

IB

May 28, 2024

IT FDN 110 A

Assignment 07

<https://github.com/uwib25/IntroToProg-Python-Mod07>

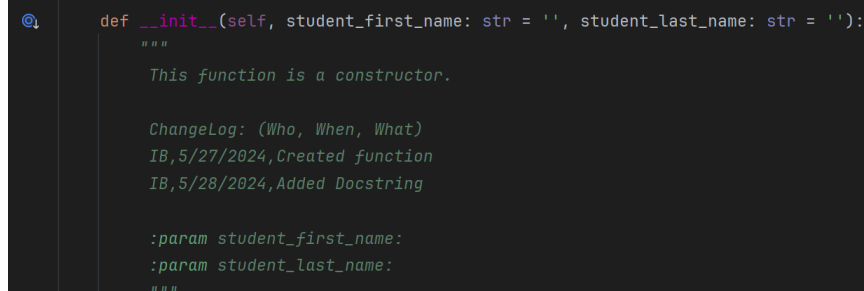
Organizing code through functions

Introduction

In the seventh module, we learned the difference between attributes and private attributes, constructors, the self keyword, and properties. This week's lesson provided a lot more context into program structuring and provided a lot more insight into the differences between global variables and attributes. Additionally, we learned the importance of inheritance to access attributes and properties from a parent class.

Constructor

This constructor method takes two parameters: the student_first_name & student_last_name. This is also the first time we have started using the self keyword as well, which we learned is a way to refer to an object's instance within a class.



```
def __init__(self, student_first_name: str = '', student_last_name: str = ''):
    """
    This function is a constructor.

    ChangeLog: (Who, When, What)
    IB,5/27/2024,Created function
    IB,5/28/2024,Added Docstring

    :param student_first_name:
    :param student_last_name:
    """
```

Figure 1: `def __init__(self,)`

Properties

I really enjoyed learning about the property function (getter & setter) this week. This is a pretty powerful function to allow us to add data validation and error handling to private attributes. I think it's pretty simple to learn and easy to understand, but adds a lot of functionality for ensuring the data is formatted correctly. This can be especially useful when used as a Super class since the validation is used whenever the attribute is used.

```
@property
def student_last_name(self):
    """
    This function is for "getting" student_last_name.

    ChangeLog: (Who, When, What)
    IB,5/27/2024, Created function
    IB,5/28/2024, Added Docstring

    :return:
    """
    return self.__student_last_name.title()
```

Figure 2: Property (Getter) function

```
@student_last_name.setter
def student_last_name(self, value: str):
    """
    This function is for "setting" student_first_name to add validation
    and error handling.

    ChangeLog: (Who, When, What)
    IB,5/27/2024, Created function
    IB,5/28/2024, Added Docstring

    :param value:
    :return:
    """
    if value.isalpha() or value == "":
        self.__student_last_name = value
    else:
        raise ValueError("The last name should only contain alphabetic characters")
```

Figure 3: Setter function

Class Inheritance

One of the main topics this week is the ability for inheriting the properties and methods from another class. In this week's acceptance criteria, we needed to create a Person class and a Student class, and

have the Student class inherit the attributes from the Person class (student_first_name & student_last_name). This allowed us to define these attributes in the Person class, and then use these attributes within the Student class without the need to recreate these attributes.

```
class Student(Person):  
    """  
    A collection of data about students.
```

Figure 4: Class Inheritance

Summary

In summary, there was a lot of content packed into this week that are more advanced concepts. I have a much better understanding why classes are so powerful now, and how using classes with inheritance is so useful. This allows us to write code once, and then use that code in other classes instead of duplicating efforts that are unneeded. Lastly, I added a lot more docstrings to my functions and classes this to make the code more transparent and easier to understand the intent.