

Anthony Cabral

11/27/2024

IT FDN 110 A Au 24

Module 7

<https://github.com/Gamezpd/IntroToProg-Python-Mod07>

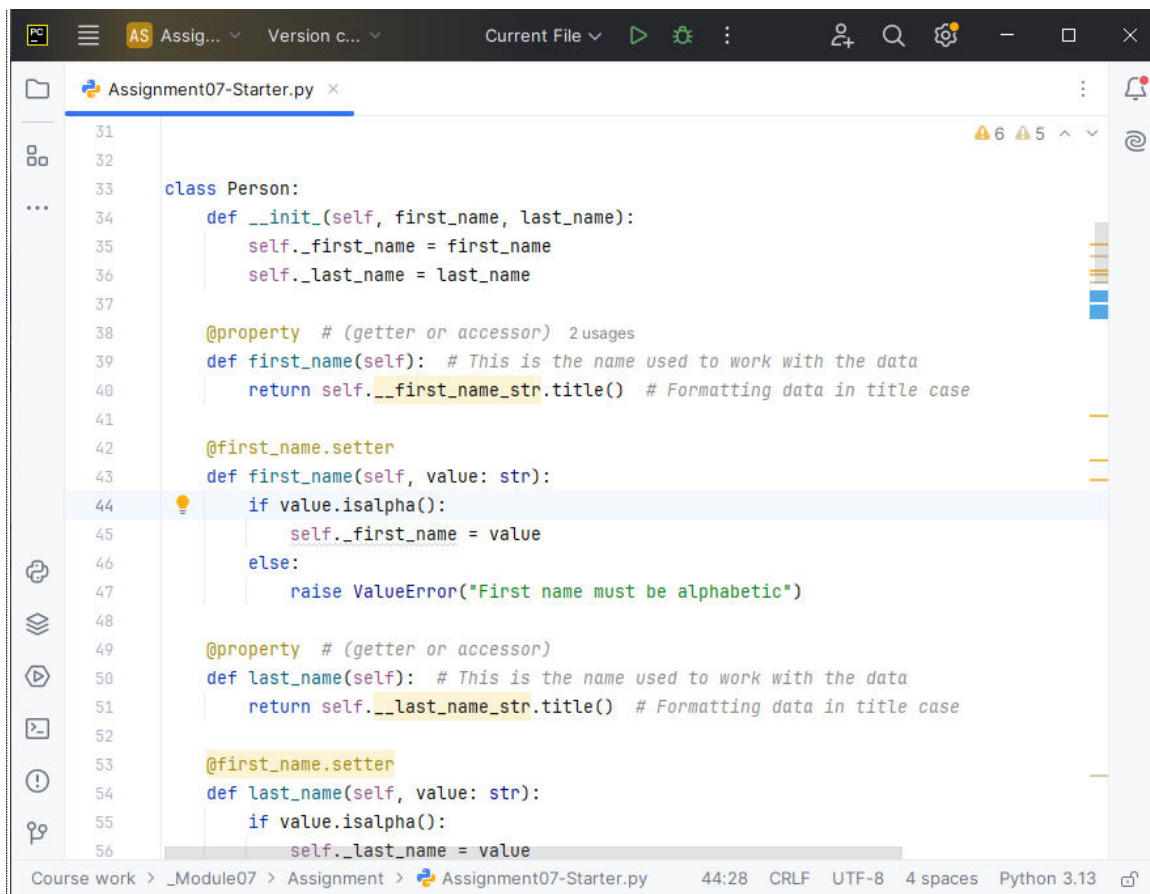
Module 7 Breakdown

Process Introduction

Module 7. I add the tasks for easy reference, not to pad my assignment write up. Just a heads up for this portion I did start with a student class first, wrote it into a functional Person class, then just changed the name. By screenshot 3 I post it and also have fixed it by then.

```
# TODO Create a Person Class
# TODO Add first_name and last_name properties to the constructor (Done)
# TODO Create a getter and setter for the first_name property (Done)
# TODO Create a getter and setter for the last_name property (Done)

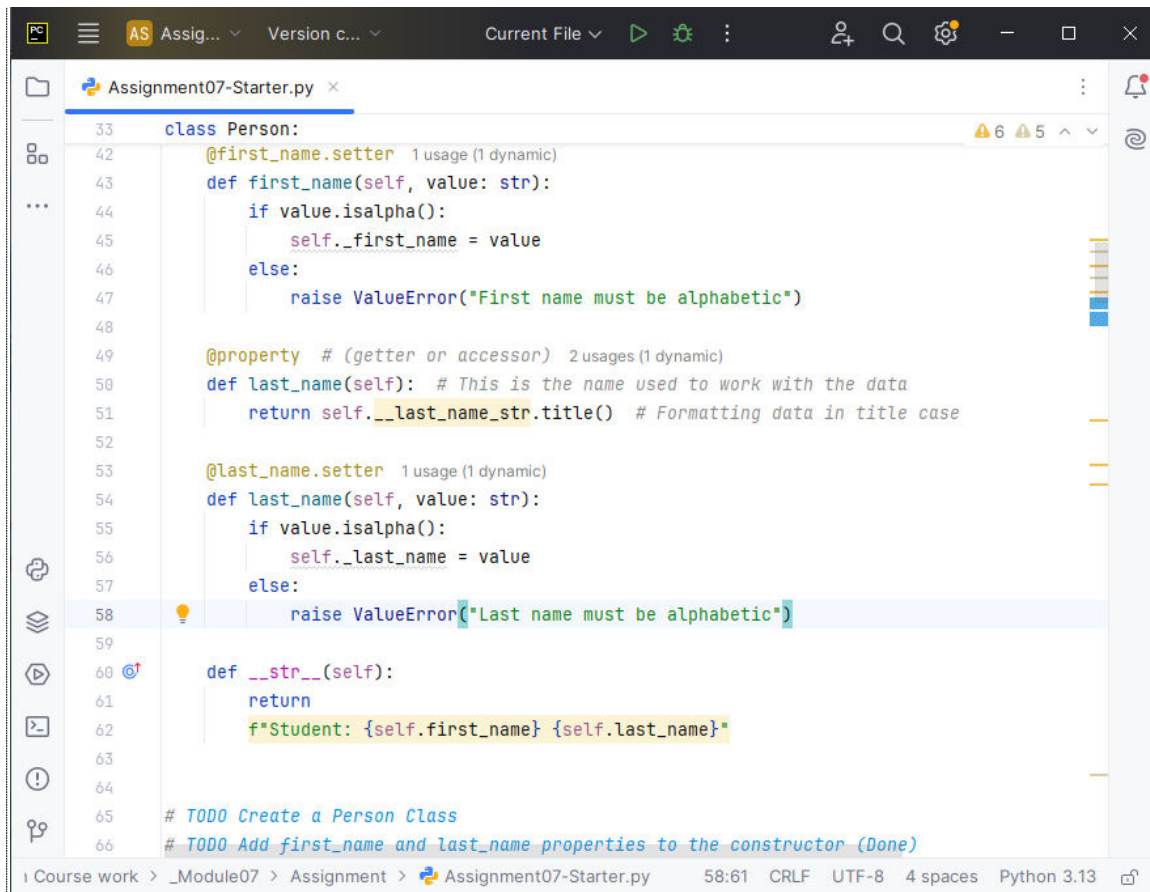
# TODO Override the __str__() method to return Person data (Done)
# TODO Create a Student class the inherits from the Person class (Done)
# TODO call to the Person constructor and pass it the first_name and last_name data (Done)
# TODO add a assignment to the course_name property using the course_name parameter (Done)
# TODO add the getter for course_name (Done)
# TODO add the setter for course_name (Done)
# TODO Override the __str__() method to return the Student data (Done)
```



```
31
32
33 class Person:
34     def __init__(self, first_name, last_name):
35         self._first_name = first_name
36         self._last_name = last_name
37
38     @property # (getter or accessor) 2 usages
39     def first_name(self): # This is the name used to work with the data
40         return self._first_name_str.title() # Formatting data in title case
41
42     @first_name.setter
43     def first_name(self, value: str):
44         if value.isalpha():
45             self._first_name = value
46         else:
47             raise ValueError("First name must be alphabetic")
48
49     @property # (getter or accessor)
50     def last_name(self): # This is the name used to work with the data
51         return self._last_name_str.title() # Formatting data in title case
52
53     @first_name.setter
54     def last_name(self, value: str):
55         if value.isalpha():
56             self._last_name = value
```

Course work > _Module07 > Assignment > Assignment07-Starter.py 44:28 CRLF UTF-8 4 spaces Python 3.13

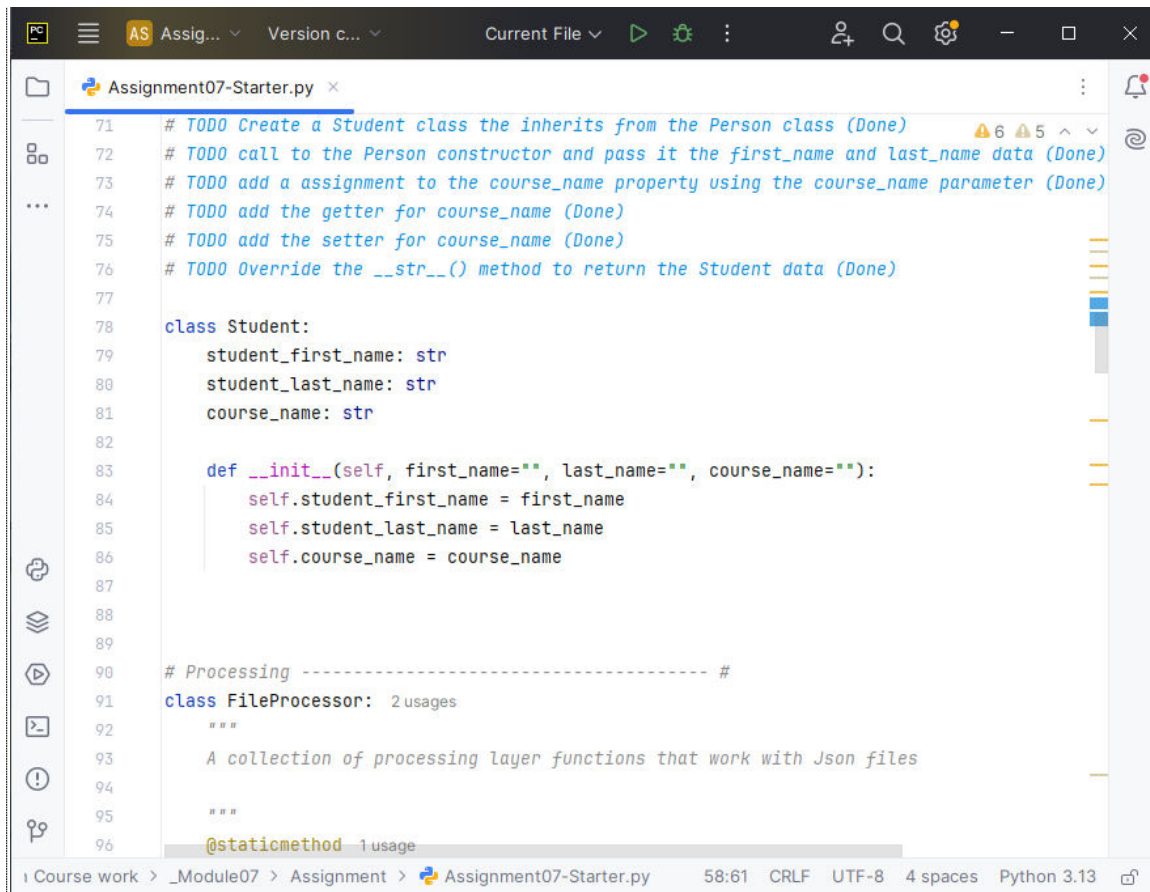
Person class. Created the function, double underscore to make it private, first name and last name properties given to the constructor, this "init" method. Getter and setter set for both first and last name. And I did catch the @setter for last_name was incorrect. Fixed it slightly later. Also here is the rest of the class:



```
33 class Person:
42     @first_name.setter 1 usage (1 dynamic)
43     def first_name(self, value: str):
44         if value.isalpha():
45             self._first_name = value
46         else:
47             raise ValueError("First name must be alphabetic")
48
49     @property # (getter or accessor) 2 usages (1 dynamic)
50     def last_name(self): # This is the name used to work with the data
51         return self._last_name_str.title() # Formatting data in title case
52
53     @last_name.setter 1 usage (1 dynamic)
54     def last_name(self, value: str):
55         if value.isalpha():
56             self._last_name = value
57         else:
58             raise ValueError("Last name must be alphabetic")
59
60     def __str__(self):
61         return
62         f"Student: {self.first_name} {self.last_name}"
63
64
65 # TODO Create a Person Class
66 # TODO Add first_name and last_name properties to the constructor (Done)
```

Course work > _Module07 > Assignment > Assignment07-Starter.py 58:61 CRLF UTF-8 4 spaces Python 3.13

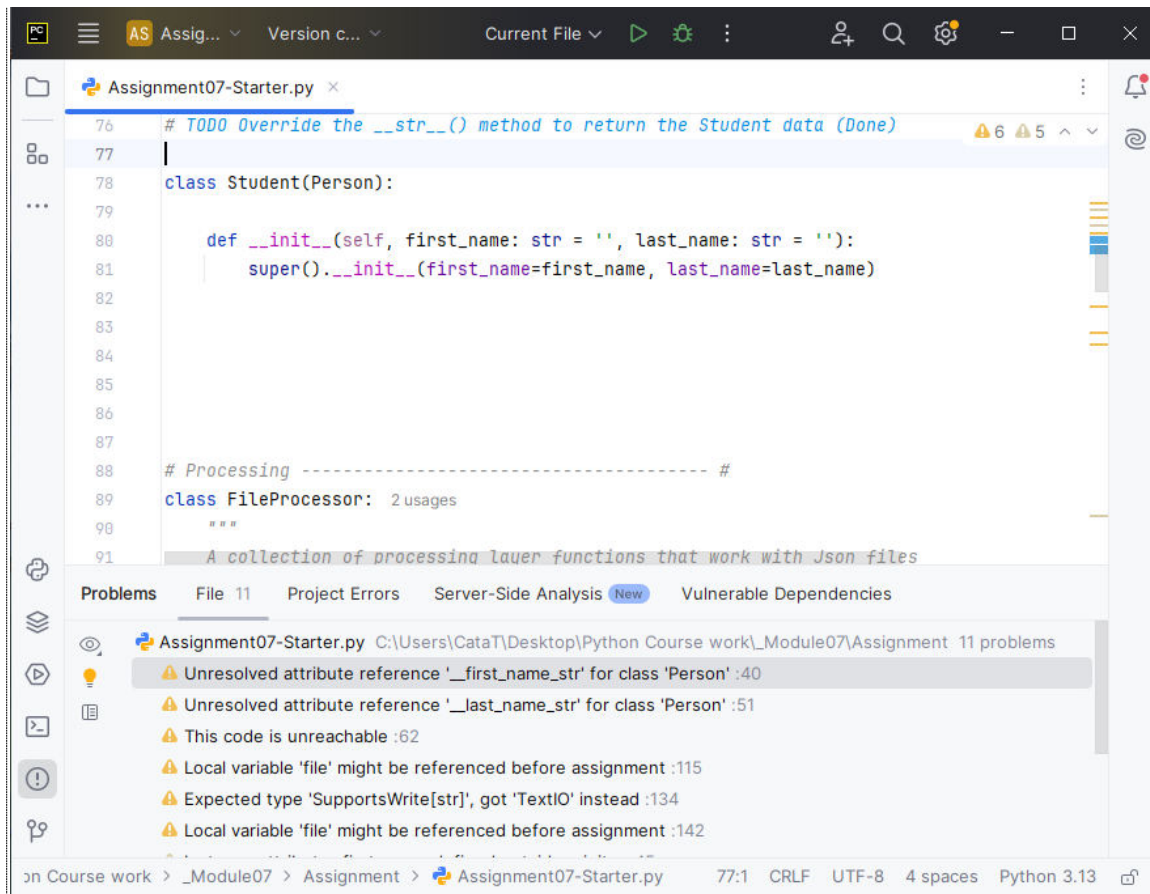
I added the string override to line 62. At least that's my understanding of the override. Basically the method is built in and would print say, the location of memory holding whatever you're pushing. But now that it is going to print Student: Vic Vu, for example. The next instance of override will pull from this class with the "super" call to constructor, but we're getting ahead of ourselves. I already made the student class ahead of time to hold its spot.



The screenshot shows a code editor window with the file name "Assignment07-Starter.py". The code is written in Python and includes several TODO comments and two class definitions. The first class is "Student", which has attributes for first name, last name, and course name, and an __init__ method. The second class is "FileProcessor", which has a docstring and a @staticmethod decorator. The editor interface includes a sidebar with icons for file explorer, search, and other tools, and a status bar at the bottom showing the current file path, line number, and encoding.

```
71 # TODO Create a Student class the inherits from the Person class (Done)
72 # TODO call to the Person constructor and pass it the first_name and last_name data (Done)
73 # TODO add a assignment to the course_name property using the course_name parameter (Done)
74 # TODO add the getter for course_name (Done)
75 # TODO add the setter for course_name (Done)
76 # TODO Override the __str__() method to return the Student data (Done)
77
78 class Student:
79     student_first_name: str
80     student_last_name: str
81     course_name: str
82
83     def __init__(self, first_name="", last_name="", course_name=""):
84         self.student_first_name = first_name
85         self.student_last_name = last_name
86         self.course_name = course_name
87
88
89
90 # Processing ----- #
91 class FileProcessor: 2 usages
92     """
93     A collection of processing layer functions that work with Json files
94
95     """
96     @staticmethod 1 usage
```

I made this first, as it was supposed to be the Person class but we can fix it now:

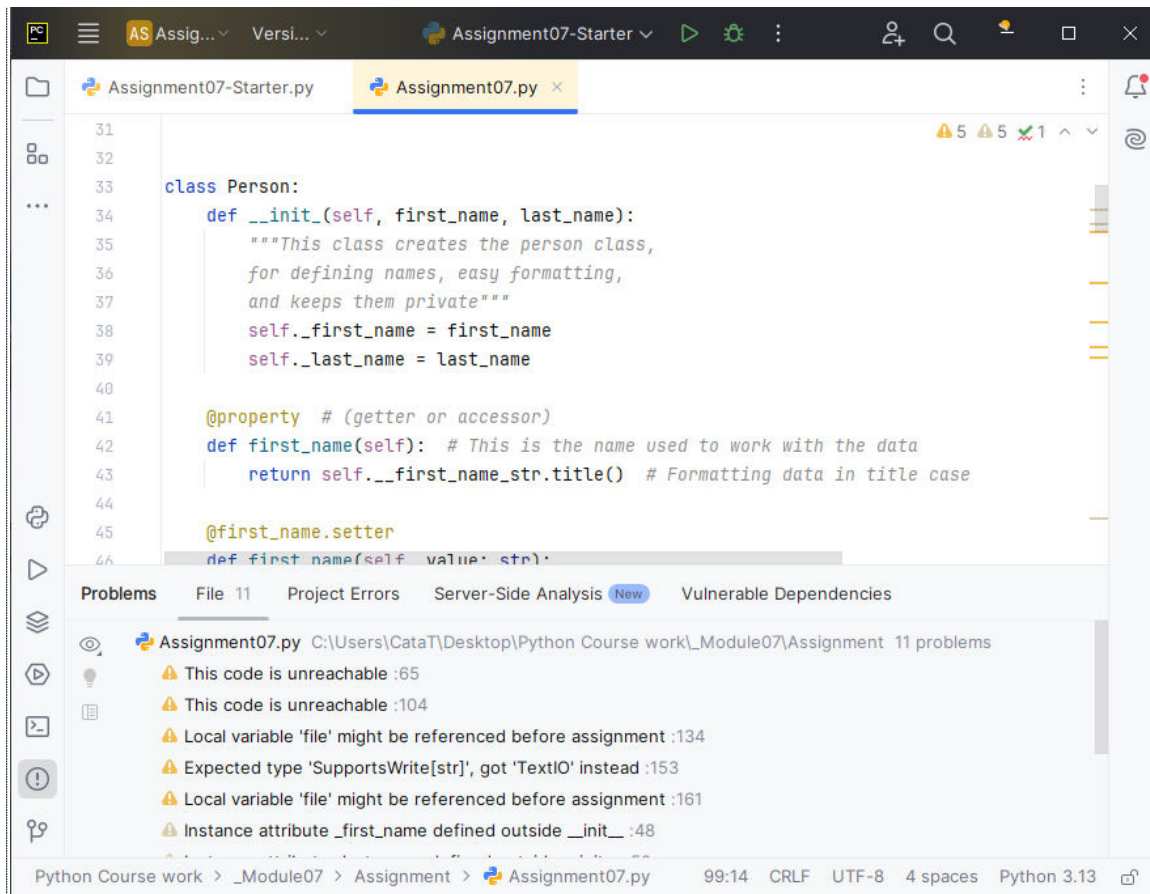


So now it calls to the Person class. The next task is to add in course names. So, can I just `self.course_name` into the Student class? I am a little confused about the "add a assignment to `course_name` property" wording. Is adding an assignment a function I missed, or does that mean just assigning a `course_name` attribute value? I hope its the latter, this module in particular is a bit rough to wrap the head around.

So now the Student class is done:

```
Assignment07-Starter.py
73 # TODO add a assignment to the course_name property using the course_name parameter
74 # TODO add the getter for course_name (Done)
75 # TODO add the setter for course_name (Done)
76 # TODO Override the __str__() method to return the Student data (Done)
77
78 class Student(Person):
79     def __init__(self, first_name: str = '', last_name: str = '', course_name: str = ''):
80         super().__init__(first_name=first_name, last_name=last_name)
81
82         self.course_name = course_name
83
84     @property # (getter or accessor) 2 usages (1 dynamic)
85     def course_name(self): # This is the name used to work with the data
86         return self.__course_name_str.title() # Formatting data in title case
87
88     @course_name.setter 2 usages (1 dynamic)
89     def course_name(self, value: str):
90         if value.isalpha():
91             self.__last_name = value
92         else:
93             raise ValueError("Course name must be alphabetic")
94
95     def __str__(self):
96         return
97         f"Student: {self.first_name} {self.last_name} is in {self.course_name}"
98
```

All the tasks are done. Thats module 7. Oh wait I missed the "All classes include descriptive document strings" requirement. Lets fix that:



Added some descriptive comments for the Student class as well. Getters and Setters, are they technically methods? I think so, but they just are also just, the Getters and Setters, so I'll leave them alone.

