

Student Class List Project 3

Purpose: To better understand

- Linked lists
- Recursion
- Unit Tests

You will write an application which stores the courses taken by a student and prints out a report. Data will be read from a data.txt file. Once all data has been read from the data.txt file, your program should print a report as shown in the Program Output section below. After the list of courses has been printed, you should then calculate and display the cumulative GPA of all courses in the list.

Important Constraints

1. You must implement a linked list as the underlying data structure.
2. Course entries will be stored in the linked list structure.
3. All operations and list traversals **MUST** use recursion. Loops are **NOT** allowed in your linked list implementation. (Loops can be used in the code with reads the data from the data.txt file, but **NOT** in any other modules)
4. Each class will be in a separate module (i.e. course.py, courselist.py) and the driver program which reads the data from the data.txt file will be in main.py.
5. To allow the UnitTest to test for recursion usage, you **MUST** add the following to each module which has recursive functions:

```
from recursioncounter import RecursionCounter
```

and at the beginning of EVERY recursive function you must add this line:

```
_ = RecursionCounter()
```

Course ADT (course.py)

You will implement a Course ADT which implements the following methods:

- constructor: must have default values for all parameters and must validate all parameters.
- number(): retrieve Course Number as an integer
- name(): retrieve Course Name as a string

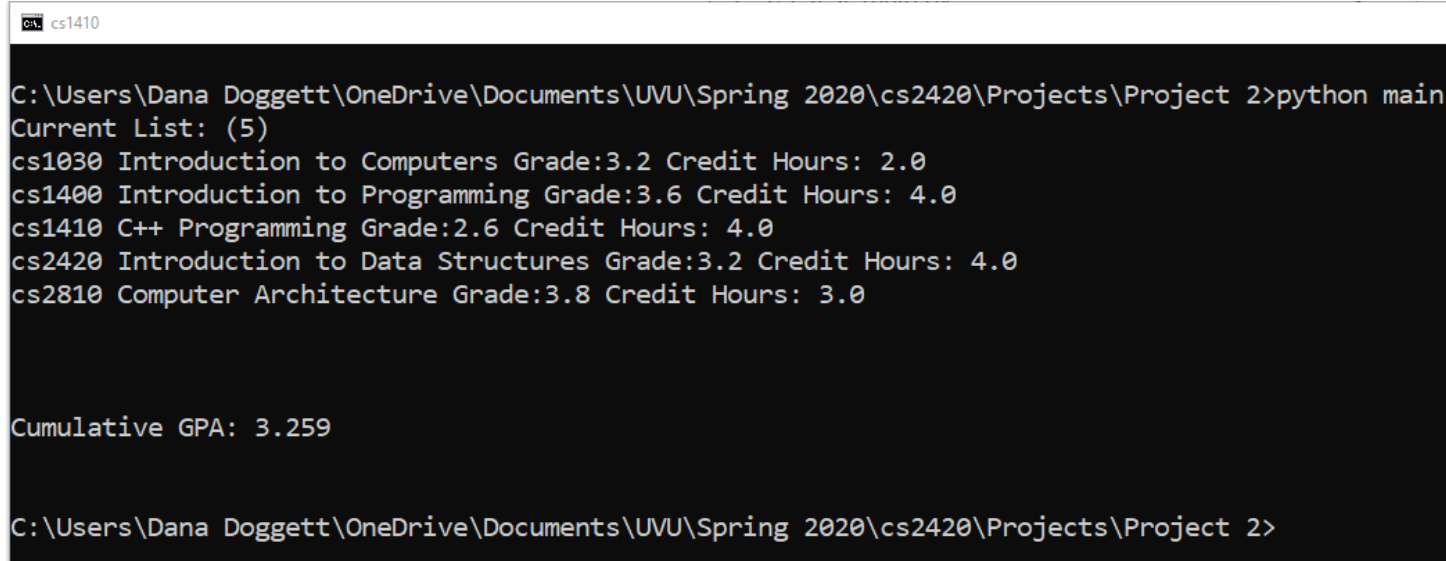
- `credit_hr()`: retrieve Credits as a floating-point number
- `grade()` : retrieve Grade as a numeric value in range 4.0 – 0.0
- `__str__()`: returns a string representing a single Course as shown in the Program Output section
- The courses taken by the student will be provided in a data file. You will have to parse the data from that file and store it in a Linked List of courses.

CourseList ADT (courselist.py)

Your CourseList ADT must use a Linked List implementation. The course list must be able to hold an **unlimited** number of Courses. The CourseList ADT implements the following methods:

- constructor to initialize all needed data for an empty list
- `insert(Course)`: insert the specified Course in Course Number ascending order
- `remove(number)`: remove the first occurrence of the specified Course
- `remove_all(number)`: removes ALL occurrences of the specified Course
- `find(number)`: find the first occurrence of the specified course in the list or return -1
- `size()`: return the number of items in the list
- `calculate_gpa()`: return the GPA using all courses in the list
- `is_sorted()`: return True if the list is sorted by Course Number, False otherwise
- `__str__()`: returns a string with each Course's data on a separate line (as shown in the Program Output)
- `__iter__()` and `__next__()`: the linked list must be iterable

Program Output



```

C:\Users\Dana Doggett\OneDrive\Documents\UVU\Spring 2020\cs2420\Projects\Project 2>python main
Current List: (5)
cs1030 Introduction to Computers Grade:3.2 Credit Hours: 2.0
cs1400 Introduction to Programming Grade:3.6 Credit Hours: 4.0
cs1410 C++ Programming Grade:2.6 Credit Hours: 4.0
cs2420 Introduction to Data Structures Grade:3.2 Credit Hours: 4.0
cs2810 Computer Architecture Grade:3.8 Credit Hours: 3.0

Cumulative GPA: 3.259

C:\Users\Dana Doggett\OneDrive\Documents\UVU\Spring 2020\cs2420\Projects\Project 2>

```

Figure 1. Example Program Output

Test Cases

Your three files (`main.py`, `course.py`, and `courselist.py`) will be auto-tested with the supplied `test_main.py` file. This is a standard Python `UnitTest` file and will exercise your code to ensure it is working properly. If you fail any test, you should be able to find the problem keep working on it until all tests pass. Use this `UnitTest` as a development tool to polish your programming skills. Your grade will be based on both the results of the `UnitTests` and proper program output (see above). (the output of `main()` will also be evaluated by `unittest`.)

In addition to fully exercising your code, the `UnitTest` will also use `PyLint` to check your “code quality”. It uses a standard Python coding standard called `Pep-8`. It may seem tedious at first, but you can integrate `PyLint` into most IDE’s and this will help you create more readable, quality code. ALMOST ALL COMPANIES USE SOME SORT OF CODING STANDARD, so get used to it now!

All python files must pass `PyLint` with a score of 8.5 or better.

Grading (100 points)

All scores are a result of the Unit Test

- Empty Course 5
- Nominal Course 5
- Empty Course List 5
- Nominal Course List 35
- Recursion 20
- Coding Standards 15
- `main()` output 15

Files to turn in through Canvas

- `main.py` (driver file)
- `course.py`
- `courselist.py`