Exercice : POO

Exercice 1

un site de vente en ligne propose des articles de bricolage à la vente. Pour chaque article, celui-ci a :

— une référence sous forme de chaine alpha-numérique ;

— une désignation sous forme de chaine de caractères ;

— un prix sous forme de nombre

On donne ci-dessous un exemple d'articles que l'on peut trouver sur le site :

ref	désignation	prix
P21	perceuse	99,50 €
M07	meuleuse	45 €
V33	visseuse sans fil	69,90 €

1) Écrire la classe Article et son constructeur permettant de créer des objets pour les outils de bricolage.

2) Créer les trois objets du site présents dans le tableau ci-dessus.

3) Le prix d'un article peut varier notamment lorsqu'il est soldé.
Écrire une méthode **solder** qui permet de modifier ce prix. Cette méthode prend en argument le pourcentage de réduction sous forme de nombre entier à appliquer. Elle renvoie le prix modifié de l'article.

Exercice 2

On définit l'objet Compte_bancaire. On donne l'interface de notre objet :

— Attributs : solde et titulaire.

— Méthodes : est_positif, crediter, debiter, virer_vers, afficher ou __repr__

1) Écrire son constructeur.

2) Écrire les différentes méthodes.

3) Tester en créant un compte bancaire **A** avec un solde de 1000 euros et un compte bancaire **B** avec un solde bancaire de 200 euros.

4) Effectuer un virement de 300 de A vers B.

5) Interdire tout virement si le solde du compte est négatif.

Exercice 3

Les fractions sont des nombres dits rationnels de la forme a/b ou $\frac{a}{b}$. Les nombres a et b sont des nombres entiers et b est strictement positif.

On va définir une classe Fraction dans laquelle nous retrouverons différentes méthodes pour les calculs. La classe Fraction possède deux attributs : **num** et **denom**.

1) Écrire le constructeur de cette classe. On prendra en compte le dénominateur strictement positif.

2) Ajouter une méthode __str__ qui renvoie une chaine de caractères de la forme "a/b", ou simplement "a" lorsque le dénominateur vaut 1.

3) Ajouter des méthodes __eq__ et __lt__ qui reçoivent une deuxième fraction en argument et renvoient True si la première fraction représente un nombre égal ou un nombre strictement inférieur à la deuxième fraction.

4) Ajouter des méthodes __add__ et __mul__ qui reçoivent une deuxième fraction en argument et renvoient respectivement la somme et le produit des deux fractions.

5) Tester toutes ces opérations

En supplément :

1) Ajouter des méthodes pour calculer la différence et le quotient de deux fractions. Ces méthodes existent en global.

2) S'assurer que les fractions sont toujours irréductibles.

Exercice 4

On va définir une classe Date pour représenter une date avec trois attributs jour, mois et annee.

1) Écrire son constructeur avec les paramètres j, m et a.

2) On peut créer une variable de classe qui sera utilisée dans la classe par différentes méthodes. L'appel de cette méthode se fera par **nom de la classe.variable**.

Créer une variable de classe mois de type liste contenant les douze mois de l'année. Cette variable sera accessible avec l'appel Date.mois

3) Ajouter une méthode __str__ qui renvoie une chaine de caractères de la forme "11 novembre 1918". Tester l'affichage avec la commande **print**.

4) Ajouter une méthode __lt__ qui permet de déterminer si une date d1 est antérieure à une date d2 en écrivant d1 < d2. La tester.

5) **a)** Modifier le constructeur avec des valeurs par défaut initialisées au 1 janvier 2000.

b) Créer un objet Date, nommé **ddn**, sans paramètres. Vérifier que les attributs de **ddn** ont pour valeurs la date du 1 janvier 2000.

c) Modifier les attributs **ddn** avec les dates de votre anniversaire.

6) Dans la question précédente, vous avez remarqué qu'il est possible de modifier la valeur des attributs d'un objet. Cela peut poser des problèmes surtout lorsqu'on a des attributs dont les valeurs ne doivent pas être accessibles.

Il est possible d'interdire l'accès aux attributs en les cachant. Il suffit d'ajouter un double souligné devant le nom de l'attribut après self.

a) Modifier dans le constructeur les attributs jour, mois et annee pour qu'ils soient cachés.
Penser aussi à modifier les méthodes qui les utilisent.

b) Vérifier qu'il n'est plus possible de modifier les valeurs d'une date une fois créée.

7) Pour accéder aux attributs cachés et les modifier, on peut créer des méthodes particulières appelées accesseurs et mutateurs.
On définit pour l'attribut caché jour, l'accesseur get_jour et le mutateur set_jour de la manière suivante :

```
class Date:
    mois=['janvier','février','mars','avril','mai','juin','juillet',\
          'août','septembre','octobre','novembre','décembre']

    def __init__(self,j=1,m=1,a=2000):
        assert 0<j<=31,"la valeur jour saisie n'est pas possible"
        assert 0<m<=12,"la valeur mois saisie n'est pas possible"
        self.__jour=j
        self.__mois=m
        self.__annee=a

    """Accesseur get_jour et mutateur set_jour pour l'attribut caché jour:
    Si d est un objet Date, alors:
    d.get_jour renvoie la valeur de l'attribut caché jour;
    d.set_jour=k affecte à l'attribut caché jour la valeur k
    """
    @property
    def get_jour(self):
        return self.__jour

    @get_jour.setter
    def set_jour(self,j):
        self.__jour=j
```

a) Ajouter ces deux méthodes dans la classe Date et vérifier que vous pouvez afficher et modifier le jour d'une date.

b) Ajouter les accesseurs et les mutateurs pour le mois et l'année.

c) Vérifier, après avoir créé une date **ddn** sans paramètres, que vous pouvez la modifier par votre date de naissance.