Zach Harvey
11/26/2023
IT FDN 110 A
Assignment 7
https://github.com/harv-uw/IntroToProg-Python-Mod07

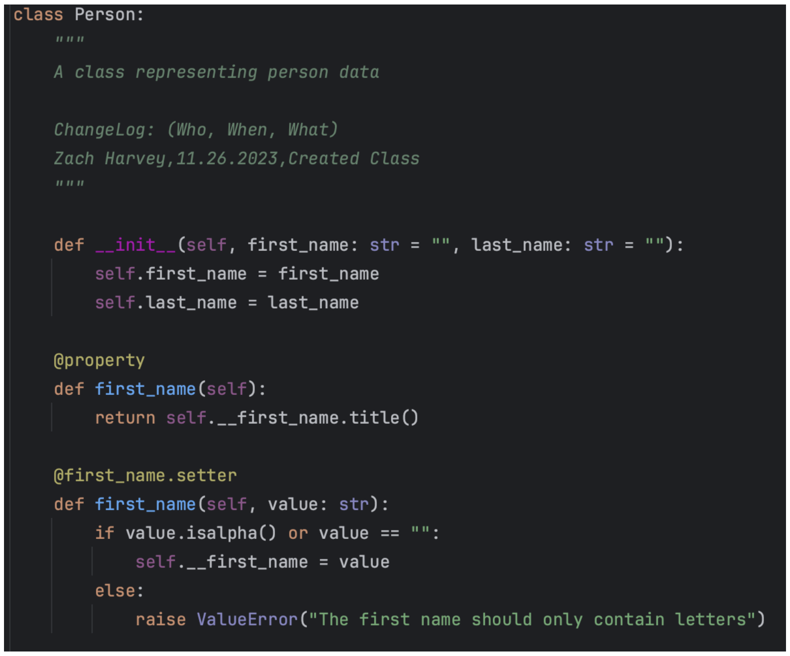
# Assignment 7 - Classes, Methods, and Inheritance

## Introduction

In this module, we expanded our usage of classes to contain constructors. This allowed us to abstract any validation and object construction away from the actual application code, providing better organization of the program. Additionally, we implemented an inherited class and overrode some class methods for better usability in our script.

## Constructors

In this assignment, we defined two more classes, Person and Student. The Person class was defined using a constructor and decorators to annotate the getter and setter and avoid using manual getters and setters. The constructor method, "__init__" is a "magic method" in Python meaning it will be invoked as soon as the class is instantiated. If no constructor method is present, Python will by default create one but it may result in improper type casting of your attributes, thus it is best to manually create one. When the constructor runs, it uses the setter method to set the passed value on the class object. This can be seen in the image below.



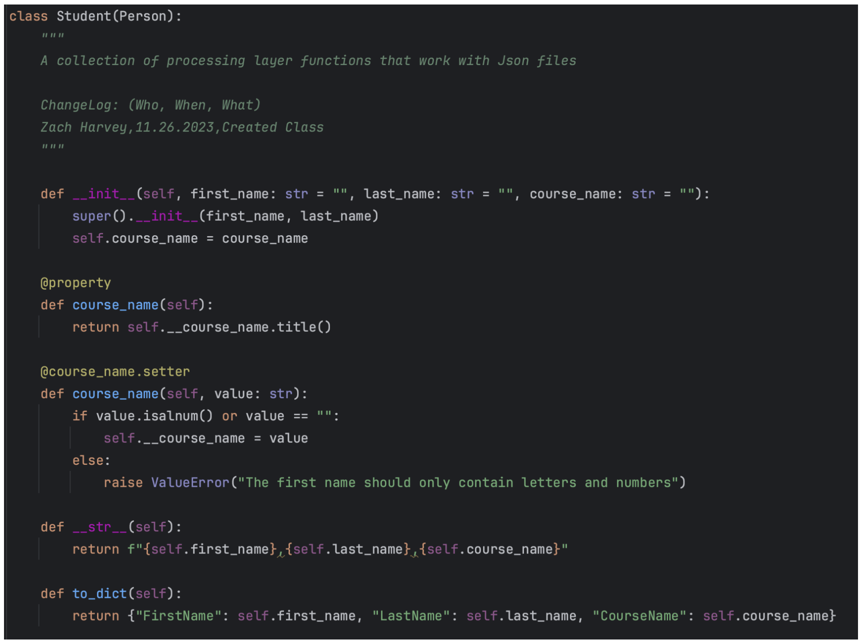
```
class Person:
    """
    A class representing person data

    ChangeLog: (Who, When, What)
    Zach Harvey,11.26.2023,Created Class
    """

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

    @property
    def first_name(self):
        return self.__first_name.title()

    @first_name.setter
    def first_name(self, value: str):
        if value.isalpha() or value == "":
            self.__first_name = value
        else:
            raise ValueError("The first name should only contain letters")
```

*Image 1: Definition of the Person class with constructor and attribute getter/setter*

The “@property” decorator above is what defines the “first_name” attribute on the class. Then, the “@first_name.setter” is defined to actually set the value on the object. Also to note, there is no need to set the attributes initially above the constructor because Python will automatically interpret them when defined in the construct

## Inheritance

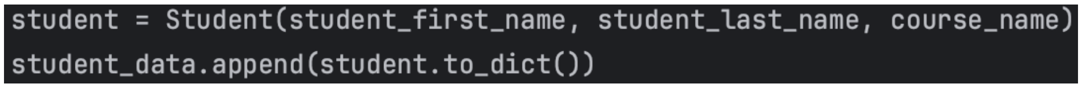
Inheritance is the practice of stemming one class from another. That means that the “child” or “sub” class inherits all the methods from its parent class. This is another means of organization and encapsulation within a Python program. The image below catpures the “Student” class inheriting the “Person” class. Logically this makes sense since a student is a person with more properties, in this case, a “couse_name”.



```python
class Student(Person):
    """
    A collection of processing layer functions that work with Json files

    ChangeLog: (Who, When, What)
    Zach Harvey,11.26.2023,Created Class
    """

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

    @property
    def course_name(self):
        return self.__course_name.title()

    @course_name.setter
    def course_name(self, value: str):
        if value.isalnum() or value == "":
            self.__course_name = value
        else:
            raise ValueError("The first name should only contain letters and numbers")

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

    def to_dict(self):
        return {"FirstName": self.first_name, "LastName": self.last_name, "CourseName": self.course_name}
```

*Image 2: Definition of the Student class inherited from the Person class*

As you can see, this class does not need to redefine the getters and setters for the “first_name” and “last_name” attributes. When the constructor is run, it only needs to call the constructor of its parent class, Person, and pass it the values to set. The parent class is called using “super()”. However, because the parent class does not have a “course_name” attribute, this class still needs to define the getter and setting for it. Finally, the Student class can be used and methods can be called like in the example below.



```python
student = Student(student_first_name, student_last_name, course_name)
student_data.append(student.to_dict())
```

*Image 3: Using the Student class and one of its methods in the script*

## Summary

Defining classes can be a bit cumbersome with Python's syntax but it provides a lot of flexibility for the programmer to define what they need in the program.  Constructors and the idea of inheritance will be essential in making large scripts where it is necessary to encapsulate certain logic outside of the business logic of the script.