Assignment07_David_Levinson

David Levinson
9/11/2024
Foundations of Programming: Python
Assignment 07

https://github.com/dl-11/IntroToProg-Python-Module7.git

Introduction

This week, again, the topics were quite big conceptually. I wanted the course lectures and videos multiple times. Some of the key topics covered were statements, functions, and classes and how you can group organize and reuse code more efficiently.

We spent most of the lesson focused on learning more about objects and classes. Classes are templates for creating objects. They define the attributes and methods shared by all objects of that class. Objects are instances of a class; each object has its own memory space and stores its own data. There are data classes and processing classes. Data Classes focus on storing data, while Processing Classes perform operations or tasks

There are 3 key components in Classes:

- Attributes: Variables that hold data specific to an object.
- Constructors: Special methods used to initialize objects (`__init__` in Python).
- Properties: Methods used to manage access to attributes (getters and setters).

Lastly, we learned about using Github desktop as a visual tool for managing Git repositories and integrating changes with GitHub. With that, I felt it was time to jump into code and help me better understand the concepts through using them.

Starting Week 7 Code

This week's starter file included a handy to do comment as shown in Figure 1! It did not show up in Pycharm's Todo tool window for some reason. Will need to figure that out later.

```
# TODO Create a Person 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
# 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: TODO Comment in Week 7 starter code

Following the to-do list, I started with the Person Class. In our case, the person class will have first name and last name and the student subclass will hold the course. I created the class Person and assigned two attributes, first name and last name. Attributes are used to store the state or characteristics of an object as shown in Figure 2.

```python
# TODO Create a Person Class
class Person:
    """
    A collection of data about a Person
    David Levinson, 9/8/2024, Created Class
    Properties:
    - first_name (str): The student's first name.
    - last_name (str): The student's last name.
    """
```

Figure 2: The Person class with 2 attributes

Constructor

Next, I create the constructor portion of the Person class as shown in Figure 3. Constructors are special methods that automatically run when you create an object from a class

```python
def __init__(self, first_name: str = "", last_name: str = ""):
    self.first_name = first_name
    self.last_name = last_name
```

Figure 3: Constructor method for the Person class

Getters and Setters

Getters and Setters are used to control access to attributes in a class. They are methods that manage the retrieval (getting) and modification (setting) of class attributes. I built out the getters and setters for the student first and last name as shown in Figure 4 below.

```python
@property
def student_first_name(self):
    return self.__first_name.title()

@student_first_name.setter
def student_first_name(self, value: str):
    if value.isalpha() or value == "":
        self.__first_name = value
    else:
        raise ValueError("The first name should not contain numbers.")

@property
def student_last_name(self):
    return self.__last_name.title()

@student_last_name.setter
def student_last_name(self, value: str):
    if value.isalpha() or value == "":
        self.__last_name = value
    else:
        raise ValueError("The last name should not contain numbers.")
```

Figure 4: Getter and Setters for Student first and last name properties

Class Inheritance

Subclasses inherit all the attributes and methods of a higher-level main class, or parent class, which is usually referred to as the base class. A subclass is defined with no indentation, because the subclass isn't a part of, or constrained within, the base class.

Subclass references the base class, in this case Person class as shown in Figure 5 below.

```python
# Student Class
class Student(Person):
    """
    A collection of data about a Student
    Properties:
    - student_first_name (str): The student's first name inherited from the Person class
    - student_last_name (str): The student's last name inherited from the Person class
    - course_name (str): The course the student is registering/registered for
        ChangeLog: (Who, When, What)
    David Levinson,9/9/2024, Created Class
    """
    # Call to the person constructor and pass it the first name and last name data plus course name
    def __init__(self, first_name: str = '', last_name: str = '', course_name: str = ''):
        super().__init__(first_name=first_name, last_name=last_name)
        self.__course_name = course_name  # Initialize the private variable

    # - - Getter for Course Name property
    @property
    def course_name(self):
        return self.__course_name.title()  # Getter with formatting

    # - - Setter for Course Name property
    @course_name.setter
    def course_name(self, value: str):
        self.__course_name = value
    #Override the __str__() method to return the student data
    def __str__(self):
        return f"{self.first_name}, {self.last_name}, {self.course_name}"
```

Figure 5: Student subclass. Adds in an extra parameter for Course Name in the subclass.

Logical error troubleshooting

With the addition of this section, I realized the code was not printing out the Course_Name input as expected. See Figure 6 to see it printing out the memory address

```
Enter your menu choice number: 2
-------------------------------------------------
John, Doe, <built-in method title of str object at 0x00000186F48DDF70>
Jane, Smith, <built-in method title of str object at 0x00000186F48DDFC0>
Alice, Johnson, <built-in method title of str object at 0x00000186F48F90B0>
Sally, Smith, <built-in method title of str object at 0x00000186F48F9130>
Mike, Jones, <built-in method title of str object at 0x00000186F48F91B0>
Bob, Jones, <built-in method title of str object at 0x00000186F48F8FB0>
```

Figure 6: Not printing the course name as I intended

I spent over an hour looking at the code back and forth and going back to my notes. I had the string override method defined. It was baffling! In the end, I realized I had missed the parentheses in the getter for course name as shown in figures 7 & 8 with the correction. This simple mistake took an hour of my life, but I won't make it again! ha!

I was not calling the method, just using the name of it.

```
@property
def course_name(self):
    return self.__course_name.title
```

Figure 7: Missing parenthesis in my code

corrected to

```
return self.__course_name.title()
```

Figure 8: Fixed and calling the getter method properly

Conclusion

With that, I updated the main body of code to call the various methods within the classes and got the code working. It was a big learning week and I intend to watch the videos again to cement in these new concepts further. It was helpful to work with code this week and learn through doing. I used Github similar to last week as a safety net if I needed to revert. I also appreciated the feedback I received on the homework as I would have missed a property name this time had I not paid extra attention to it.