loop to allow the user to add, drop and list courses until the user wants to exit. Call methods of the Student object to handle the tasks. This function has no return value. |
| admin\_session(r\_list, m\_list, a\_list) | This function creates an administrator session. It has two parameters: c\_list is the list of Course objects; a\_list is the list of Admin objects. It calls the login function for the administrator user to log in. If login is successful, use a loop to allow the user to show class roster and change max class size. Call methods of the Admin object to handle the tasks. This function has no return value. |
| main() | This function manages the whole registration system. It has no parameter. It creates student list, administrator list and course list. It uses a loop to allow users to create student and administrator sessions until the user wants to stop. Call either the student\_session function or the admin\_session function, depending on what the user chooses. This function has no return value. |

The main function is partially written for you. You must include the following code in your program without modifications.

def main():

course\_list = []

student\_list =[]

admin\_list = []

init\_lists(course\_list, student\_list, admin\_list)

The main function creates three lists and calls an init\_lists function. This function adds a few elements to these lists. This function makes testing and grading of the program easier. The code of this function has been written. You just need to copy and paste it in your main module.

**def** init\_lists(c\_list, s\_list, a\_list):  
  
 *# ------------------------------------------------------------  
 # This function adds elements to course\_list, student\_list and  
 # admin\_list. It makes testing and grading easier. It has  
 # three paramters: c\_list is the list of Course objects;  
 # s\_list is the list of Student objects; a\_list is the list of  
 # Admin objects. This function has no return value.  
 # -------------------------------------------------------------* course1 = Course(**"CSC121"**, 2)  
 course1.add\_student(**"1004"**)  
 course1.add\_student(**"1003"**)  
 c\_list.append(course1)  
 course2 = Course(**"CSC122"**, 2)  
 course2.add\_student(**"1001"**)  
 c\_list.append(course2)  
 course3 = Course(**"CSC221"**, 1)  
 course3.add\_student(**"1002"**)  
 c\_list.append(course3)  
  
 student1 = Student(**"1001"**, **"111"**)  
 s\_list.append(student1)  
 student2 = Student(**"1002"**, **"222"**)  
 s\_list.append(student2)  
 student3 = Student(**"1003"**, **"333"**)  
 s\_list.append(student3)  
 student4 = Student(**"1004"**, **"444"**)  
 s\_list.append(student4)  
  
 admin1 = Admin(**"7001"**, **"777"**)  
 a\_list.append(admin1)  
 admin2 = Admin(**"8001"**, **"888"**)  
 a\_list.append(admin2)

The following is an example.

Student 1004 added to CSC121

Student 1003 added to CSC121

Student 1001 added to CSC122

Student 1002 added to CSC221

Enter 1 if you are student, 2 if you are administrator, 0 to quit: 1

Enter ID: 1001

Enter PIN: 1

ID or PIN incorrect

Enter 1 if you are student, 2 if you are administrator, 0 to quit: 1

Enter ID: 1001

Enter PIN: 111

ID and PIN verified

Enter 1 to add course, 2 to drop course, 3 to see courses you have registered, 0 to exit: 1

Enter course you want to add: csc

Course not found

Enter 1 to add course, 2 to drop course, 3 to see courses you have registered, 0 to exit: 1

Enter course you want to add: CSC121

Course already full

Enter 1 to add course, 2 to drop course, 3 to see courses you have registered, 0 to exit: 1

Enter course you want to add: CSC122

You are already enrolled in this course

Enter 1 to add course, 2 to drop course, 3 to see courses you have registered, 0 to exit: 3

Course registered:

CSC122

Number of courses registered: 1

Enter 1 to add course, 2 to drop course, 3 to see courses you have registered, 0 to exit: 2

Enter course you want to drop: CSC

Course not found

Enter 1 to add course, 2 to drop course, 3 to see courses you have registered, 0 to exit: 2

Enter course you want to drop: CSC121

You are not enrolled in this course

Enter 1 to add course, 2 to drop course, 3 to see courses you have registered, 0 to exit: 2

Enter course you want to drop: CSC122

Student 1001 removed from CSC122

Enter 1 to add course, 2 to drop course, 3 to see courses you have registered, 0 to exit: 3

Course registered:

Number of courses registered: 0

Enter 1 to add course, 2 to drop course, 3 to see courses you have registered, 0 to exit: 0

Student session ended.

Enter 1 if you are student, 2 if you are administrator, 0 to quit: 2

Enter ID: 7001

Enter PIN: 777

ID and PIN verified

Enter 1 to show class roster, 2 to change max class size, 0 to exit: 2

Enter course: CSC221

Current enrollment: 1

Current max size: 1

Enter new size: 0

New max size cannot be smaller than current enrollment

Enter new size: 2

Enter 1 to show class roster, 2 to change max class size, 0 to exit: 1

Enter course: CSC121

1003

1004

Number of student: 2

Enter 1 to show class roster, 2 to change max class size, 0 to exit: 0

Administrator session ended.

Enter 1 if you are student, 2 if you are administrator, 0 to quit: 1

Enter ID: 1001

Enter PIN: 111

ID and PIN verified

Enter 1 to add course, 2 to drop course, 3 to see courses you have registered, 0 to exit: 1

Enter course you want to add: CSC221

Student 1001 added to CSC221

Enter 1 to add course, 2 to drop course, 3 to see courses you have registered, 0 to exit: 3

Course registered:

CSC221

Number of courses registered: 1

Enter 1 to add course, 2 to drop course, 3 to see courses you have registered, 0 to exit: 0

Student session ended.

Enter 1 if you are student, 2 if you are administrator, 0 to quit: 0

# Submission Requirments

You must create five Python files: one for the Course class, one for the User class, one for the Student class, one for the Admin class, and one for the main module. Add appropriate comments to your code. At the minimum, you must have program-level comment and function-level comments. Submit the five Python files to Blackboard for credit.

# Grading Rubric

main function [5 points]

login function [5 points]

student\_session function 5 points]

admin\_session function [5 points]

constructor of User class [3 points]

get\_id method of User class [3 points]

get\_pin method of User class [3 points]

constructor of Student class [3 points]

add\_course method of Student class [5 points]

drop\_course method of Student class [5 points]

list\_courses method of Student class [5 points]

constructor of Admin class [3 points]

show\_roster method of Admin class [5 points]

change\_max\_size method of Admin class [5 points]

constructor of Course class [3 points]

add\_student method of Course class [5 points]

drop\_student method of Course class [5points]

display\_roster method of Course class [5 points]

change\_max\_size method of Course class [5 points]

get\_course\_code method of Course class [3 points]

student\_in\_course method of Course class [3 points]

Program-level and function-level comments [6 points]

Putting class, method and function definitions in correct modules [5 points]