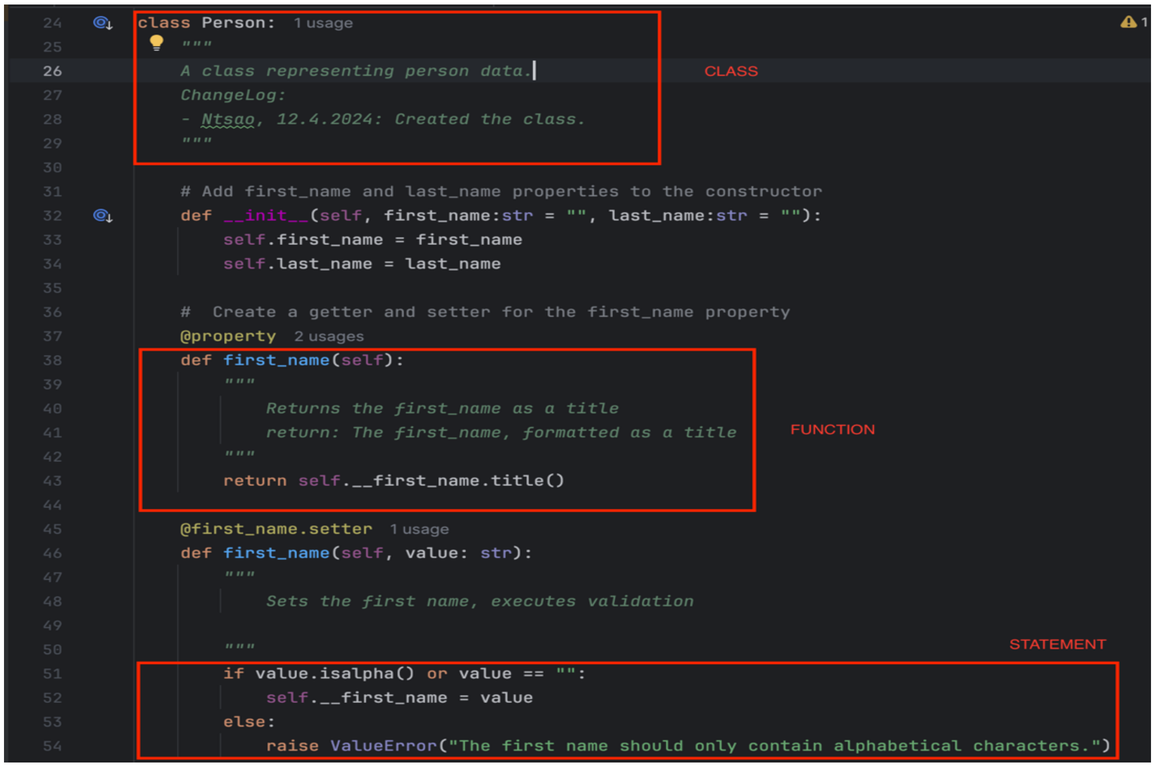
Nicole Tsao
December 5, 2024
IT FDN 100A
Assignment 07

# Classes, Functions, and Beyond

## Introduction:

In module 07 we continue to dive deeper into the roles of functions and classes and how they maintain our code. We tackle new terminologies and concepts that are essential to understanding object-oriented programming. I'll walk through how Assignment 07 contributed to my understanding of the key concepts.

1.) Statements, Functions, Classes – We have been working with **statements** since the beginning of the class. Each statement performs a specific action or task, such as assigning a value to a variable, controlling the flow of execution, or defining a function. Statements are often found wrapped in **functions** when they are meant to be called or used multiple times. While Classes house the data for the object and the functions that operate on the data.



```
class Person:
    """
    A class representing person data.
    ChangeLog:
    - Ntsao, 12.4.2024: Created the class.
    """

    # Add first_name and last_name properties to the constructor
    def __init__(self, first_name:str = "", last_name:str = ""):
        self.first_name = first_name
        self.last_name = last_name

    #  Create a getter and setter for the first_name property
    @property
    def first_name(self):
        """
            Returns the first_name as a title
            return: The first_name, formatted as a title
        """
        return self.__first_name.title()

    @first_name.setter
    def first_name(self, value: str):
        """
            Sets the first name, executes validation

        """
        if value.isalpha() or value == "":
            self.__first_name = value
        else:
            raise ValueError("The first name should only contain alphabetical characters.")
```

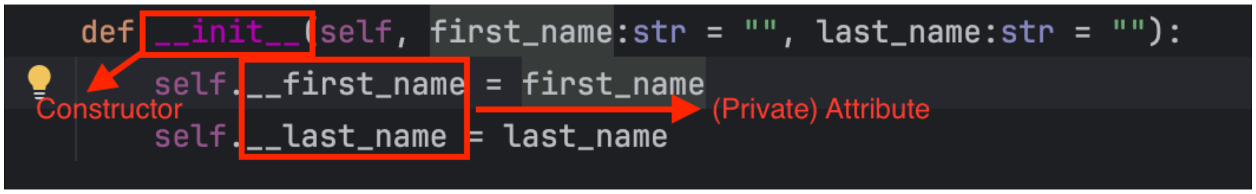
Figure 1 – In this assignment, *class Person* is used as the **parent class or the superclass**

2.) Types of Classes:

a. Presentation Class - It manages the display of information and often translates user actions into commands that are sent to the Processing Class or Data Class.

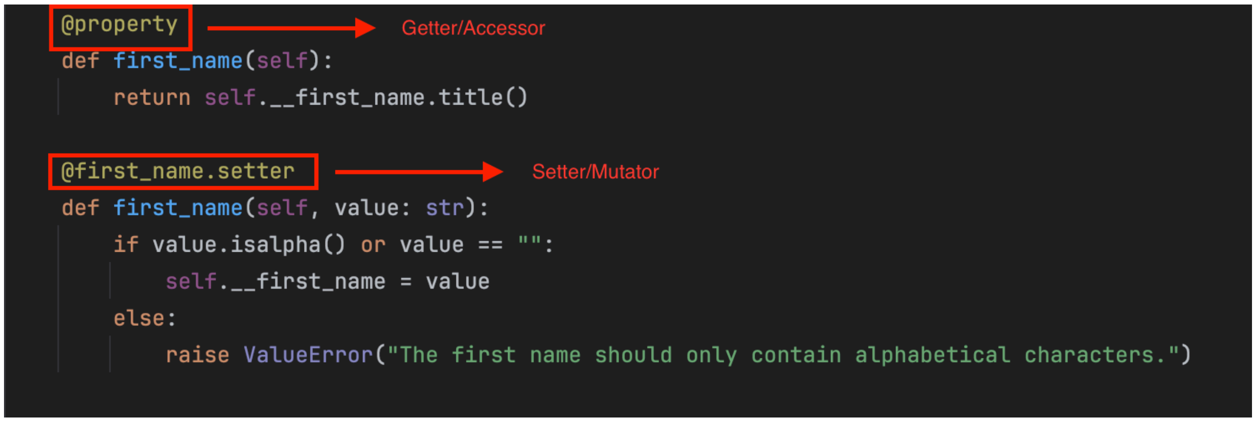
b. Processing Class - It is responsible for manipulating data received from the Presentation Class and working with it, often by calling methods in the Data Class.

**c.** Data Class - Primarily used to store and manage data. This includes **attributes** (basically the **variables** that store the data inside the class), **constructors** (special method used to create and set up new objects) and **properties** (functions designed to manage attribute data).



```python
def __init__(self, first_name:str = "", last_name:str = ""):
    self.__first_name = first_name
    self.__last_name = last_name
```

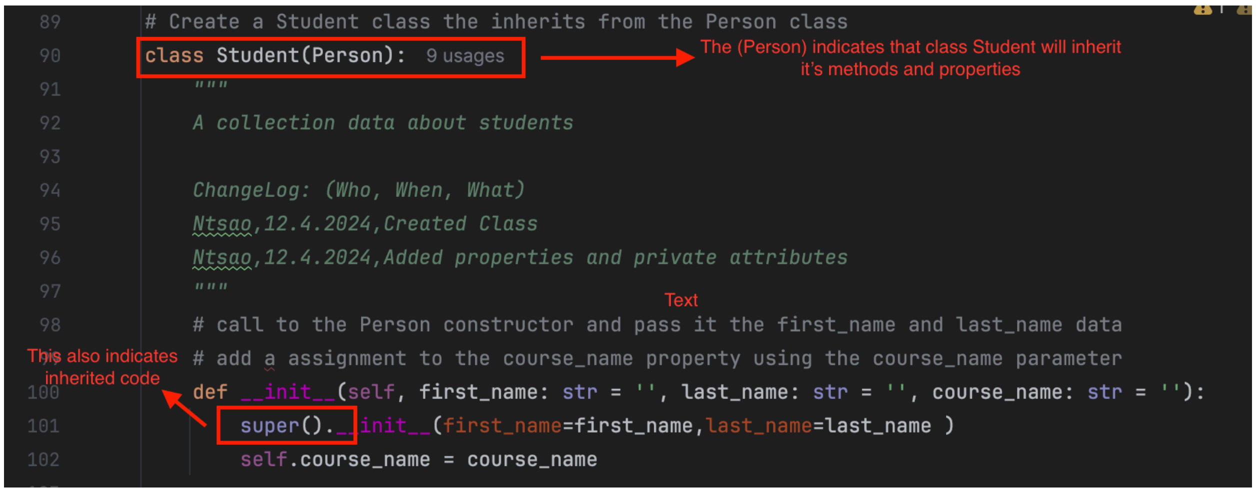
NOTE: Private attributes are used to prevent the code from being altered when used outside of the class.



```python
@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 alphabetical characters.")
```

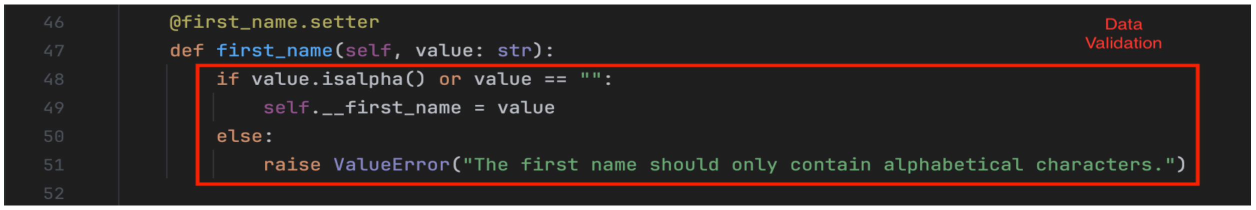
3.) Child class/subclass - can inherit data and behaviors (properties and methods) from an existing class (parent class), this is termed as “inherited code”. I like to think of the parent class/superclass as the base code, then subclass is to add on other properties or methods on top of that code. This was you do not have to be redundant and enter the exact same script all over again.


The (Person) indicates that class Student will inherit it's methods and properties
Text
This also indicates inherited code

```
# Create a Student class the inherits from the Person class
class Student(Person):  9 usages
    """
    A collection data about students

    ChangeLog: (Who, When, What)
    Ntsao,12.4.2024,Created Class
    Ntsao,12.4.2024,Added properties and private attributes
    """
    # call to the Person constructor and pass it the first_name and last_name data
    # add a assignment to the course_name property using the course_name parameter
    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
```

4.) Data Validation – Similar to error handling, data validation in Python is just making sure that the values you give to an object's attributes are correct and follow the rules you set.



```
    @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 alphabetical characters.")
```

## Summary:

We have uncovered even more uses of classes, functions and statements and how to use them to our advantage. It has been interesting to learn and practice how to convert a script using one method to another. Once you get the hang of these, you can write code that's cleaner, faster, and way easier to understand. It's like learning how to use the right tools to build something awesome!