EXERCICE 5

Cet exercice est un exercice d'algorithmique et de programmation en langage Python. Il aborde la programmation orientée objet.

Des élus d'un canton français décident de mettre en place une monnaie locale complémentaire dans leur circonscription, appelée le « sou ». L'objectif est d'encourager la population à acheter auprès de vendeurs et producteurs locaux.

La plus petite valeur du sou est le billet de 1 sou, il ne peut donc pas être séparé en dessous de ce montant. Le sou a un taux de change à parité avec l'euro (1 sou = 1 euro), de façon à ce que le prix d'un article puisse être intégralement payé en sous. Si le prix d'un article n'est pas entier, la différence peut être payée en euros. Par exemple, un article coûtant 3,50 € peut être payé avec 3 sous et 50 centimes d'euros.

Le canton a créé une association chargée de gérer les comptes de ses futurs adhérents au moyen d'une application en langage Python. On a commencé à créer une classe `Compte` dont les propriétés sont : le numéro de compte, le nom de l'adhérent, son adresse et le solde du compte. Lors de la création d'un compte, il faudra renseigner toutes les propriétés sauf le solde qui sera mis à 0 sou. Les méthodes de cette classe sont les suivantes :

Méthodes de la classe `Compte`	Description
`__init__(self, numero, adherent, adresse)`	Crée un nouveau compte et met le solde à 0.
`crediter(self, montant)`	Ajoute montant au solde.
`debiter(self, montant)`	Enlève montant au solde.
`est_positif(self)`	Renvoie Vrai si le solde du compte est positif ou nul.

Partie 1 : Création de la classe `compte`

On a commencé à écrire la classe `Compte`.

```
class Compte:
    def __init__(self, numero, adherent, adresse):
        self.numero = numero
        self.adherent = adherent
        self.adresse = adresse
        self.solde = 0
```

1) **Écrire** sur la copie la méthode `crediter`.
2) **Écrire** sur la copie la méthode `debiter`.
3) **Écrire** sur la copie la méthode `est_positif`.

Partie 2 : Utilisation de la classe `compte`

Monsieur Martin souhaite rejoindre la communauté des utilisateurs du sou et déposer 200 € sur son compte en sou.

1) **Écrire** la ligne de code en langage Python permettant de créer le compte `cpt_0123456A` dont le numéro est "`0123456A`", le titulaire est "`MARTIN Dominique`" qui habite à l'adresse "`12 rue des sports`".

2) **Écrire** la ligne de code en langage Python permettant de déposer 200 € sur le compte de monsieur Martin.

3) Monsieur Martin souhaite transférer 50 sous de son compte « `cpt_0123456A` » vers un autre compte « `cpt_2468794E` ». On a besoin pour cela d'ajouter une méthode à la classe `Compte`.

   Écrire la méthode `transferer`(self, `autre_compte`, `montant`) qui transfère le montant passé en paramètre vers un autre compte. On suppose que le compte courant est suffisamment approvisionné. On utilisera les autres méthodes de la classe `Compte` sans en modifier directement les attributs.

Partie 3 : Gestion des comptes

Pour en faciliter la gestion, on crée une liste Python `liste_comptes` contenant tous les comptes des adhérents. Cette liste sera donc composée d'objets de type `compte`.

Le gestionnaire de l'association souhaite relancer les adhérents dont le compte est débiteur, c'est-à-dire dont le solde est négatif. Il faut, pour cela, ajouter une autre fonction au programme en langage Python.

1) **Écrire** la fonction `rechercher_debiteurs` qui prend en argument `liste_comptes` et renvoie une liste de dictionnaires dont la première clé est le nom de l'adhérent et la seconde clé le solde de son compte en négatif.

   Exemple de résultat :
   `liste_debiteurs = [{'Nom': 'DUPONT Thomas', 'solde': -60}, {'Nom': 'CARNEIRO Sarah', 'solde': -75}]`.

2) **Écrire** la fonction `urgent_debiteur` qui prend en argument la liste de dictionnaires et renvoie le nom de l'adhérent le plus endetté. On suppose qu'aucun adhérent n'a la même dette.