Ben Stafford

4 March 2024

IT FDN 110 A

Assignment 07

Assignment 7: Object Classes and Constructors

Introduction

Assignment 7 focused on adding in object classes which allow properties to be set up and changed ahead of time, and then inherited to future classes. We learned to set up constructors, getters, setters, and overrides. This allows us to carry over properties from more general classes like 'person', to more specific ones like 'student'.

Rewriting Script to Include Classes and Functions

Starting with constants and variables, it seemed like everything was correct in the starter file. I began by defining the Person class according to the instructions. This step largely followed the same work we did in the lab. It also adds an additional check on value errors, as the property is now defined up top and will not allow the user to continue using the program with incorrect inputs. For the Student class, I set it up to inherit the properties from the Person class, and added a definition for the course_name variable. I briefly had an issue because I forgot the double underscores in the getter but I added them. (**Figure 1**)

```
# TODO call to the Person constructor and pass it the first_name and last_name data (Done)
    def __init__(self, first_name: str = '', last_name: str = '', course_name: str = ''):
        super().__init__(first_name=first_name, last_name=last_name)

# TODO add a assignment to the course_name property using the course_name parameter (Done)
        self.course_name = course_name

# TODO add the getter for course_name (Done)
    @property
    def course_name(self):
        return self.__course_name.title()

# TODO add the setter for course_name (Done)
    @course_name.setter
    def course_name(self, value: str):
        self.__course_name = value

# TODO Override the __str__() method to return the Student data (Done)
    def __str__(self):
        return f'{self.first_name},{self.last_name},{self.course_name}'
```

Figure 1: Student Class

Upon testing the script, I encountered some errors with the wording of the file reader function. Apparently, it could not read the file since the 'file' local variable was not previously defined. I was unsure why this issue only occurred after adding in the object classes (it worked fine in the starter), but I reoriented the function to use the 'with' statement, which cleared up the error. **(Figure 2)**

```
try:
    with open(file_name, "r") as file:
        student_data = json.load(file)
except Exception as e:
    IO.output_error_messages(message="Error: There was a problem with reading the file.", error=e)
finally:
    if 'file' in locals() and not file.closed:
        file.close()
return student_data
```

Figure 2: Reorganized File Reader

Summary

I wasn't fully sure of all the reasons for why the components for object classes are set up the way they are in Python, but I understand the reasoning for using them. The ability to establish your object properties up top and even change them seems like it would be a powerful tool, especially with a really complex code. I also recognize the usefulness in the Git function in PyCharm, even if we aren't using it in the assignment. Version control is always important with complex, frequently updating programs.