Christina Alvarez
November 27th, 2024
Foundations of Programming (Python)
Assignment 07
GitHub link: https://github.com/Christina64/IntroToProg-Python-Mod07.git

# Assignment 07: Classes and Objects

**Introduction**
In this document I will cover assignment seven where we continue to iterate on classes and objects. The document will be broken out into the following parts:

- Terms to know
- Steps taken for assignment 07
- Summary of learnings

**Terms to Know**
Functions: a block of code that runs when it is called upon. (per w3schools.com).
Classes: classes are a way to bring functionality and data together. (per docs.python.org).

- One can use functions within a class when using the syntax ClassName.function_name while also using the @staticmethod function decorator. (per Mod 07 notes).

Parameters: a variable noted within the parentheses of the function definition. (per w3schools.com).
Arguments: this is the value a function sends when it is called. (per w3schools.com).

**Steps Taken for Assignment 07**
I started this assignment with getting the starter code and updating the header to reflect myself as the now author of the script. From there I began adding classes to support the tasks of creating a person. See figure 1.

```
32  class Person:
33      def __init__(self,first_name,last_name):
34          self._first_name=first_name
35          self._last_name=last_name
```

Figure 1. Class for person

Properties were added for the classes. And I used a title property to denote that the first letter of the names should be capitalized. Figure 2 below.

```
37      @property
38      def first_name(self):
39          return self._first_name.title()
40      @property
41      def last_name(self):
42          return self._last_name.title()
```

Figure 2. Getter properties

Setters for the properties were created as well as value errors to ensure that only letters are user for both the first and last names. Figure 3 shows these updates.

```python
class Person:
    def __init__(self,first_name:str, last_name:str):
        self._first_name=first_name
        self._last_name=last_name

    @property
    def first_name(self)->str:
        return self._first_name.title()
    @first_name.setter
    def first_name(self,value:str):
        if value.isalpha():
            self._first_name=value
        else:
            raise ValueError("First name must be letters only.")

    @property
    def last_name(self)->str:
        return self._last_name.title()
    @last_name.setter
    def last_name(self,value:str):
        if value.isalpha():
            self._last_name=value
        else:
            raise ValueError("Last name must be letters only.")
```

Figure 3. Setters for the properties

The string method was added as seen in figure 4.

```python
    def __str__(self):
        return f'{self.first_name},{self.last_name}'

```

Figure 4. string method

The student class was then created as a child to the person class. With the student class everything from the person class is inherited. I also overloaded the class meaning I added more parameters to it. For this assignment the additional parameter is the course name. See figure 5.

```
class Student(Person):
    def __init__(self, first_name: str, last_name: str,course_name:str):
        super().__init__(first_name,last_name)
        self._course_name = course_name

```
Figure 5. Student class added

Properties and setters were added for the course name. See figure 6.

```
class Student(Person):
    def __init__(self, first_name: str, last_name: str,course_name:str):
        super().__init__(first_name,last_name)
        self._course_name = course_name
    @property
    def course_name(self)->str:
        """
        Returns the course_name.
        :return: Course_name
        """
        return self._course_name
    @course_name.setter
    def course_name(self,value)->None:
        self._course_name = value
```
Figure 6. Course name property and setter

The program was then updated to use the classes person and student. The dictionaries were updated and moved them below the properties. See figure 7.

```
# Define the Data Variables
students: list[Student] = []  # a table of student data
menu_choice: str  # Hold the choice made by the user.
```
Figure 7. updated students

The program was then updated from the main body down to update the list of students was used instead of a list of dictionaries. Below figures show where the updates were made.

```
    @staticmethod
    def read_data_from_file(file_name: str, student_data: list[Student])->list[Student]:
        """ This function reads data from a json file and loads it into a list of dictionary rows
```

Figure 8. Read data from file updated

```
252     @staticmethod
253     def input_student_data(student_data: list[Student])->list[Student]:
254         """ This function gets the student's first name and last name, with a course
```

Figure 9. Input a list of students and output a list of students

Removed redundant code for checking the students since the classes will now do this. Figure 10 shows the update.

```
264         try:
265             student_first_name = input("Enter the student's first name: ")
266             student_last_name = input("Enter the student's last name: ")
267             course_name = input("Please enter the name of the course: ")
268             student = {"FirstName": student_first_name,
269                        "LastName": student_last_name,
270                        "CourseName": course_name}
```

Figure 10. Removal of checks

Output for student and course name was updated to use the classes.

```
234     @staticmethod
235     def output_student_and_course_names(student_data: list[Student]):
236         """ This function displays the student and course names to the user
237
238         ChangeLog: (Who, When, What)
239         RRoot,1.1.2030,Created function
240
241         :param student_data: list of dictionary rows to be displayed
242
243         :return: None
244         """
245
246         print("-" * 50)
247         for student in student_data:
248             print(f'Student {student.first_name} '
249                   f'{student.last_name} is enrolled in {student.course_name}')
250         print("-" * 50)
```

Figure 11. Output for student and course name updated

Write data to file needed to be updated as it is currently a json dump. See figure 12 for updates.

```python
    @staticmethod
    def write_data_to_file(file_name: str, student_data: list):
        """ This function writes data to a json file with data from a list of dictionary rows

        ChangeLog: (Who, When, What)
        RRoot,1.1.2030,Created function

        :param file_name: string data with name of file to write to
        :param student_data: list of dictionary rows to be writen to the file

        :return: None
        """
        file_data=[]
        for student in student_data:
            file_data.appemd({'first_name':student.first_name,
                              'last_name':student.last_name,
                              'course_name':student.course_name})
        file=None
        try:
            file = open(file_name, "w")
            json.dump(file_data, file)
            file.close()
            IO.output_student_and_course_names(student_data=student_data)
        except Exception as e:
```

Figure 12. update to write data file

Lastly I ran the program to ensure all is working as expected. Figure 13 shows the output for menu choice one and two.

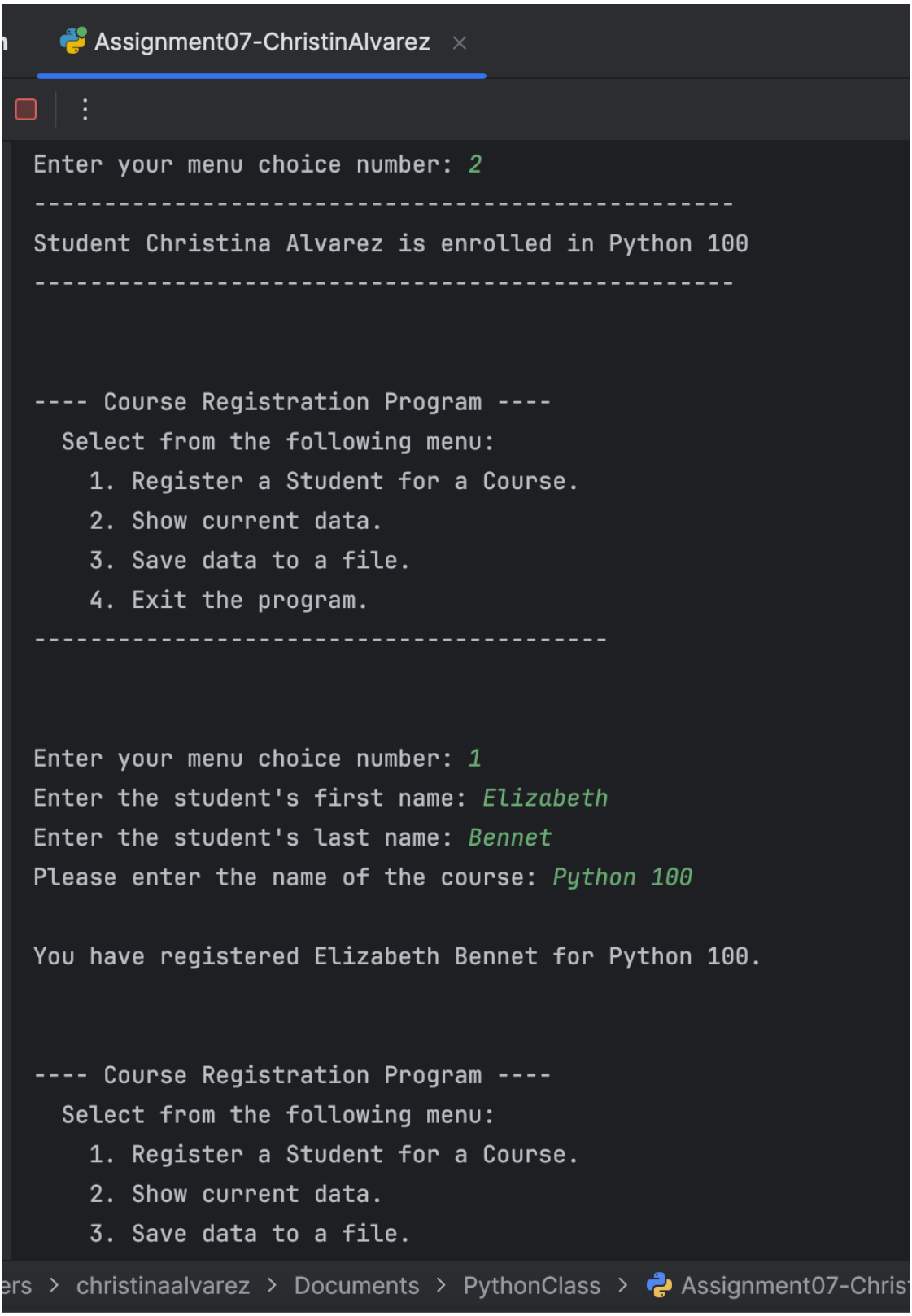

Figure 13. Program running with menu choice 1 and 2

**Summary of learnings**

Assignment seven was a great assignment to build more confidence in using functions, classes and dictionaries. This assignment was also a great exercise for building cleaner and more concise code. Additionally, if code is done well it can be reused for other projects to cut down on time and effort.