Name: Andrew Wysocki
Date: 02/28/24
Course: Introduction to Programming - Python
https://github.com/jaspercasidino/IntroToProg-Python-Mod07

Assignment 07 - Classes and Objects

Introduction

This week's lesson covers Classes and Objects and posting directly to GitHub.

Topic - Classes and Objetcs

Classes are a way of grouping variables, constants and functions by collecting them under a common name and utilizing that name when trying to use these variables, constants and functions. Classes are a cornerstone of Object Oriented Program and can be powerful tools that help simplify reading and creating code through the power of grouping. In the module this week, we defined 2 types of classes: one for input/output manipulation of files and one for processing data.

To create a class, you simply write *class* and the name that you want for the class. Then under the class name, you can create functions or variables that will be utilized when working with the class. To define a function, you type *@staticmethod* then return followed by the function that you would like to define.

To call a function that is part of a class, you call out the class name followed by a dot followed by the function name (complete with arguments if the function requires it).

You need to initialize the class with a constructor using the __init__(self) function. You can add additional parameters to help initialize your variables. The constructors are only used once.

Example:

```
class Person:

    # TODO Add first_name and last_name properties to the constructor (Done)
    new *
    def __init__(self, first_name: str = "", last_name: str = ""):
        self.__first_name = first_name
        self.__last_name = last_name
```

To reference the internal variables that define the class, you need to establish getters and setters. Getters grab the information in the variables from the object and setters set the variables given an input of some type. To make a getter you use the @property identifier and then followed by def then the attribute followed by (self). From there you can return the value of the variable and provide formatting should you desire.

Example:

```
@property
def first_name(self) -> str:
    return self.__first_name.capitalize()
```

For Setters, you use @attribute_name.setter followed by def attribute_name(self, parameter1, parameter2, etc.). You can even do error checking within the setter to ensure that the value you are setting makes sense within the scope and usage of your program,

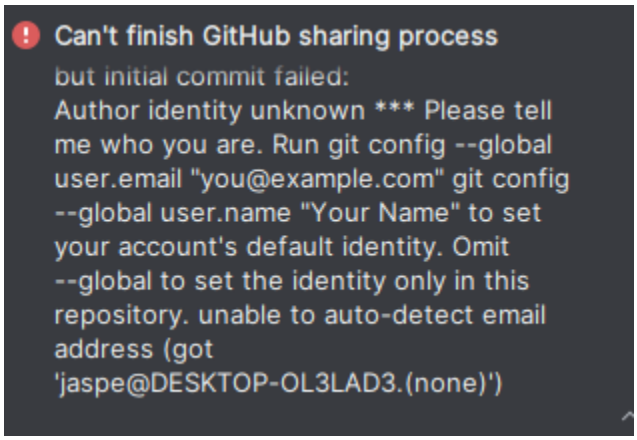
Example:

```
@first_name.setter
def first_name(self, value: str = "") -> None:
    if value.isalpha() or value == "":  # Optional validation code
        self.__first_name = value
    else:
        raise ValueError("Error! The first name should not contain numbers!")
```

You can also use 2 underscores when defining your attributes and that will protect those attributes from being manipulated or changed outside of the class functions.

Topic - GitHub

You can post directly to GitHub through PyCharm which can help with pushing code and making backups. To do so, you select your project and Share it via the GitHub option. You create a repository and Share. You will need to log into GitHub and then post the files you want. I had a lot of difficulty because it kept failing to upload the initial commit. I got the following error message and gave up on the initial commit. I had no idea what it meant. I was able to push my files via the Commit and Push function when it asked for my name and email. Looks like it finally took.



Summary

This module covered the basic foundations of functions and classes.