Andrea Martinez

November 27, 2024

Python 100

Assignment 07

GitHub URL: https://github.com/martia72/IntroToProg-Python-Mod07.git

Assignment 07 – Programming with Python Classes and Objects

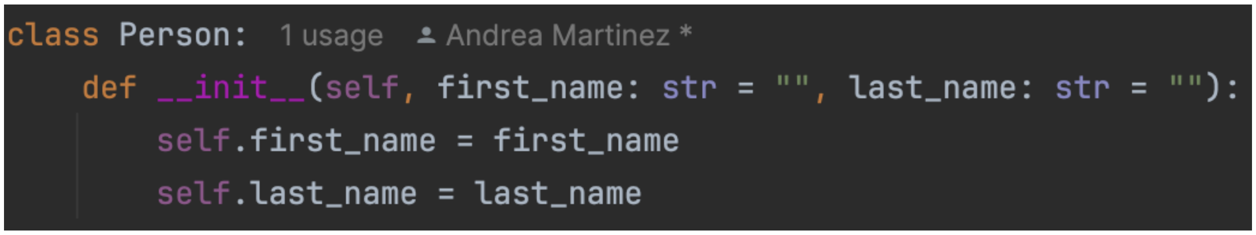
Introduction

In module 07 of this course, one of the main objectives was to use classes and objects. In this document, I'll briefly describe my initial thoughts about using these two new methods of coding. In addition, I'll describe the steps taken to use inheritance from parent classes. I'll also show how constructors can be used when an object of a class is created. Finally, I will mention how implementing classes and functions can be a powerful tool for reusability for large programs.

Classes and Objects

In this module, I created two classes named Person and Student. Student is the child class, while the Person is the parent class because the Student class is defined ad class Student(Person) in the script. In Python, a child class can inherit attributes and methods from a parent class. This allows to reuse code and create specialized classes based on a more general class (https://www.codecademy.com/resources/docs/python/inheritance , 2024). In object-oriented programming (OOP), an object is an instance of a class. In this case, Figure 1 shows the creation of two objects: first_name and last_name respectively. One can see that the constructor method is used since an object of the Person class was created. The 'self' keyword was also used because I applied the constructor method. According to the Module 07 notes, the keyword refers to data or functions found in an object instance and not directly in the class.

Additionally, I also applied the concept of overriding methods in this script by using super(). In python, In Python, the super() function is used to call a method from a parent class. It's particularly useful when you're working with inheritance and want to extend the functionality of a method defined in the parent class.

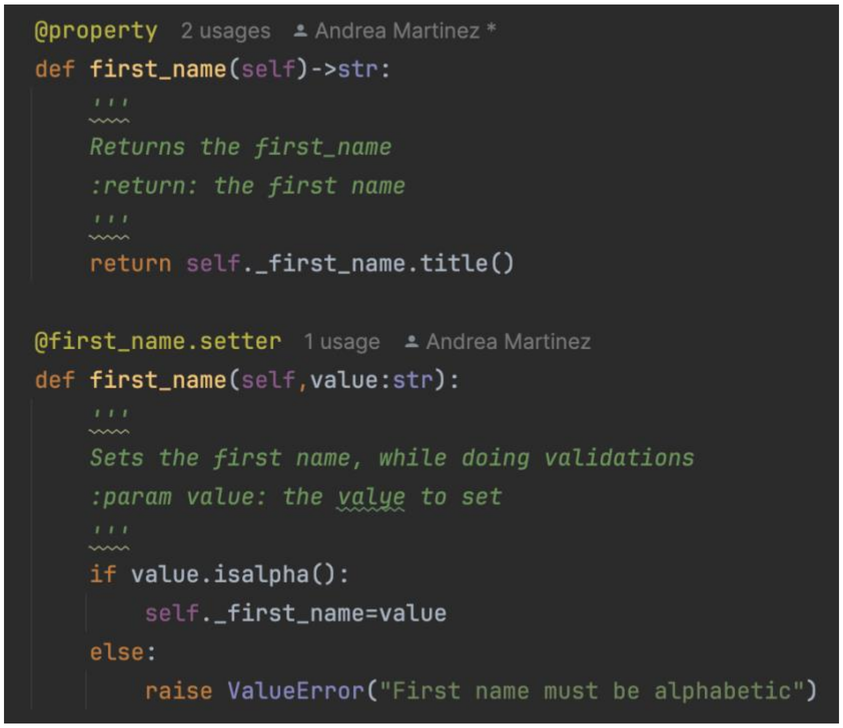


```
class Person:  1 usage  Andrea Martinez *
    def __init__(self, first_name: str = "", last_name: str = ""):
        self.first_name = first_name
        self.last_name = last_name
```

Figure 1: Object creation with the constructor method

Properties

I used properties in this code to manage attribute data. For each attribute I had (first_name, last_name, and course_name), I created two properties, one for getting and one for setting. The more formal names for getter and setters are ‘accessors’ or ‘mutators’. I started out the getter property function using the @property decorator, which allows me to implement additional logic or validation if needed. In my case, I used the title() statement to make sure that the first letter of both the first and last name were capitalized in case the user entered the names all in lower case. On the other hand, the setter property allows the developer to add additional validation of the data such as error handling. In my case, I added the if-statement to ensure that the data entered was alphabetic and did not contain any typos. The setter property must include the property.setter decorator like shown in figure 2.



```
@property  2 usages  Andrea Martinez *
def first_name(self)->str:
    '''
    Returns the first_name
    :return: the first name
    '''
    return self._first_name.title()

@first_name.setter  1 usage  Andrea Martinez
def first_name(self,value:str):
    '''
    Sets the first name, while doing validations
    :param value: the valye to set
    '''
    if value.isalpha():
        self._first_name=value
    else:
        raise ValueError("First name must be alphabetic")
```

Figure 2: Use of both the setter and getter properties

Testing the Program

As a developer, I had to test my code to make sure that it was doing what it was supposed to by meeting all the requirements set by the instructor. Writing this code was very interesting because I learned that defining functions and classes at the start of the code do not get executed unless I call them later in the script. This is a very powerful way to script because I can reuse functions in other scripts as long as I import them or call them in the code. Avoiding repeated lines of code can help save a lot of time and memory space in a computer when running long scripts. The script below in Figure 3 shows that the execution portion of the code can be shorten to roughly 30 lines when it used to be approximately 80 in the module 05 assignment. It also looks far cleaner and more organized to have a concise execution code that calls functions that are created earlier in the script.

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

# When the program starts, read the file data into a list of lists (table)
# Extract the data from the file
students = FileProcessor.read_data_from_file(file_name=FILE_NAME,
student_data=students)
print(students)
# Present and Process the data
while (True):

    # Present the menu of choices
    IO.output_menu(menu=MENU)

    menu_choice = IO.input_menu_choice()

    # Input user data
    if menu_choice == "1":  # This will not work if it is an integer!
        students = IO.input_student_data(student_data=students)
        continue

    # Present the current data
    elif menu_choice == "2":
        IO.output_student_and_course_names(students)
        continue

    # Save the data to a file
    elif menu_choice == "3":
        FileProcessor.write_data_to_file(file_name=FILE_NAME,
student_data=students)
        continue

    # Stop the loop
    elif menu_choice == "4":
        break  # out of the loop
    else:
        print("Please only choose option 1, 2, or 3")

print("Program Ended")
```
Figure 3: Execution code reduced to 30 lines

Summary

In conclusion, the planning, writing, and execution of this code taught me how to properly use object-oriented programming while making the Student and Person classes, and how I can extend one class from the other by making a parent-child class connection. I strengthened my skills I learned in Module 06 writing classes and functions. I learned to apply inheritance and how it helps the developer to reuse code or even extend on it. While making classes and objects, I also learned the power of using constructors to initialize data. Lastly, I learned how to make commits to GitHub directly from PyCharm, which provides a more efficient version-control process.