TP 8 Programmation orientée objet

1 Définition de méthodes et de classes

Exercice 1 : Création d'une méthode

Taper le code suivant dans un script Python et ajouter une méthode distance qui permet de calculer la distance entre deux points.

```
class Point(object):
    def __init__(self, x=0, y=0):
        self.x, self.y = x, y
    def __str__(self):
        return 'Les coordonnées du point sont \n
        x = {}\n
        y = {}'.format(self.x, self.y)
```

Exercice 2 : Création d'une classe

1. Définir une classe « Fraction » représentant les fractions rationnelles réduites.
2. Créer un constructeur qui possède les caractéristiques suivantes :
   - Gestion de valeurs par défaut : numérateur et dénominateur initialisés à 1 ;
   - Interdiction d'instancier une fraction ayant un dénominateur nul ;
   - Définition de trois attributs d'instance :
     - **num** : Valeur absolue du numérateur ;
     - **den** : Valeur absolue du dénominateur ;
     - **signe** : Signe de la fraction (+1 ou -1).
3. Définir la méthode spéciale « __str__() », permettant d'afficher la fraction.
   Exemple d'affichage : **(-5/10)**
4. Définir les méthodes de surcharge d'opérateurs suivantes :
   - **__neg__(self)** retourne la fraction opposée ;
   - **__add__(self, other)** retourne la fraction somme ;
   - **__sub__(self, other)** retourne la fraction différence ;
   - **__mul__(self, other)** retourne la fraction produit.
5. Créer une méthode « simpliFrac() » qui permet de simplifier une fraction (modification des attributs de la fraction à simplifier).
   Réutiliser l'algorithme d'Euclide codé dans un précédent TP.
6. Modifier les méthodes de cette classe pour qu'elles rendent un résultat simplifié lorsque c'est nécessaire.

2 Héritage surcharge et polymorphisme

Exercice 3 : Compte bancaire simple

Nous nous intéressons à un compte simple caractérisé par un solde exprimé en Euros qui peut être positif ou négatif.

1. Créer une classe « CompteSimple » respectant les caractéristiques ci-dessus.
2. Surcharger la méthode « __str__() » afin d'obtenir l'affichage suivant : Le solde du compte est de XXX Euro(s).
3. Créer une méthode « enregistrerOperation() » qui permet de créditer le compte ou de le débiter.
4. Tester cette classe.

Exercice 4 : Compte courant

Une banque conserve pour chaque compte l'historique des opérations qui le concernent (on se limite ici aux opérations de crédits et de débits). On souhaite modéliser un tel compte qu'on appelle compte courant. En plus des méthodes d'un compte simple, un compte courant offre les méthodes suivantes :

— **afficherReleve** : affiche l'ensemble des opérations effectuées ;

— **afficherReleveCredits** : affiche seulement les opérations de crédit ;

— **afficherReleveDebits** : affiche seulement les opérations de débit.

Pour représenter l'historique, on utilisera une liste. Pour enregistrer une opération, on conservera simplement le montant de l'opération (Crédit : positif ; Débit : négatif).

1. Créer la classe « CompteCourant ».
2. Tester cette classe.

Exercice 5 : La banque

Une banque est un organisme qui gère un grand nombre de comptes, qu'ils soient simples ou courants.

1. Créer la classe « Banque ».
   On choisit de stocker tous les comptes dans une unique liste.
2. Définir une méthode pour ouvrir un compte simple et une méthode pour ouvrir un compte courant.
3. Définir une méthode qui donne le cumul des soldes disponibles sur chacun des comptes.
4. On considère que la banque prélève périodiquement des frais de tenue de compte. Définir une méthode qui débite sur tous les comptes la somme de 2 Euros.
5. Écrire une méthode qui affiche le solde de tous les comptes de la banque (appel de la fonction « __str__() » de chaque compte).
6. Écrire une méthode qui édite les relevés de tous les comptes courants (seulement les comptes courants).

Exercice 6 : Numéro de compte

Chaque compte en France doit être identifié par un numéro IBAN dont le format est le suivant : **FRkk BBBB BGGG GGCC CCCC CCCC CKK**

— FRkk = code pays ;

- B = code banque ;
- G = code guichet ;
- C = numéro de compte ;
- K = clef.

Dans notre classe « Banque » on prendra :

- « FR76 » pour le code pays ;
- « 12345 » pour le code banque ;
- « 00001 » pour le code guichet (On suppose que notre banque n'a qu'une seule agence) ;
- Un numéro incrémenté de 9 chiffres pour chaque numéro de compte auquel on ajoute à la fin un code de 2 chiffres pour le type de compte :
  - Compte simple : Commence par « 000000001 » et on concatène « 60 » à la fin ;
  - Compte courant : Commence par « 000000001 » et on concatène « 40 » à la fin.
- La clé est une valeur calculée par la fonction : KK=(CC CCCC CCCC C) modulo 2.
  (La fonction pour calculer la clé est volontairement très simple pour ne pas s'attarder sur ce point)

1. Modifier les classes « CompteSimple » et « CompteCourant » afin d'ajouter un attribut contenant l' « IBAN ».
2. Modifier les méthodes d'affichage de la classe « CompteSimple » et « CompteCourant » afin d'afficher l' « IBAN » du compte.

3 Références

- TP Python 3, L. POINTAL ;
- TP Programmation objet (Stage Python Liesse), X. CREGUT.