Deric Fetters

6/14/2024

IT FDN 110 A

Assignment 07

Classes and Objects

Introduction

Module 5 most notably introduces working with class attributes, constructors and inherited code. These topics improve the functionality, usability and shareability of our code.

Classes

Classes were utilized in this assignment to create objects representing people. This enabled information about a particular student to be isolated and worked with through methods and attributes. Attributes are variables that hold data about an object. The attributes “first_name”, “last_name” and “course_name” were used. See Figure 1.

```
class Person:
    """This class creates a person object..."""
    def __init__(self, first_name: str = "", last_name: str = ""):...

    7 usages
    @property #Getter
    def first_name(self):...

    5 usages
    @first_name.setter
    def first_name(self, value):...

    7 usages
    @property #Getter
    def last_name(self):...

    5 usages
    @last_name.setter
    def last_name(self, value):...

    def __str__(self):...

11 usages
class Student(Person):

    """This class creates a Student object from the Person Class..."""

    def __int__(self, first_name: str = "", last_name: str = "", course_name: str = ""):...

    3 usages
    @property
    def course_name(self):...

    2 usages
    @course_name.setter
    def course_name(self, value):...

    def __str__(self):...
```

Figure 1: Assignment07.py Class Attributes (some code hidden)

Constructors

Constructors are methods, or functions within a class, that are executed every time and object is created from a class. The constructors were used to set initial values of attributes. In Python, constructors are named "__init__". See Figure 2.

```
1 usage
class Person:
    """This class creates a person object..."""
    def __init__(self, first_name: str = "", last_name: str = ""):
        self.__first_name = first_name
        self.__last_name = last_name
```

Figure 2: Assignment07.py Constructors

Inherited code

Inherited code refers to classes retaining properties and method from the existing classes they are made from. This concept was used when creating the Student() class from the Person() class:

Class Student(Person):

The student() class added the "course_name" attribute to the inherited "first_name" and "last_name" attributes from the person class. The "super" method was used to call the Person() class constructor to initialize the two inherited attributes. See figure 3.

```
class Student(Person):

    """ This class creates a Student object from the Person Class

    ChangeLog: (Who, When, What)
    DFetters,6.12.2024,Created function

    """

    def __int__(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

    3 usages
    @property
    def course_name(self):
        return self.__course_name()

    2 usages
    @course_name.setter
    def course_name(self, value):
        self.__course_name = value

    def __str__(self):
        return f"{self.first_name},{self.last_name}"
```

Figure 3: Assignment07.py Inherited Code

GitHub

A GitHub account with a repository for classwork was created. It can be found at:

https://github.com/fetterdl/IntroToProg-Python-Mod07

Summary

In Module 7, I learned about working with class attributes, constructors and inherited code. All of these topics improve the functionality, usability and shareability of our code.