




Gestion d'un Compte Bancaire



Le but est de définir une classe **CompteBancaire** qui permette :

- d'instancier des objets tels que **compte1,compte2**, ...
- ces objets "compte" auront deux attributs : **nom** (=titulaire du compte) et **solde** (=argent sur le compte),
- de connaître le nom du titulaire d'un compte, ainsi que la somme présente sur ce compte,
- de gérer les transferts d'argent sur le compte en "sécurité".
- et même de créer une méthode "protégée" pour pirater un compte.

- et même de créer une méthode "protégée" pour pirater un compte.

- 1.  Créez la classe **CompteBancaire**.
- 2.  Créez le constructeur de cette classe en faisant en sorte qu'un nouveau compte s'ouvre par défaut avec 0 euro dessus.
- 3.  Créez les accesseurs à chacun des deux attributs des objets de la classe.

```
class CompteBancaire :  
    def __init__(self,nom,solde=0):  
        self.__nom = nom  
        self.__solde = solde  
    def get_nom(self):  
        return self.__nom  
  
    def get_solde(self):  
        return self.__solde
```

Objects

CompteBancaire class

<code>__CompteBancaire__set_variation</code>	function <code>__set_variation(self, valeur)</code>		
<code>__init__</code>	function <code>__init__(self, nom, solde)</code> default arguments: <table><tr><td><code>solde</code></td><td><code>0</code></td></tr></table>	<code>solde</code>	<code>0</code>
<code>solde</code>	<code>0</code>		
<code>get_nom</code>	function <code>get_nom(self)</code>		
<code>get_solde</code>	function <code>get_solde(self)</code>		

- Créez un mutateur privé qui permet de faire évoluer la somme placée sur un compte de **variation** euros.
- Rajoutez à ce mutateur un test qui permet d'afficher un message si le compte est à découvert à l'issue de la modification.
- Créez une méthode **depot** qui utilise le mutateur privé précédent pour ajouter une certaine somme sur un compte bancaire.
- Créez une méthode **retrait** qui utilise le mutateur privé précédent pour retirer une certaine somme sur un compte bancaire.
- Testez cette méthode **retrait** de sorte qu'un compte passe dans le négatif. Est-ce que l'affichage prévu dans le mutateur privé gérant l'attribut **solde** s'affiche lors de l'utilisation de cette méthode publique ?

<code>_CompteBancaire__set_variation</code>	function <code>__set_variation(self, valeur)</code>
<code>depot</code>	function <code>depot(self, valeur)</code>
<code>retrait</code>	function <code>retrait(self, valeur)</code>

```
def __set_variation(self,valeur):
    self.__solde = valeur
    if self.__solde <= 0:
        print("Votre compte est à découvert d'une somme équivalente à " + str(self.__solde))

def depot(self,valeur):
    self.__set_variation(self.__solde + valeur)
def retrait(self,valeur):
    self.__set_variation(self.__solde - valeur)
```

```
compteA = CompteBancaire('Ilyan',0)
```

```
compteA.depot(10)
print(compteA.get_solde())
compteA.retrait(10)
print(compteA.get_solde())
print(compteA.afficher())
```

```
10
Votre compte est à découvert d'une somme équivalente à 0
0
```

- Créez une méthode `afficher` qui affiche le nom du titulaire et le solde de son compte.
- Créez un mutateur privé qui permette de changer le nom du titulaire.
- Créez une méthode protégée `_pirater` qui utilise le mutateur précédent et qui permet de changer le nom du titulaire d'un compte bancaire.
- Utilisez cette méthode `_pirater` pour vous attribuer un compte bancaire. Oh ! Ce n'est pas bien du tout !

```
def afficher(self):
    return self.__nom, self.__solde

def set_nom(self,valeur):
    self.__nom = valeur

def _pirater(self,item):
    self.__nom = item
```

<code>_pirater</code>	function <code>_pirater(self, item)</code>
<code>afficher</code>	function <code>afficher(self)</code>
<code>set_nom</code>	function <code>set_nom(self, valeur)</code>

```
compteA = CompteBancaire('Ilyan',17)
compteB = CompteBancaire('Alex',20)
compteA.depot(10)
print(compteA.get_solde())
compteA.retrait(10)
print(compteA.get_solde())
print(compteA.afficher())

pirate_compteB = compteB._pirater('Connor')
```

CompteBancaire instance

<code>__CompteBancaire__nom</code>	"Alex"
<code>__CompteBancaire__solde</code>	20

CompteBancaire instance

<code>__CompteBancaire__nom</code>	"Connor"
<code>__CompteBancaire__solde</code>	20

```

class CompteBancaire :
    def __init__(self,nom,solde=0):
        self.__nom = nom
        self.__solde = solde
    def get_nom(self):
        return self.__nom

    def get_solde(self):
        return self.__solde

    def __set_variation(self,valeur):
        self.__solde = valeur
        if self.__solde <= 0:
            print("Votre compte est à découvert d'une somme équivalente à " + str(self.__solde))

    def depot(self,valeur):
        self.__set_variation(self.__solde + valeur)
    def retrait(self,valeur):
        self.__set_variation(self.__solde - valeur)

    def afficher(self):
        return self.__nom, self.__solde

    def set_nom(self,valeur):
        self.__nom = valeur

    def _pirater(self,item):
        self.__nom = item

compteA = CompteBancaire('Ilyan',17)
compteB = CompteBancaire('Alex',20)
compteA.depot(10)
print(compteA.get_solde())
compteA.retrait(10)
print(compteA.get_solde())
print(compteA.afficher())

pirate_compteB = compteB._pirater('Connor')

```

