Ericka Moreno

Nov 26, 2024

IT FDN 110 A

Assignment_07

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

Classes and Objects

Introduction

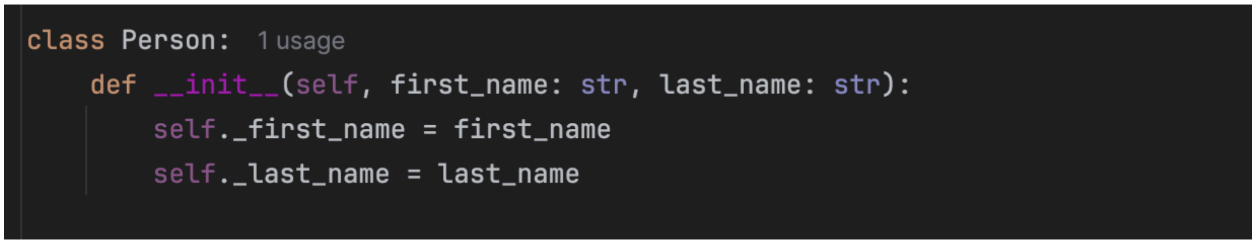
In this module we learned about classes and objects and how we can use them to organize code and increase efficiency. Classes can be a parent or child, and children classes inherit from parent classes. We use a constructor for classes and can overload these. We used python magic methods within these constructors. This module further developed on topics we have learned throughout this course to make our script function more complex.

Classes

In this assignment we create a class Person and a class Student. We then create objects from these classes.

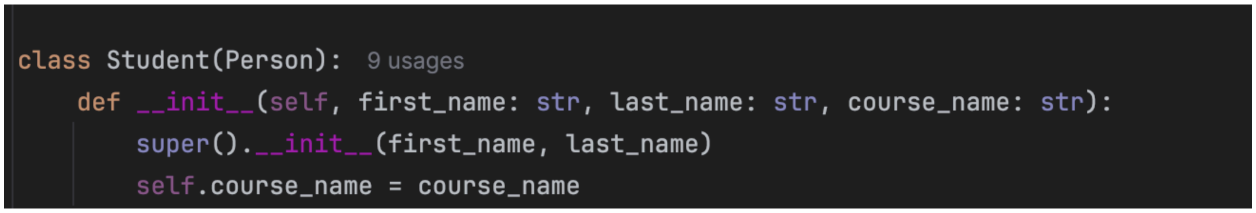
Objects

Objects are instances of classes. We have two classes in this script: Person and Student. With these classes we create instances of Person and Student and these instances are called objects. The Person is the parent class and the Student is the child class. This means that Student extends Person.



```
class Person:  1 usage
    def __init__(self, first_name: str, last_name: str):
        self._first_name = first_name
        self._last_name = last_name
```

Figure 1 Person class (parent)



```
class Student(Person):  9 usages
    def __init__(self, first_name: str, last_name: str, course_name: str):
        super().__init__(first_name, last_name)
        self.course_name = course_name
```

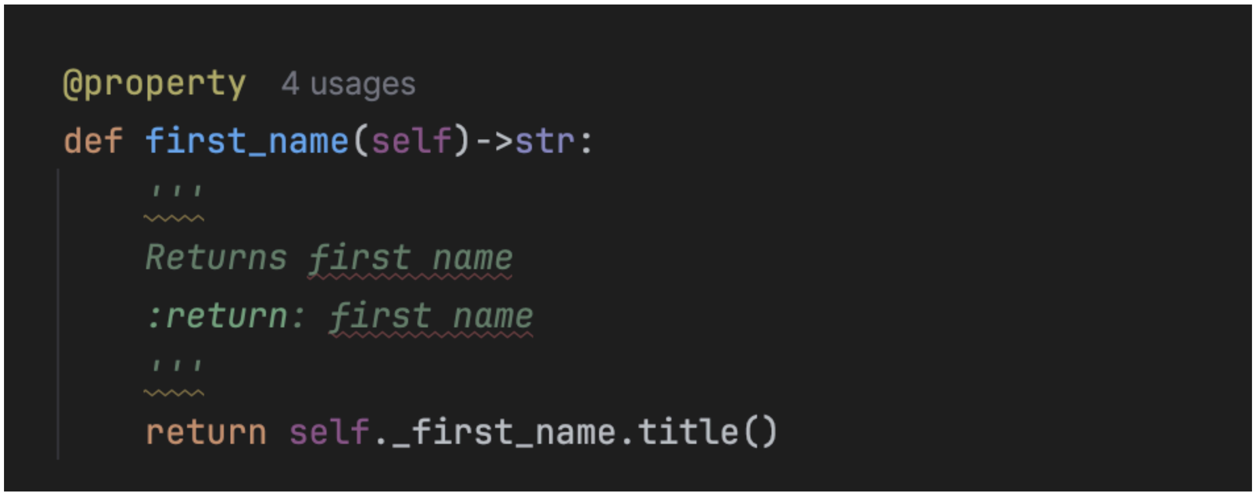
Figure 2 Student class (child)

Inheritance

Child class extends Parent class, in this assignment, the Student class extends Person class. The Student class inherits from Person and then also takes in the course name.

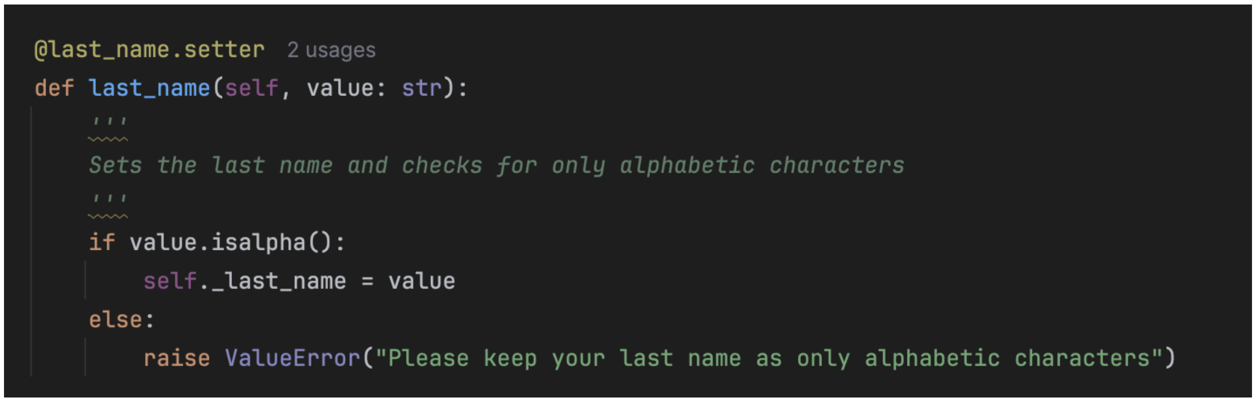
Properties

Properties are used in this script to access and modify attribute data. Within attributes we get data and set data. These are what access and modify the data. The property decorator indicates a function as a "getter". The setter property in figure 4 allows us to add validation and error handling.



```
@property  4 usages
def first_name(self)->str:
    '''
    Returns first name
    :return: first name
    '''
    return self._first_name.title()
```

Figure 3 Property



```
@last_name.setter  2 usages
def last_name(self, value: str):
    '''
    Sets the last name and checks for only alphabetic characters
    '''
    if value.isalpha():
        self._last_name = value
    else:
        raise ValueError("Please keep your last name as only alphabetic characters")
```

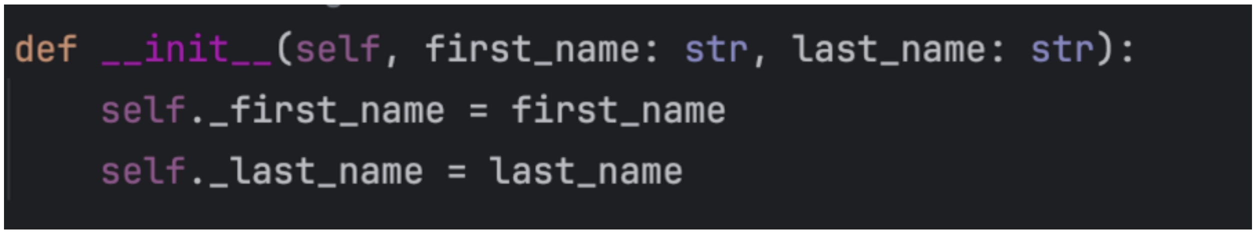
Figure 4 Setter

Constructor

The constructor is a function that is called when an object of a class is created. It set's the object's attributes when the object is created.

Self keyword

Self in the constructor is used to refer to data in an object instance, these are used in the script in the constructor for Person and Student. This is important so that we are referring to the correct object instance in memory.

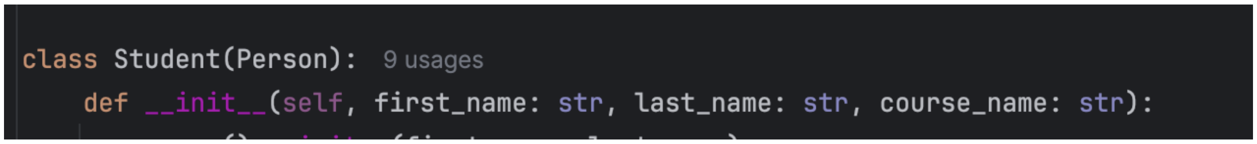


```
def __init__(self, first_name: str, last_name: str):
    self._first_name = first_name
    self._last_name = last_name
```

Figure 4

Overloading

Overloading a constructor is taking the same function definition but adding in more parameters. This is used in the Student class to extend on Person. This means that first_name and last_name are used as in Person, but then course_name is also added which is overloading the constructor.



```
class Student(Person):  9 usages
    def __init__(self, first_name: str, last_name: str, course_name: str):
```

Figure 5

Summary

In this module we learned about Classes and Objects as part of object oriented programming. Specifically, we applied this information to our script using the Person and Student class. The Person class demonstrates a parent class and the Student object demonstrates the child class. The Student extends from Parent. We used constructors to initialize data and overloaded the constructor for Student to add in the same information as Person and also add in course_name. With classes we can use them to create instances of the class which is an object.