Name: Alex Kover
Date: 06/01/24
Course: Introduction to Programming with Python
GitHub: https://github.com/AKover-UW/IntroToProg-Python-Mod07

Assignment 07: Classes and Objects

Introduction

This assignment builds upon the concept of classes that were introduced in the previous assignment and adds data classes, with unique features such as attributes, constructors, and properties, along with inherited code and overriding to provide more flexibility and organization.

Data Classes

In the previous assignment, classes such as FileProcessor and IO handled the processing and input and output in the code, respectively. In this assignment, data classes that manage data are introduced. Data classes differ from processing and IO classes in that in addition to using methods, they also use attributes, constructors, and properties, which I will cover in the next topic. In this program, the Person and Student classes are custom classes that organize data into two different groups: persons and students.

```
# Data Classes ------------------------------------ #
1 usage
class Person:...


class Student(Person):...
```

Figure 1: Data classes used in this program: Person and Student (which builds on the data in the Person class)

Attributes, Constructors, and Properties

Classes that process data or handle input/output utilize methods to perform their actions, but data classes can also use attributes, constructors, and properties in addition to methods. Attributes are variables that hold specific data for an object, such as the first and last name of a person. Constructors are special methods that is called when an object is created and sets that object's attributes upon creation. They assign initial values to the attributes for each object and are often called initializers for this reason. Properties are functions that are designed to handle attribute

data. For each attribute, two properties are created known as “Accessors” and “Mutators”, but more commonly as “getters” and “setters”. Getter properties enable you to access data and apply formatting, while setter properties allow you to add data validation and error handling.

```
class Person:  # TODO Create a Person Class (Done)

    # TODO Add first_name and last_name properties to the constructor (Done)
    # constructor
    # adds first and last name of the person
    def __init__(self, first_name: str = "", last_name: str = ""):
        self.first_name = first_name
        self.last_name = last_name
```

Figure 2: Constructor for the Person class initializes the first_name and last_name attributes

Inherited Code and Overriding

Inheritance is a concept where a new class can use or “inherit” data and behaviors from other existing classes. This relationship is commonly referred to as parent and child classes. In many programming languages, there is a base “object” class that all other classes inherit code from. In this program, the Student Class inherits the code of the Person class and builds on it. Inherited methods can also be changed, or overridden. For example, the __str__() method will return the address of an object, but when inherited by another class it can be modified to return the values of the properties of the classes instead.

```
# TODO Override the __str__() method to return Person data (Done)
# overrides the string method and returns the person's first and last name
def __str__(self):
    return f"{self.first_name},{self.last_name}"
```

Figure 3: Overriding the string method to return the first and last name properties from the Person class

Summary

Data Classes and their attributes, constructors, properties, along with inherited code and overriding are all concepts that allow us to build on the ideas from the previous assignment and build a more organized and flexible program.