

Unit 7: Data Structures

Lesson 4: Objects and Return Values

Name:

Do your coding in the **U7L4 Objects and Return Values** Replit

You might want [this code](#) from the demo as a reference.

GOAL: In the following lab, the goal is to create a new Student class, then to write a program that uses a Student object to track a student's test scores and calculate the average test score.

1. Add a new **student.py** file to your project and write a new Student class (capital "S").

2. Give your Student class an `__init__` method with *two* parameters: `self` and `name`. Use this method to initialize the `name` attribute of the object (`self.name`) to the value of the parameter. Then, initialize `self.test_scores` to an empty list (inside the `__init__` function)

3. Add an `add_test` method with two parameters (`self` and `test_score`) to your Student class. This method should add `test_score` to the end of the `self.test_scores` list. The method doesn't print or return anything.

TEST: In **main.py**, copy/paste/run this test code:

```
from student import Student

student1 = Student("Ben")
student1.add_test(92)
student1.add_test(94)
student1.add_test(90)
student1.add_test(96)
student1.add_test(87)
print(student1.test_scores)
```

EXPECTED OUTPUT:

```
[92, 94, 90, 96, 87]
```

4. Add a `get_average` method with one parameter (just `self`) to your Student class. This method should calculate and return the average of the test scores currently stored in `self.test_scores` list.

TEST: in **main.py**, copy/paste/run this test code:

```
from student import Student
```

```
student1 = Student("Ben")
student1.add_test(92)
student1.add_test(94)
student1.add_test(90)
student1.add_test(96)
student1.add_test(87)
print(student1.test_scores)
print(student1.get_average())
```

EXPECTED OUTPUT:

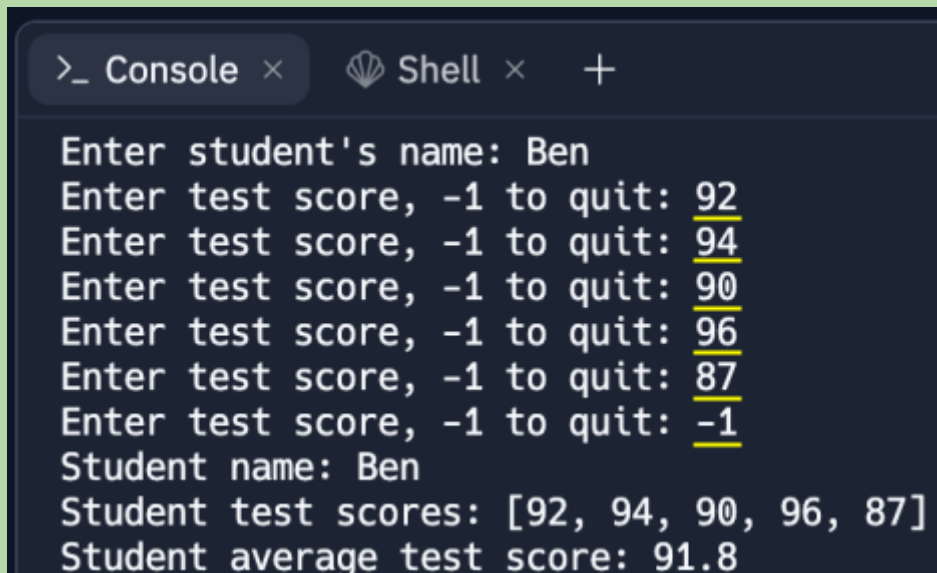
```
[92, 94, 90, 96, 87]
91.8
```

Time to use your Student class in an actual program that does something!

5. Clear out the test code from main.py
6. In main.py, write a small program that asks the user to enter a student's name, and allows them to type in various test scores (as integers) for that student until they enter -1 to quit. After they quit, print out the stats for that student: student name, test scores, average test score.

Your program should create and use a Student object in your project to store the student's name and test scores and perform the average calculation.

SAMPLE PROGRAM EXECUTION:



```
>_ Console x Shell x +
Enter student's name: Ben
Enter test score, -1 to quit: 92
Enter test score, -1 to quit: 94
Enter test score, -1 to quit: 90
Enter test score, -1 to quit: 96
Enter test score, -1 to quit: 87
Enter test score, -1 to quit: -1
Student name: Ben
Student test scores: [92, 94, 90, 96, 87]
Student average test score: 91.8
```

7. Challenge! Update your program so that a Student class can also have a list of courses that they are taking. Then, when a student is entering their grades, they should also specify which course they are adding their grades to.

When your Student class is created, they should have no courses. The user should be able to add courses first, then add grades for their course. They should then be able to see their average for all of their courses.

You will have to be able to properly associate a list of grades with a specific course. You can do this with a list of courses and a list of grades where each item in the grade list is *another* list of grades. For example:

Courses:

"Algebra 2"	"AP CSP"	"English"	"Chemistry"	"AP World"
-------------	----------	-----------	-------------	------------

Grades:

[91, 92, 99, 85]	[100, 100, 100, 99]	[75, 77, 90, 85]	[85, 94]	[81, 80, 94, 99, 65]
--------------------	-----------------------	--------------------	------------	------------------------

Based on the relationship between these lists, AP CSP (index 1), for example, corresponds with the grades 100, 100, 100, 99.

You should create a menu that allows the user to add courses, add a list of grades for a course, or quit the program. All of your data (courses, test grades per course) and all of the logic (getting averages by course, adding a course, adding test grades to a course) should happen INSIDE your Student class (not in main.py)

Here is a [sample video](#) of how the program should function.

Copy/paste your main program code below (your entire main.py file):

Copy/paste your Student class code below (your entire student.py file):

U7L4 Rubric

Description	Points
<u>Create the Student class</u> <ul style="list-style-type: none">• <i>Stores information about the student's name, test scores, and allows the user to calculate the average grade for their list of scores</i>	20
<u>Write the main program utilizing the Student class:</u> <ul style="list-style-type: none">• <i>Allows the user to enter their name</i>• <i>Provide a menu option that allows the user to input a list of grades and prints out their average grade</i>• <i>Allows the user to loop back and add more grades until they quit</i>	8
<u>Challenge task</u> <ul style="list-style-type: none">• <i>Complete the challenge task according to all specifications</i>	2

→ **Done!** ←

Submit in Google Classroom

Turn in