Classes and Object Oriented Programming

This lab is not graded. Complete it for practice and to get familiar with classes in Python.

Create a class for a playing card

- Create a new Python module (file) called `card.py`.
- Create the class `Card`.
- The constructor for this class takes two arguments:
  - a value for the card (between 1 and 10 included). This can only be a number, or a string that is a number (eg `"9"` or `9`)
  - a color for the card (a string, must be either `red` or `black`)
  - if these conditions are not respected, the class should raise an `AttributeError`
- Create the method `is_stronger_than`. It receives another instance of `Card` as an argument, and returns `True` if the card received as argument has a lower value (regardless of the color)

```
five_black = Card(5, "black")
ten_red = Card(10, "red")
ten_red.is_stronger_than(five_black)    # True (10 is stronger than 5)
```

You can use tests to check your work: `pytest test_card.py`.

Create a class for a countdown timer

- Create a new Python file called `counter.py`.
- Create the class `Countdown`.
- The constructor for this class takes:
  - one required argument, `start` (an integer)
  - one optional argument, `step` (an integer)
- Class instances have an attribute `current` (the current value of the countdown timer)
- This class has one method `down`
  - When called, the value of `current` is decremented by the `step`
- Class instances have an attribute `complete`. This is a boolean attribute
  - it is True if `current` is less than or equal to 0
  - `False` otherwise

You can use tests to check your work: `pytest test_counter.py`.

Create a class for a bank account

- Create a new Python module (file) called `bank.py`.
- Create the class `BankAccount`.

- Make sure that this class has an **instance attribute** `amount`. This amount should be equal to 0 when creating a new instance.
- Create two methods on your class:
    - `deposit`: allows you to deposit money on your account. Takes an argument (the amount you want to deposit).
    - `withdraw`: allows you to withdraw money from your account. Takes an argument (the amount you want to withdraw).

You can use tests to check your work: `pytest test_bank_account.py`.

Go further

- Transform the `amount` attribute into a property.
- Make sure the amount on the account cannot go below 0!

You can use tests to check your work: `pytest test_bank_account2.py`.