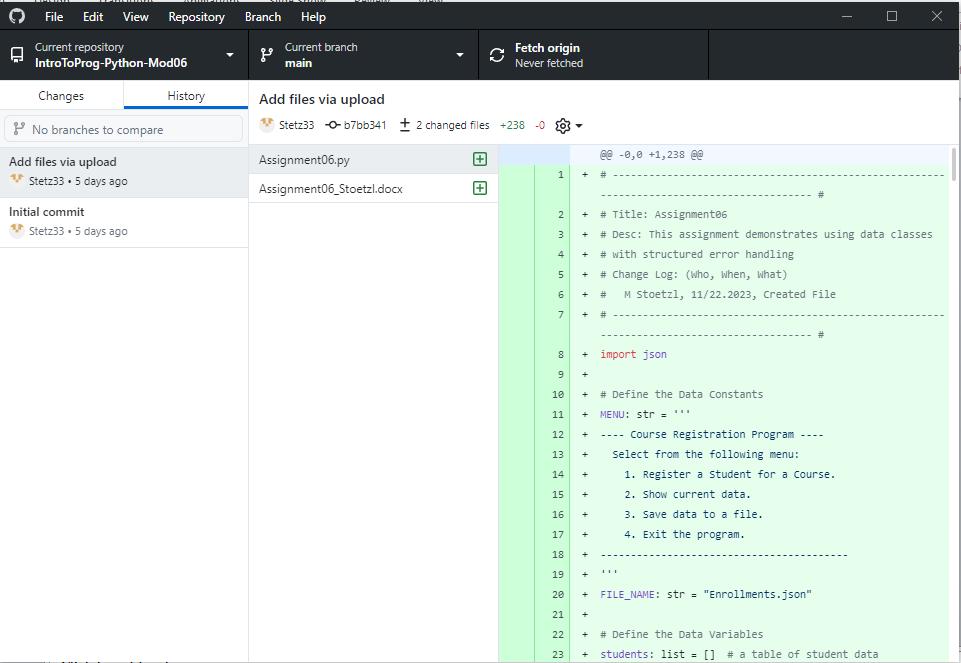
Mark Stoetzl  
11/27/2023  
IT FDN 110 AAu 23:Foundations Of Programming Python  
Assignment 07

GitHub repository link: https://github.com/Stetz33/IntroToProg-Python-Mod07

Screenshot of GitHub Desktop created



Creating Python Program Assignment 07

Introduction  
Assignment #7 takes the concept of classes and expands on it. A specific class is created for person along with a specific class for student. An important aspect of this is that the student class inherits the attributes from the parent “person” class. This allows python code to be even more flexible and powerful.

Step 1  
The first step for assignment #7 is to add two specific classes to the code of assignment #6 to handle the data of person and student. The “person” class contains the typical information that would be gathered for any person, such as first name and last name. The second class is “student” which has its own unique information that pertains only to someone who is a student. In this case that is the enrolled class for each student. The first name and last name information is inherited from the person class into the student class. Each class is defined by using constructors for the first and last names as self.XXXX . An example of this is shown below in illustration #1. In this case it is for the person class.

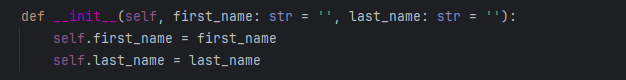


Illustration #1 – example of self.first\_name and self.last\_name

Step 2

The second step for creating individual classes for person and the inherited class of students is to change the way the way a method works by overriding. The overriding method has the same name and parameters as the original method, however in this case the properties of \_\_str\_\_(self) that were originally defined are overridden to return the student data. An example of this is shown below in illustration #2 below

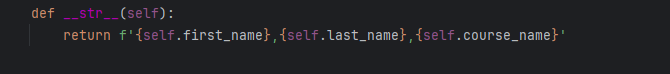


Illustration #2 – example of overriding

Step 3

Another powerful aspect of using defined methods for specific data is that the data’s attributes can be easily modified using getters and setters. Typically, getter methods return the target attribute’s value, while setter methods take a new value and assign it to the underlying attribute. In the example shown in illustration #3 below the course\_name’s attributes are set using the following code:

“def course\_name(self):

return self.\_\_course\_name

The setter as shown below in illustration #3 below can be used to assign error handling and code validation sections that are applicable to all of the lists contained within the student class.

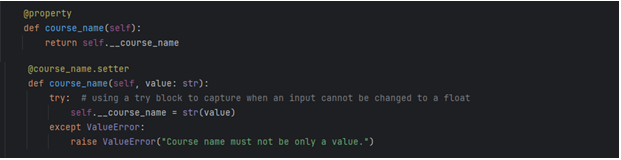


Illustration #3 – example of getter and setters

Summary  
Classes provide a means of bundling data and functionality together. Creating a new class creates a new type of object, allowing new instances of that type to be made. Each class instance can have attributes attached to it for maintaining its state. Class instances can also have methods (defined by its class) for modifying its state.