Day 7 Python Challenge

It's time to program with object-oriented programming principles in mind. I'm going to ask you to create a code that allows a person to perform operations on their bank account.

Don't be scared, the instructions will be well defined so that you can do it quickly:

1. First, you are going to create a class called **Person**, and the person is going to have only two attributes: first name and last name. That's it.
2. Then, you are going to create a second class called **Customer**. This class will inherit from Person because customers are persons, but it will also have its own attributes such as **account number** and **balance**. *For example, the bank account balance.*
3. But that's not all: the Customer will also have three methods:
    1. The first one is going to be one of the **special methods**, and it is going to be the one that allows us to **print** our client. When the code asks to print the client, this method will allow their data to be displayed, including their account balance.
    2. Then, a method called **deposit**, which allows the user to decide how much money they want to add to their account.
    3. And finally, a method called **withdraw**, which allows the user to take money and deduct it from their account.
4. Once you have created these two classes, you have to create the code for your program to run, asking the user to choose whether they want to make a deposit or withdrawal. The user can make as many transactions as they want until they decide to exit the program. Therefore, our code has to keep track of how much money is in the balance, and you must make sure that the customer **never** withdraws more money than they have.

Remember, now that you know how to create stable classes and objects that retain information, you don't need to create functions that return the balance, since the client instance can constantly know what its balance is. This is a direct result of its ability to conduct operations by calling directly to that attribute and not to a separate variable.

To make your program work, you can organize your code as you want. There are many ways to do it, but my recommendation is that you basically create two functions:

- One that is in charge of creating the client, asking the user for all the necessary information, and returning, *through a return*, a client object already created.
- Another function that organizes code execution: first, it calls the create_client function and then it is in charge of maintaining the user in a loop that asks them all the time if

they want to make a deposit withdrawal or exit the program, and to show them the balance every time they carry over a modification

For this program not to become super long or complex, I propose that this time we do not look at controls to see if the user inputs allowed options or not, if they have put numbers or not, if they have put upper or lower case...

Let's create the code **trusting** that the user is always going to enter appropriate information. If you prefer to include all those controls, that's fine. We are going to dedicate ourselves to simply creating the hard code so that the explanation doesn't become super long.

So are you ready? We are. Start programming and have a lot of fun and we'll be waiting for you in the next lecture to show you our solution.