## Account

Create a single class called **Account**. Upon initialization, it should receive **owner** (str) and **amount** (int, **optional**, 0 by default). It should also have an attribute called **\_transactions** (**empty list** upon initialization). Create the following **methods**:

* **add\_transaction(amount)** - if the amount is **not an integer**, raise **ValueError** with message **"please use int for amount"**, otherwise, **add the amount** to the transactions
* **balance()** - property that returns **sum** of the **amount** and all the **transactions**
* **validate\_transaction(account: Account, amount\_to\_add)**
  + If the balance becomes **less than zero**, raise **ValueError** with message **"sorry cannot go in debt!"** and **break the transaction**.
  + Otherwise, **complete it** and **return** a message **"New balance: {account\_balance}"**

Implement the correct **magic methods**, so the code in the example bellow works properly

### Examples

|  |  |
| --- | --- |
| **Test Code** | **Output** |
| acc = Account('bob', 10)  acc2 = Account('john')  print(acc)  print(repr(acc))  acc.add\_transaction(20)  acc.add\_transaction(-20)  acc.add\_transaction(30)  print(acc.balance)  print(len(acc))  for transaction in acc:  print(transaction)  print(acc[1])  print(list(reversed(acc)))  acc2.add\_transaction(10)  acc2.add\_transaction(60)  print(acc > acc2)  print(acc >= acc2)  print(acc < acc2)  print(acc <= acc2)  print(acc == acc2)  print(acc != acc2)  acc3 = acc + acc2  print(acc3)  print(acc3.\_transactions) | Account of bob with starting amount: 10  Account(bob, 10)  40  3  20  -20  30  -20  [30, -20, 20]  False  False  True  True  False  True  Account of bob&john with starting amount: 10  [20, -20, 30, 10, 60] |