Programmation objet

Exercices corrigés

Exercice 1 - Créer et manipuler des rectangles

Consignes :

Créer une classe Rectangle :

- Attributs :
  - longueur : un entier ou un flottant (la longueur du rectangle).
  - largeur : un entier ou un flottant (la largeur du rectangle).
- Méthodes :
  - __init__(self, longueur, largeur) : initialise les attributs longueur et largeur.
  - perimetre(self) : calcule et retourne le périmètre du rectangle.
  - surface(self) : calcule et retourne la surface du rectangle.
  - afficher_infos(self) : affiche la longueur, la largeur, la surface et le périmètre du rectangle.

Étape supplémentaire :

- Ajouter une méthode pour redimensionner le rectangle en modifiant ses attributs longueur et largeur.

Exemple d'utilisation :

```
# Créer un rectangle
rectangle1 = Rectangle(10, 5)

# Afficher les informations du rectangle
rectangle1.afficher_infos()

# Calculer et afficher le périmètre
print("Périmètre :", rectangle1.perimetre())

# Calculer et afficher la surface
print("Surface :", rectangle1.surface())
```

Correction :

```
class Rectangle:
    # Constructeur de la classe, initialise longueur et largeur
    def __init__(self, longueur, largeur):
        self.longueur = longueur
        self.largeur = largeur

    # Méthode pour calculer le périmètre du rectangle
    def perimetre(self):
        return 2 * (self.longueur + self.largeur)

    # Méthode pour calculer la surface du rectangle
    def surface(self):
        return self.longueur * self.largeur

    # Méthode pour afficher les informations du rectangle
    def afficher_infos(self):
        print(f"Longueur : {self.longueur}")
        print(f"Largeur : {self.largeur}")
        print(f"Surface : {self.surface()}")
        print(f"Périmètre : {self.perimetre()}")


# Méthode pour redimensionner le rectangle
    def redimensionner(self, nouvelle_longueur, nouvelle_largeur):
        self.longueur = nouvelle_longueur
        self.largeur = nouvelle_largeur
```

Exercice 2 : Gestion d'un compte bancaire

Objectif : Créer une classe représentant un compte bancaire et permettre de gérer des opérations simples comme déposer de l'argent, retirer de l'argent et consulter le solde.

Consignes :

Créer une classe CompteBancaire :

Attributs :

- titulaire : une chaîne de caractères (le nom du titulaire du compte).
- solde : un entier ou un flottant (le montant disponible sur le compte, initialisé à 0).

Méthodes :

- __init__(self, titulaire) : initialise le titulaire et le solde (par défaut à 0).
- deposer(self, montant) : ajoute un montant au solde du compte.
- retirer(self, montant) : retire un montant du solde si le solde est suffisant.

Il faut vérifier que le retrait ne dépasse pas le solde disponible et afficher un message d'erreur si le solde est insuffisant.

- afficher_solde(self) : affiche le solde actuel du compte.

Exemple d'utilisation :

```
# Création d'un compte bancaire pour "Alice"
compte_alice = CompteBancaire("Alice")

# Afficher le solde initial
compte_alice.afficher_solde()

# Déposer de l'argent
compte_alice.deposer(200)

# Retirer de l'argent
compte_alice.retirer(50)

# Afficher le solde après les opérations
compte_alice.afficher_solde()
```

Correction :

```
class CompteBancaire:
    # Constructeur, initialise le titulaire et le solde à 0 par défaut
    def __init__(self, titulaire):
        self.titulaire = titulaire
        self.solde = 0

    # Méthode pour déposer de l'argent
    def deposer(self, montant):
        self.solde += montant
        print(f"Montant déposé : {montant} euros. Nouveau solde : {self.solde} euros.")

    # Méthode pour retirer de l'argent si le solde est suffisant
    def retirer(self, montant):
        if montant <= self.solde:
            self.solde -= montant
            print(f"Montant retiré : {montant} euros. Nouveau solde : {self.solde} euros.")
        else:
            print("Retrait impossible : solde insuffisant.")

    # Méthode pour afficher le solde actuel
    def afficher_solde(self):
        print(f"Solde actuel de {self.titulaire} : {self.solde} euros.")
```

Exercice 3 : Gestion d'une Bibliothèque

Objectif : Créer une petite application de gestion de livres en utilisant la programmation orientée objet.

Consignes :

1. **Créer une classe Livre** :
    - Attributs :
        - titre : chaîne de caractères (le titre du livre)
        - auteur : chaîne de caractères (le nom de l'auteur)
        - annee_publication : entier (l'année de publication)
        - disponible : booléen (indique si le livre est disponible ou non). Le livre est disponible par défaut.
    - Méthodes :
        - __init__(self, titre, auteur, annee_publication) : initialise les attributs du livre avec les valeurs passées en paramètre.
        - emprunter(self) : cette méthode rend le livre indisponible (change l'attribut disponible à False), à condition qu'il soit disponible.
        - rendre(self) : cette méthode rend le livre disponible (change l'attribut disponible à True).
        - afficher_infos(self) : affiche les informations du livre (titre, auteur, année, disponibilité).
2. **Créer une classe Bibliotheque** :
    - Attributs :
        - livres : une liste d'objets de type Livre (cette liste contiendra tous les livres de la bibliothèque).
    - Méthodes :
        - ajouter_livre(self, livre) : ajoute un objet Livre à la liste livres.
        - afficher_livres(self) : affiche tous les livres de la bibliothèque avec leurs informations.
        - emprunter_livre(self, titre) : permet d'emprunter un livre en fonction de son titre. Il faut vérifier s'il est disponible.

- rendre_livre(self, titre) : permet de rendre un livre emprunté en fonction de son titre.

Exemple d'utilisation :

```
# Créer des livres
livre1 = ...
livre2 = ...

# Créer une bibliothèque
bibliotheque = ...

# Ajouter les livres à la bibliothèque
bibliotheque....
bibliotheque....

# Afficher les livres de la bibliothèque
bibliotheque....

# Emprunter un livre
bibliotheque....

# Tenter de ré-emprunter le même livre
bibliotheque....

# Rendre le livre
bibliotheque....

# Afficher à nouveau les livres de la bibliothèque
bibliotheque.afficher_livres()
```

```
class Livre:
  # Constructeur pour initialiser le titre, l'auteur, l'année et la disponibilité


  # Méthode pour emprunter un livre (le rendre indisponible)


  # Méthode pour rendre un livre (le rendre disponible)


  # Méthode pour afficher les informations du livre
  def afficher_infos(self):
    disponibilite = "disponible" if self.disponible else "non disponible"
    print(f"Titre : {self.titre}, Auteur : {self.auteur}, Année : {self.annee_publication},
Disponibilité : {disponibilite}")
```

```
class Bibliotheque:
  # Constructeur pour initialiser la liste de livres


  # Méthode pour ajouter un livre à la bibliothèque


  # Méthode pour afficher tous les livres de la bibliothèque


  # Méthode pour emprunter un livre par son titre
  def emprunter_livre(self, titre):
    trouve = False
    for livre in self.livres:
      if livre.titre == titre:
        livre.....
        trouve = True
        break
    if !trouve :
      print(f"Le livre {titre} n'est pas disponible dans la bibliothèque.")

  # Méthode pour rendre un livre par son titre
  def rendre_livre(self, titre):
    trouve = False
    for livre in self.livres:
      if livre.titre == titre:
        livre.....
        trouve = True
        break
    if !trouve :
      print(f"Le livre {titre} n'a pas été trouvé dans la bibliothèque.")
```

Correction :

```
class Livre:
    # Constructeur pour initialiser le titre, l'auteur, l'année et la disponibilité
    def __init__(self, titre, auteur, annee_publication):
        self.titre = titre
        self.auteur = auteur
        self.annee_publication = annee_publication
        self.disponible = True  # Le livre est disponible par défaut

    # Méthode pour emprunter un livre (le rendre indisponible)
    def emprunter(self):
        if self.disponible:
            self.disponible = False
            print(f"Vous avez emprunté le livre : {self.titre}")
        else:
            print(f"Le livre {self.titre} n'est pas disponible.")

    # Méthode pour rendre un livre (le rendre disponible)
    def rendre(self):
        if not self.disponible:
            self.disponible = True
            print(f"Vous avez rendu le livre : {self.titre}")
        else:
            print(f"Le livre {self.titre} est déjà disponible.")

    # Méthode pour afficher les informations du livre
    def afficher_infos(self):

        disponibilite = ""

        if self.disponible :

            disponibilite = "disponible"

        else :

            disponibilite = "non disponible"

        print(f"Titre : {self.titre}, Auteur : {self.auteur}, Année : {self.annee_publication},
Disponibilité : {disponibilite}")
```

```
class Bibliotheque:
  # Constructeur pour initialiser la liste de livres
  def __init__(self):
    self.livres = []

  # Méthode pour ajouter un livre à la bibliothèque
  def ajouter_livre(self, livre):
    self.livres.append(livre)
    print(f"Le livre {livre.titre} a été ajouté à la bibliothèque.")

  # Méthode pour afficher tous les livres de la bibliothèque
  def afficher_livres(self):
    if self.livres:
      print("Livres disponibles dans la bibliothèque :")
      for livre in self.livres:
        livre.afficher_infos()
    else:
      print("La bibliothèque ne contient aucun livre pour l'instant.")

 # Méthode pour emprunter un livre par son titre
  def emprunter_livre(self, titre):
    trouve = False
    for livre in self.livres:
      if livre.titre == titre:
        livre.....
        trouve = True
        break
    if !trouve :
      print(f"Le livre {titre} n'est pas disponible dans la bibliothèque.")

  # Méthode pour rendre un livre par son titre
  def rendre_livre(self, titre):
    trouve = False
    for livre in self.livres:
      if livre.titre == titre:
        livre.......
        trouve = True
        break
    if !trouve :
      print(f"Le livre {titre} n'a pas été trouvé dans la bibliothèque.")
```

```
# Créer des livres
livre1 = Livre("Le Petit Prince", "Antoine de Saint-Exupéry", 1943)
livre2 = Livre("1984", "George Orwell", 1949)
livre3 = Livre("Les Misérables", "Victor Hugo", 1862)

# Créer une bibliothèque
bibliotheque = Bibliotheque()

# Ajouter des livres à la bibliothèque
bibliotheque.ajouter_livre(livre1)
bibliotheque.ajouter_livre(livre2)
bibliotheque.ajouter_livre(livre3)

# Afficher tous les livres de la bibliothèque
bibliotheque.afficher_livres()

# Emprunter un livre
bibliotheque.emprunter_livre("1984")

# Essayer d'emprunter un livre déjà emprunté
bibliotheque.emprunter_livre("1984")

# Rendre un livre
bibliotheque.rendre_livre("1984")

# Essayer de rendre un livre déjà disponible
bibliotheque.rendre_livre("1984")

# Afficher à nouveau tous les livres de la bibliothèque
bibliotheque.afficher_livres()
```