Important

1. Due Date: **Wednesday, November 30**
2. This homework is graded out of 100 points.
3. This is an Individual Assignment. You may collaborate with other students in this class. Collaboration means talking through problems, assisting with debugging, explaining a concept, etc. Students may only collaborate with fellow students currently taking CS 1301, the TA's and the lecturer. You should not exchange code or write code for others. For individual assignments, each student must turn in a unique program. Your submission must not be substantially similar to another student's submission. Collaboration at a reasonable level will not result in substantially similar code.
4. For Help:
    - TA Helpdesk (Schedule posted on class website.)
    - Email TA's or use Piazza Forums Notes:
    - How to Think Like a Computer Scientists [http://openbookproject.net/thinkcs/python/english3e/]
    - CS 1301 Python Debugging Guide [http://www.cc.gatech.edu/classes/AY2016/cs1301_spring/CS-1301-Debugging-Guide/index.html]
5. Don't forget to include the required collaboration statement (outlined on the syllabus).
6. Do not wait until the last minute to do this assignment in case you run into problems.
7. **Read the entire specifications document before starting this assignment.**

Object-Oriented Programming

Python is an **object-oriented programming language**. That means it provides features that support object-oriented programming (**OOP**). Object-oriented programming has its roots in the 1960s, but it wasn't until the mid 1980s that it became the main programming paradigm used in the creation of new software.

In object-oriented programming the focus is on the creation of objects, which contain both data and functionality together. Usually, each object definition corresponds to some object or concept in the real world and the functions that operate on that object correspond to the ways real-world objects interact. We've already seen classes like `Turtle`, `Math`, `Random` and many others. We are now ready to create our own user-defined class.

[http://openbookproject.net/thinkcs/python/english3e/classes_and_objects_I.html]

Introduction

The goal of this homework is for you to get practice and understand the basics of Object Oriented Programming. Your mission, should you choose to accept it, involves the creation of three classes: `Student, Course, and Major`. Each class will contain several methods for you to implement. What you should do for each method is described below and in the docstrings of the file. Below you will find more information to complete your assignment. Read it thoroughly before you begin. The key to this homework is to see the bigger picture of how the 3 classes are interconnected i.e., a Course has a Major or a Student can have many Courses. You have until – **Wednesday, November 30** - to complete this mission.

Student

```
class Student(builtins.object)
 |  Student have the following properties:
 |  Attributes:
 |      name: A string representing the student's name. (i.e. George)
 |      gtid: A string representing the student's GT id. (i.e. 1234)
 |      major: A Major object, representing the student's major. (i.e. <CS>)
 |      courses: A list representing the courses a student has taken or is currently
enrolled in. A new Student may or may not come with Courses already taken. (default
to empty list)
 |
 |  Methods defined here:
 |
 |  __init__(self)
 |      Initialize a Student object whose name is *name*, gtid is *gtid*, major is
*major*, courses is *courses* (default: empty)
 |
 |  __str__(self)
 |      String representation of a Student object
 |
 |  drop(self, course)
 |      Drop a course if student is registered to it. Raise a
DropNotEnrolledCourseException if student is not registered for that course. (TODO:
implement this simple exception yourself, helpful toturial
[http://www.programiz.com/python-programming/user-defined-exception]). Pass a message
of the exception, exactly this: "Not enrolled in: {0}".format(course)
 |
 |  get_class_standing(self)
 |      return a string representation of the student's class standing. Freshman: 0-9
credits; Sophomore: 10-19; Junior: 20-29; Senior: 30-39
 |
 |  get_missing_credits(self)
 |      return the student's missing amount of credits to graduate as as an int
(hint: Missing Credits = Required by Major - Total Credits)
 |
 |  get_total_credits(self)
 |      return the student's total amount of credits as an int
 |
 |  is_taking(self, course)
 |      Returns true if student is registered for a course
 |
 |  register(self, course)
 |      Add a course to the student's courses. No duplicate courses are allowed. If
student is registered for X course with code "ZZ 1234", it should not be allowed to
registered for Y course with code "ZZ 1234". In other words, what makes a course
unique is its code. (hint: implement this in the Course class; major hint: override
the __eq__ method in Course). Also, in our system a student can only register for
courses that belong to his/her major.
 |
 |  register_many(self, courses)
 |      register courses from a list of Course objects
```

Course

```
class Course(builtins.object)
 |  docstring of a Course object.
 |  Course have the following properties:
 |  Attributes:
 |      code: A string representing the course code (i.e CS1301)
 |      major: A Major object, representing the major a Course belongs to. (i.e
<CS>)
 |      credits: An integer representing the course's amount of credits (i.e 3)
 |      instructor: A  string representing the instructor of the course (i.e Jay
Summet)
 |
 |  Methods defined here:
 |
 |  __eq__(self, other)
 |      Override the eq method
 |
 |  __init__(self)
 |      Initialize a Course object whose code is *code*, credits is *cresits*,
and instructor is *instructor*.
 |
 |  __repr__(self)
 |      Return repr(self).
 |
 |  __str__(self)
 |      String representation of a Course object
```

Major

```
class Major(builtins.object)
 |  docstring of a Major object.
 |  Major have the following properties:
 |      Attributes:
 |          name: A string representing the major name (i.e CS)
 |          required_credits: An int representing the amount of credits required
to graduate (i.e 40)
 |
 |  Methods defined here:
 |
 |  __init__(self)
 |      Initialize an Instructor object whose name is *name* and required_credits
are *required_credits*
```

college_test.py

We have provided you with a python file called `college_test.py`. In this file we have created a series of tests for your usage. We understand you probably have never been exposed to testing code so you are not expected to do anything to this file or even use this file if you don't want to. However, we encourage you to use it as it will be highly beneficial in testing our your code. Feel free to add your own tests to the file to cover any additional cases you would like to test.

If you do desire to test your code, all you have to do is have the `college.py` and the `college_test.py` files in the same directory. Open and run `college_test.py`. After running the tests you should see their results. Check the results and start debugging if needed. If you pass all the tests you should see something like this as your output:

```
test_init_course_credits (__main__.CollegeTest)
A Course should be initialized with credits ... ok
test_init_course_instructor (__main__.CollegeTest)
A Course should be initialized with an instructor ... ok
test_init_empty_courses (__main__.CollegeTest)
A Student should be initialized with no courses ... ok
test_init_major_name (__main__.CollegeTest)
A Course should be initialized with a code ... ok
test_init_nonempty_courses (__main__.CollegeTest)
A Student can be initialized with some courses ... ok
test_init_student_id (__main__.CollegeTest)
A Student should be initialized with a gtID ... ok
test_init_student_name (__main__.CollegeTest)
A Student should be initialized with a name ... ok
test_is_taking (__main__.CollegeTest)
is_taking method works ... ok
test_register_1 (__main__.CollegeTest)
Register a single course ... ok
test_register_2 (__main__.CollegeTest)
Register more than one course ... ok
test_register_3 (__main__.CollegeTest)
No duplicates courses allowed in registration ... ok
test_register_4 (__main__.CollegeTest)
No courses of different major allowed in registration ... ok
test_register_many (__main__.CollegeTest)
A Student can register many courses at once ... ok

----------------------------------------------------------------------
Ran 27 tests in 2.873s

OK
```

Tips & Hints

1. Get familiar with OOP before you start coding. Revise your class notes and read section 15 of the book again.
2. Even if you do not want to use the test file, open it and read it, as this might help you understand what we are looking for and it might give you great hints on how to implement your classes and methods. Huge spoilers in this file.
3. Use the methods you are writing inside other methods
4. In `college_test.py` the module `setUp()` is always called first. Each test is supposed to be independent to all others. Read more about unittest here: [https://docs.python.org/3/library/unittest.html]

Provided

The following file(s) have been provided to you. There are several, but you will only edit one of them:

1. `college.py`
   This is the file you will edit and implement. All instructions for what the methods should do are in the docstrings.
2. `college_test.py`
   This is a test file that contains a set of tests if you wish to test your code. This part is completely optional and it's provided just for your benefit**. It is not intended to be exhaustive and does not guarantee any type of grade**. **Write your own tests if you wish to ensure you cover all edge cases.**

Deliverables

You must submit all of the following file(s). Please make sure the filename matches the filename(s) below. Be sure you receive the confirmation email from T-Square, and then download your uploaded files to a new folder and run them.

1. `college.py`