Name: Michael Keel, mkeel

Date Submitted: 06/04/25

Course: Foundations of Programming: Python

GitHub: https://github.com/mikeel4real/IntroToProg-Python-Mod07

Assignment 07

Introduction

In this document I will review the events from Week 7 of Foundations of Programming: Python. I started this week with review of the Module Notes and Videos to clarify the specifics of how objects differ from classes, and the addition of constructors, and how to benefit from inheritance.

Order of Operations

The first steps I completed this week were to read through the module notes and tried to review the new concepts covered in Week 7. I have some familiarity with looking at __init__ initialization method from my work but did not understand it's full context. Now I understand it is necessary to initialize objects created from classes so that they can assign internal object attributes. Similar to Week 6 I started my shell with generic assignment of classes seen in Figure 1 as directed by the assignment.

```
# TODO Create a Person Class
class Person:
    """Person data to be included is first name and last name
    Change Log: Michael Keel, 06/02/2025, Created class
    """

# TODO Add first_name and last_name properties to the constructor
# TODO Create a getter and setter for the first_name property
# TODO Create a getter and setter for the last_name property
# TODO Override the __str__() method to return Person data

# TODO Create a Student class the inherits from the Person class
class Student(Person):
    """Student data and inhereted Person data;
    Includes gpa along with Person data
    Change Log: Michael Keel, 06/02/2025, Created class
    """
# TODO call to the Person constructor and pass it the first_name and last_name data
# TODO add a assignment to the course_name property using the course_name parameter
# TODO add the getter for course_name
# TODO add the setter for course_name
# TODO Override the __str__() method to return the Student data
```

Figure 1. Start with Class entries

Once these were in place I moved from top to bottom of the TODO statements to write code entries that satisfy the requirements. The first entry is extremely close to the "Student" class entry contained within Lab01 as seen in Figure 2.

```
class Student:
    """
    A class representing student data.

    Properties:
    - first_name (str): The student's first name.
    - last_name (str): The student's last name.
    - gpa (float): The gpa of the student.

    ChangeLog:
    - RRoot, 1.1.2030: Created the class.
    """

    def __init__(self, first_name: str = '', last_name: str = '', gpa: float = 0.00):
        self.first_name = first_name
        self.last_name = last_name
        self.gpa = gpa
```

Figure 2. Example provided from *Mod07-Notes.docx* page 10

From this example I created the TODO 2 comment for name properties to the "Person" class. For the first getter and setter obligations, I used a convenient example introduced on page 16 of the *Mod07-Notes.docx*. The next TODO referred to the override method which I found on page 22 of the notes document. Subsequently the rest of the TODOs were slightly altered versions of these base formats and thus I applied the tools previously covered to resolve. The only addition outside of these was the super function used to call a parent class. This component was also discussed clearly in the notes and supported with relevant examples.

Remainder of the Work

The remaining code block I left unaltered because it appeared to work and not be the intent of the assignment. I small amount of debug compared to the previous weeks and ran my code. To my shock it worked on the first try as seen in Figure 3.

```
Enter your menu choice number: 3
-----------------------------------------------
Student Vic Vu is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student frimple pants is enrolled in python 101
-----------------------------------------------


---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
----------------------------------------


Enter your menu choice number: 4
Program Ended

Process finished with exit code 0
```

Figure 3. Sample intended output from code

Summary

Over the course of the previous week, I reviewed an object's relationship with a class and the utility of having parent-child inheritance and other minor features that increase ease of use to code authors. The use of getters and setters is also a functional way to separate the actions to be referenced separately as needed. In all we are continuing to add complexity week by week to the same input and output which makes comprehension easier to manage from a learner's perspective.