**Bank Account**

Objectives

* Practice writing classes

As we continue thinking about our banking application, we realize that it would be more accurate to assign a balance not to the user directly, but that in the real world, users have *accounts*, and *accounts* have balances. This gives us the idea that maybe an account *is its own class*! But as we stated, it is not completely independent of a class; accounts only exist because users open them.

The BankAccount class should have a balance. When a new BankAccount instance is created, if an amount is given, the balance of the account should initially be set to that amount; otherwise, the balance should start at $0. The account should also have an interest rate, saved as a decimal (i.e. 1% would be saved as 0.01), which should be provided upon instantiation. (Hint: when using default values in parameters, the order of parameters matters!)

The class should also have the following methods:

* **deposit(self, amount)** - increases the account balance by the given amount
* **withdraw(self, amount)** - decreases the account balance by the given amount if there are sufficient funds; if there is not enough money, print a message "Insufficient funds: Charging a $5 fee" and deduct $5
* **display\_account\_info(self)** - print to the console: eg. "Balance: $100"
* **yield\_interest(self)** - increases the account balance by the current balance \* the interest rate (as long as the balance is positive)

This means we need a class that looks something like this:

*class BankAccount:*

*def \_\_init\_\_(self, int\_rate, balance): # don't forget to add some default values for these parameters!*

*# your code here! (remember, this is where we specify the attributes for our class)*

*# don't worry about user info here; we'll involve the User class soon*

*def deposit(self, amount):*

*# your code here*

*def withdraw(self, amount):*

*# your code here*

*def display\_account\_info(self):*

*# your code here*

*def yield\_interest(self):*

*# your code here*

* Top of Form
* Create a BankAccount class with the attributes interest rate and balance
* Add a deposit method to the BankAccount class
* Add a withdraw method to the BankAccount class
* Add a display\_account\_info method to the BankAccount class
* Add a yield\_interest method to the BankAccount class
* Create 2 accounts
* To the first account, make 3 deposits and 1 withdrawal, then yield interest and display the account's info all in one line of code (i.e. chaining)
* To the second account, make 2 deposits and 4 withdrawals, then yield interest and display the account's info all in one line of code (i.e. chaining)

Bottom of Form