James Eardley

March 01, 2024

IT FDN 100 A

Assignment 07

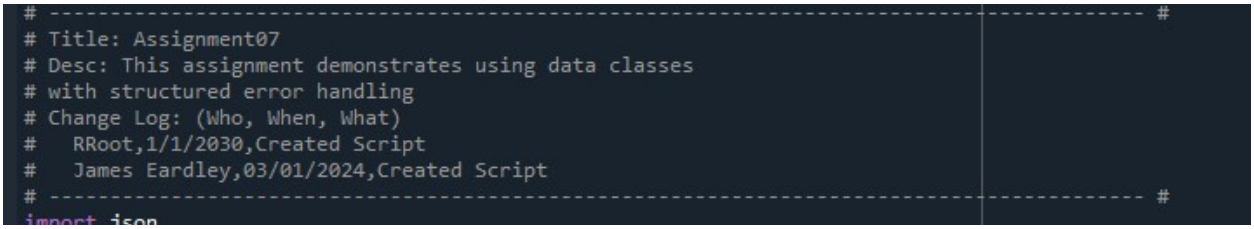
Create a Program

Introduction

In this paper I will detail how I created a Python program that builds on the things learned in the prior modules by introducing properties, creating objects from classes, data encapsulation.

Creating the Program

When starting to write a program in any programming language, including Python, it's important to create a header that provides a title of the program, a description of what the program does, and a change log. This is helpful for allowing other users to more clearly understand the purpose of a script and who authored it.



```
# ------------------------------------------------------------------------------------------ #
# Title: Assignment07
# Desc: This assignment demonstrates using data classes
# with structured error handling
# Change Log: (Who, When, What)
#   RRoot,1/1/2030,Created Script
#   James Eardley,03/01/2024,Created Script
# ------------------------------------------------------------------------------------------ #
import json
```

Figure 1.1: Header displaying the title of the assignment, description, and change log.

Using classes to define objects allow us to create a template for building objects. This enables data encapsulation, which is a fundamental part of object oriented programming and is important for establishing isolation (each object is distinct despite coming from the same class), reusable (classes as templates, and abstraction (you can interact with objects based on their attributes).

__Init__ is a special method called a constructor that initializes the object's attributes – it assigns initial values to the objects member variables.

The self keyword is used to refer to data or functions found in an object instance and not directly in the class.

```python
class Person:
    """
    A collection of presentation layer functions that manage user input and output

    ChangeLog: (Who, When, What)
    James Eardley,03.01.2024,Created Class

    """

    def __init__(self, first_name: str = '', last_name: str = ''):
        self.first_name = first_name
        self.last_name = last_name
```

Figure 1.2: The above introduces use of classes for creating objects, constructors, and the self keyword.

I've included some data validation using .isalpha() to ensure that the first name is not entered as a number or empty, which would likely be incorrect.

The underscores preceding first_name make sure the attribute is private, which ensures that the attribute cannot be changed by code outside of the class.

@property decorator indicates the function as a getter or accessor of attribute data. You then reference the function again and add .setter do create the setter function.

```python
#  Create property getter and setter for first name using the same code as in the Student class
@property  # (Use this decorator for the getter or accessor)
def first_name(self):
    return self.__first_name.title()  # formatting code

@first_name.setter
def first_name(self, value: str):
    if value.isalpha() or value == "":  # is character or empty string
        self.__first_name = value
    else:
        raise ValueError("The last name should not contain numbers.")
```

Figure 1.3: The above introduces use properties, data validation, and private attributes

Super introduces the idea of inheritance which allows code to be inherited from another sources, such as a parent class or codebase.

```python
class Student(Person):
    """
    A class representing student data.

    Properties:
    - first_name (str): The student's first name.
    - last_name (str): The student's last name.
    - course_name (str): The course name.

    ChangeLog:
      James Eardley,03.01.2024,Created Class
    """

    def __init__(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
```

Figure 1.4: Super introduces the idea of inheritance.

Testing the Program

When done correctly, the script runs, prompting the user with a menu of items to choose from. Choosing 1 allows the user to register a student with a class. This can be run over and over to allow for the registration of multiple students. Action 2 displays the current data that has been input. Action 3 sabes the data to a csv. Finally, action 4 breaks the loop and exits the menu. Action steps 1 and 3 now utilized error handling.

```
Enter your menu choice number: 1

Enter the student's first name: James

Enter the student's last name: Eardley

Please enter the name of the course: Python 100

You have registered James Eardley for Python 100.


---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
----------------------------------------



Enter your menu choice number: 2
--------------------------------------------------
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student James Eardley is enrolled in Python 100
Student Chloe Palmer is enrolled in Python 200
Student James Eardley is enrolled in Python 200
Student James Eardley is enrolled in Python 100
--------------------------------------------------


---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
----------------------------------------



Enter your menu choice number: 3


---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
----------------------------------------



Enter your menu choice number: 5
Please, choose only 1, 2, 3, or 4

Please only choose option 1, 2, or 3


---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
----------------------------------------



Enter your menu choice number: 4
Program Ended
```

Figure 1.6: The output of the script as displayed in the Console of the IDE. I tested the error handling by including a number in the first name.

The script was also run in cmd prompt to ensure that the program could be run in multiple ways.

Summary

Assignment 07 built on the prior assignments by including the use of classes for creating objects, parameters, and constructors – key aspects of object oriented programming.

Repository Link:
https://github.com/jmeardley/introtoprog-python-mod07