Programmer avec Python
Exercices

Séance n° 3	**La programmation orientée objet**

Objectif : Écrire et utiliser des classes basiques

1 Rectangle

1. Écrivez une classe qui permet de représenter un rectangle. Cette classe devra posséder deux attributs, représentant la longueur et la largeur du rectangle. Lorsqu'il instanciera un nouveau rectangle, l'utilisateur devra être obligé de préciser sa longueur et sa largeur.
2. Écrivez un programme de test qui crée 3 rectangles de la taille de votre choix. Calculez et affichez la surface de chacun. Combien de fois avez-vous dû écrire le calcul ?
3. Complétez alors votre classe Rectangle : ajoutez une méthode afficherSurface() qui permet d'afficher la surface d'un rectangle.
4. Adaptez votre programme de test pour appeler cette méthode et vérifiez qu'elle fonctionne bien
5. Si vous désirez que votre programme de test trouve le plus grand de ces trois rectangles, pourquoi cette méthode ne convient-elle pas ? Ajoutez une méthode calculerSurface() dans la classe Rectangle qui retourne la surface du rectangle
6. Complétez votre programme de test en affichant quel rectangle possède la plus grande surface

2 Arbre

1. Écrivez une classe qui permet de représenter un arbre. On veut connaître l'age (en années) et la taille (en cm) de l'arbre . Quand on le crée, il a une taille et un âge égaux à zéro
2. Écrivez une méthode afficher qui permet d'afficher les caractéristiques de l'arbre à l'écran
3. Écrivez une méthode grandir qui prend en paramètre une taille. Cette méthode augmente l'âge de l'arbre de 1 an et augmente sa taille de la taille reçue en paramètre.
4. Testez votre classe en créant un programme principal qui crée, fait grandir et affiche les propriétés d'un arbre

3 Compte bancaire

3.1 Sans découvert autorisé

Écrire une classe qui permet de simuler un compte bancaire. Un compte bancaire possède un nom de titulaire et un solde. Le titulaire est obligatoire, et le solde est initialisé à zéro lors de la création du compte. On vous demande ensuite d'ajouter les opérations suivantes :

1. **créditer**, qui prend en paramètre un montant et le crédite au compte
2. **débiter**, qui prend en paramètre un montant et le débite du compte. Attention, si le compte ne possède pas assez d'argent, on affiche un message d'erreur "débit refusé" et on ne modifie pas le solde (interdiction d'avoir un solde négatif)
3. **afficher**, qui affiche le nom du titulaire et le solde du compte

Testez ensuite votre classe et toutes ses méthodes dans un programme principal.

3.2 Avec découvert

On souhaite maintenant autoriser un découvert à nos comptes bancaires.

1. Modifiez votre classe pour rajouter un attribut **decouvert_autorise**. Cet attribut devra forcément être précisé quand on crée le compte.
2. Modifiez votre méthode débiter afin qu'elle ne bloque les débit que s'ils amèneraient le solde en-dessous du découvert autorisé
3. Ajoutez une méthode **est_a_decouvert** qui renvoie True si le solde est négatif, False sinon
4. Testez !

4 Vinyles

On va maintenant réaliser un programme qui gère les disques vinyles d'un collectionneur. Pour ceci, on veut retenir les informations suivantes :

- le titre du vinyle
- le nom de l'artiste
- la note du vinyle du site Discogs (un réel allant de 1 à 5)
- l'année de parution

On vous demande ensuite de réaliser les étapes suivantes :

1. Écrivez la classe Vinyle. Toutes les caractéristiques du vinyle doivent être renseignées quand on le crée.
2. Ajoutez une méthode afficher à la classe vinyle qui affichera toutes les caractéristiques du vinyle

On va maintenant passer au programme principal qui gère les vinyles (en-dehors de votre classe).

1. Écrivez une fonction qui prend en paramètre une liste, demande les caractéristiques d'un vinyle à l'utilisateur, crée un vinyle et l'ajoute à la liste.
2. Écrivez une fonction qui prend en paramètre une liste de vinyles et un nom d'artiste, et affiche tous les vinyles de cet artiste
3. Écrivez une fonction qui prend en paramètre une liste de vinyles et une année, et affiche tous les vinyles sortis cette année-là
4. Écrivez une fonction qui prend en paramètre une liste de vinyles et un artiste. Elle doit afficher tous les vinyles de l'artiste donné dont la note est supérieure à la moyenne des notes de tous les vinyles de cet artiste.
5. Écrivez un menu qui boucle qui permet d'appeler toutes ces fonctions

5 Chronomètre

Écrire une classe Chrono qui permet de simuler un chronomètre. On veut qu'il dispose des méthodes suivantes :

1. **demarrer**, qui permet de démarrer le chronomètre. Tous les temps seront comptabilisés à partir du moment où le chronomètre a été démarré
2. **afficher_temps_ecoule**, qui permet d'afficher le temps écoulé depuis que le chrono a été lancé. Si le chrono n'a pas été lancé, on affiche un message d'erreur à la place.

On vous précise qu'il existe une fonction disponible dans le module time qui permet de donner l'heure actuelle (en fait, le nombre de secondes écoulées depuis le 1er janvier 1970). Le module time s'utilise en écrivant `import time` tout en haut de votre programme. Ensuite, vous pouvez récupérer le temps actuel en appelant la fonction `time.time()`.
Pour tester, vous pouvez utiliser la console et vérifier que les intervalles de temps entre lesquels vous appelez vos fonctions vous semblent cohérents.