Exercises For Classes and Objects

Exercise 1

1. Using the `library_end`(From class_obj_code_1 file) create `checkout_book(book)` method on the `Library` class that will remove a book from the list of books in the `Library`.

2. Using the `course_example_end`(From class_obj_code_2 file) add a method on the `Course` named `get_assignment_average(assignment_id)` that will return the average grade for the assignment.

3. Using the `course_example_end` add a method on the `Student` named `get_average()` that will return the average grade for the student.

4. Using the `course_example_end` add methods on the `Course` named `list_students()` and `list_assignments()` whichc will list all students and assignments in the course and their ids respectively.

5. Using the `course_example_end` add a new class called `School` in the tools directory:

   `School` has the following properties:

   - name of school
   - list of courses

   `School` has the following methods:

   - add_course(): Adds a new course to the school
   - list_courses(): Lists the name of all available courses

6. Convert the course_app.py into a fully functioning application for a single school where you can do the following:

   - add a new course to the School
   - list all of the courses in the School
   - manage a course (create new function in course_app.py called `manage_course(course)` for this):
     - add student (no duplicate ids)
     - add assignments (no duplicate ids)
     - add submissions (no duplicate ids)
     - get course average
     - get assignment average
     - get student average
   - Note: Make use of the methods in the `Course` class in your `manage_course()` function.
   - Create a function in course_app.py called `load_data(school)` that will load the course from the existing example into the School at the beginning of main (after creating the school)
   - Ensure proper error handling where appropriate
   - Follow best practice for classes and objects.