Modularité

Exercices

Exercice 1 : Création d'un module de gestion de bibliothèque

Objectif

Vous devez créer un module Python qui permet de gérer une petite bibliothèque de livres. Le but est de vous familiariser avec la création de fonctions simples, l'organisation du code en modules, et la documentation de ce module.

Étapes

1. **Créer un module** appelé bibliotheque.py qui contiendra plusieurs fonctions pour manipuler des livres.
2. **Chaque fonction** est déjà documentée avec une docstring explicative, expliquant les paramètres et la valeur de retour. A vous d'implémenter le code.
3. **Tester le module** en créant un fichier Python séparé (test_bibliotheque.py) qui importe le module et utilise les fonctions.

Spécifications du module

Le module devra contenir les fonctions suivantes :

1. **Ajouter un livre**

```
def ajouter_livre(bibliotheque, titre, auteur):
    """
    Ajoute un livre à la bibliothèque.

    Paramètres:
    - bibliotheque (list): La liste des livres (chaque livre est représenté par un dictionnaire).
    - titre (str): Le titre du livre à ajouter.
    - auteur (str): L'auteur du livre à ajouter.

    Retourne:
    - list: La bibliothèque mise à jour avec le nouveau livre.
    """
    pass  # Remplacez par votre code
```

2. Supprimer un livre

```
def supprimer_livre(bibliotheque, titre):
    """
    Supprime un livre de la bibliothèque selon son titre.

    Paramètres:
    - bibliotheque (list): La liste des livres.
    - titre (str): Le titre du livre à supprimer.

    Retourne:
    - list: La bibliothèque mise à jour sans le livre.
    """
    pass  # Remplacez par votre code
```

3. Rechercher un livre par titre

```
def rechercher_livre(bibliotheque, titre):
    """
    Recherche un livre dans la bibliothèque selon son titre.

    Paramètres:
    - bibliotheque (list): La liste des livres.
    - titre (str): Le titre du livre à rechercher.

    Retourne:
    - dict ou None: Le livre correspondant s'il est trouvé, sinon None.
    """
    pass  # Remplacez par votre code
```

4. Lister tous les livres

```
def lister_livres(bibliotheque):
    """
    Liste tous les livres de la bibliothèque.

    Paramètres:
    - bibliotheque (list): La liste des livres.

    Retourne:
    - None: Affiche chaque livre avec son titre et son auteur.
    """
    pass  # Remplacez par votre code
```

Exemple de structure d'un livre

Chaque livre dans la bibliothèque est un dictionnaire avec les clés suivantes :

```
livre = {
    'titre': 'Le Petit Prince',
    'auteur': 'Antoine de Saint-Exupéry'
}
```

Fichier de test (test_bibliotheque.py)

Dans un fichier séparé, vous devez tester le module en important les fonctions du module **bibliotheque.py** et en les utilisant :

```
...

# Initialisation de la bibliothèque
ma_bibliotheque = []

# Ajouter des livres
ma_bibliotheque =
ma_bibliotheque =

# Lister les livres


# Rechercher un livre
livre =
print(livre)

# Supprimer un livre
ma_bibliotheque =

# Lister les livres après suppression
lister_livres(ma_bibliotheque)
```

Corrigé :

Fichier : bibliotheque.py

```
# Module bibliotheque.py

def ajouter_livre(bibliotheque, titre, auteur):
    """
    Ajoute un livre à la bibliothèque.

    Paramètres:
    - bibliotheque (list): La liste des livres (chaque livre est représenté par un dictionnaire).
    - titre (str): Le titre du livre à ajouter.
    - auteur (str): L'auteur du livre à ajouter.

    Retourne:
    - list: La bibliothèque mise à jour avec le nouveau livre.
    """
    livre = {'titre': titre, 'auteur': auteur}
    bibliotheque.append(livre)
    return bibliotheque


def supprimer_livre(bibliotheque, titre):
    """
    Supprime un livre de la bibliothèque selon son titre.

    Paramètres:
    - bibliotheque (list): La liste des livres.
    - titre (str): Le titre du livre à supprimer.

    Retourne:
    - list: La bibliothèque mise à jour sans le livre.
    """
    for livre in bibliotheque:
        if livre['titre'] == titre:
            bibliotheque.remove(livre)
            break
    return bibliotheque
```

```
def rechercher_livre(bibliotheque, titre):
    """
    Recherche un livre dans la bibliothèque selon son titre.

    Paramètres:
    - bibliotheque (list): La liste des livres.
    - titre (str): Le titre du livre à rechercher.

    Retourne:
    - dict ou None: Le livre correspondant s'il est trouvé, sinon None.
    """
    for livre in bibliotheque:
        if livre['titre'] == titre:
            return livre
    return None


def lister_livres(bibliotheque):
    """
    Liste tous les livres de la bibliothèque.

    Paramètres:
    - bibliotheque (list): La liste des livres.

    Retourne:
    - None: Affiche chaque livre avec son titre et son auteur.
    """
    if len(bibliotheque) == 0:
        print("La bibliothèque est vide.")
    else:
        for livre in bibliotheque:
            print(f"Titre : {livre['titre']}, Auteur : {livre['auteur']}")
```

Fichier : test_bibliotheque.py

```
# Fichier test_bibliotheque.py
from bibliotheque import ajouter_livre, supprimer_livre, rechercher_livre, lister_livres

# Initialisation de la bibliothèque
ma_bibliotheque = []

# Ajouter des livres
print("Ajout de livres dans la bibliothèque :")
ma_bibliotheque = ajouter_livre(ma_bibliotheque, 'Le Petit Prince', 'Antoine de Saint-
Exupéry')
ma_bibliotheque = ajouter_livre(ma_bibliotheque, '1984', 'George Orwell')

# Lister les livres
print("\nListe des livres après ajouts :")
lister_livres(ma_bibliotheque)

# Rechercher un livre
print("\nRecherche du livre '1984' :")
livre_trouve = rechercher_livre(ma_bibliotheque, '1984')
if livre_trouve:
   print(f"Livre trouvé : Titre = {livre_trouve['titre']}, Auteur = {livre_trouve['auteur']}")
else:
   print("Livre non trouvé")

# Supprimer un livre
print("\nSuppression du livre '1984' :")
ma_bibliotheque = supprimer_livre(ma_bibliotheque, '1984')

# Lister les livres après suppression
print("\nListe des livres après suppression :")
lister_livres(ma_bibliotheque)
```

Exercice 2 : La création d'un **paquetage Python** avec deux modules

Objectif de l'exercice

1. Créer un paquetage Python avec deux modules.
2. Utiliser des **docstrings** pour documenter les fonctions dans chaque module.
3. Importer et utiliser les fonctions des modules dans un programme principal.

Étapes de l'exercice

1. Création de la structure du paquetage

Vous allez d'abord créer la structure suivante pour leur projet :

```
calculs/
    __init__.py
    arithmetique.py
    geometrie.py
main.py
```

- `calculs/` : Le dossier principal qui contient les modules.
- `__init__.py` : Un fichier vide qui indique à Python que `calculs` est un paquetage.
- `arithmetique.py` : Un module qui contiendra des fonctions arithmétiques.
- `geometrie.py` : Un module qui contiendra des fonctions liées à la géométrie.
- `main.py` : Le programme principal qui importera et utilisera les modules.

2. **Contenu du module arithmetique.py**

Vous devez écrire deux fonctions simples dans ce module, avec des **docstrings** pour documenter leur fonctionnement.

```
# arithmetique.py

def addition(a, b):



def soustraction(a, b):

```

3. **Contenu du module geometrie.py**

```
# geometrie.py

def aire_rectangle(longueur, largeur):



def perimetre_rectangle(longueur, largeur):

```

4. Utilisation des modules dans main.py

Dans ce fichier, vous devez importer les fonctions des modules arithmetique et geometrie, puis les utiliser pour effectuer des calculs.

```
# main.py

#imports


# Test des fonctions du module arithmetique
resultat_addition =
resultat_soustraction =

print(f"Addition : 10 + 5 = {resultat_addition}")
print(f"Soustraction : 10 - 5 = {resultat_soustraction}")

# Test des fonctions du module geometrie
aire =
perimetre =

print(f"Aire du rectangle (5x3) = {aire}")
print(f"Périmètre du rectangle (5x3) = {perimetre}")
```