Programmation objet

Corrections de la fiche d'exercices n° 8

Exercice 1 - Adresses IPv4

```
class AdresseIP:
    def __init__(self, adresse):
        self.adresse = adresse

    def liste_octets(self):
        """renvoie une liste de nombres entiers,
        la liste des octets de l'adresse IP"""
        # Note : split découpe la chaine de caractères
        # en fonction du séparateur
        return [int(i) for i in self.adresse.split(".")]

    def est_reservee(self):
        """renvoie True si l'adresse IP est une adresse
        réservée, False sinon"""
        reservees = [ '192.168.0.0', '192.168.0.255' ]
        return self in reservees

    def adresse_suivante(self):
        """renvoie un objet de AdresseIP avec l'adresse
        IP qui suit l'adresse self si elle existe et None sinon"""
        octets = self.liste_octets()
        if octets[3] == 254:
            return None
        octet_nouveau = octets[3] + 1
        return AdresseIP('192.168.0.' + str(octet_nouveau))


adresse1 = AdresseIP('192.168.0.1')
adresse2 = AdresseIP('192.168.0.2')
adresse3 = AdresseIP('192.168.0.0')

print(adresse1.liste_octets()) #[192, 168, 0, 1]
print(adresse1.est_reservee()) #False
print(adresse3.est_reservee()) #True
print(adresse2.adresse_suivante().adresse)
```

Exercice 2 - Livres de science-fiction

Citer un attribut et une méthode de la classe Livre.

Dans la classe `Livre`, les attributs sont les suivants :

- `id` : L'identifiant du livre.
- `titre` : Le titre du livre.
- `auteur` : Le nom de l'auteur du livre.
- `ann_pub` : L'année de première publication du livre.
- `note` : La note attribuée au livre sur une échelle de 1 à 10.

Méthodes :
get_id(self) :
get_titre(self):
get_auteur(self):
get_ann_pub(self):

Écrire la méthode get_note de la classe Livre. Cette méthode devra renvoyer la note d'un livre.

```
def get_note(self):
    return self.note
```

Écrire le programme permettant d'ajouter le livre Blade Runner à la fin de la "bibliothèque" en utilisant la classe Livre et la classe Bibliotheque (voir le tableau en début d'exercice).

```
# Création d'une instance de la classe Livre pour "Blade Runner"
livre_blade_runner = Livre(8, 'Blade Runner', 'K.Dick', 1968, 8)

# Création d'une instance de la classe Bibliotheque
bibliotheque = Bibliotheque()

# Ajout du livre "Blade Runner" à la bibliothèque
bibliotheque.ajout_livre(livre_blade_runner)
```

Recopier et compléter la méthode titre_livre de la classe Bibliotheque.
Cette méthode prend en paramètre l'identifiant d'un livre et renvoie le titre du livre si l'identifiant existe, ou None si l'identifiant n'existe pas.

```
class Bibliotheque:
    def __init__(self):
        self.liste_livre = []

    def ajout_livre(self, livre):
        self.liste_livre.append(livre)

    def titre_livre(self, id_livre):
        for livre in self.liste_livre:
            if livre.get_id() == id_livre:
                return livre.get_titre()
        return None
```

Exercice 3

```
class Plat:
    def __init__(self, nom, prix, quantite):
        """
        Constructeur de la classe Plat.
        :param nom: Le nom du plat (chaîne de caractères)
        :param prix: Le prix unitaire du plat (float)
        :param quantite: La quantité commandée (entier)
        """
        self.nom = nom
        self.prix = prix
        self.quantite = quantite

    def modifier_quantite(self, nouvelle_quantite):
        """
        Modifie la quantité du plat.
        :param nouvelle_quantite: La nouvelle quantité à fixer (entier)
        """
        self.quantite = nouvelle_quantite

    def calculer_sous_total(self):
        """
        Calcule le sous-total du plat (prix * quantité).
        :return: Le sous-total du plat (float)
        """
        return self.prix * self.quantite
```

```
class Commande:
    def __init__(self):
        """
        Constructeur de la classe Commande.
        Initialise une liste vide de plats.
        """
        self.plats = []

    def ajouter_plat(self, plat):
        """
        Ajoute un plat à la commande.
        :param plat: Un objet de type Plat à ajouter
        """
        self.plats.append(plat)

    def supprimer_plat(self, nom_plat):
        """
        Supprime un plat de la commande en fonction de son nom.
        :param nom_plat: Le nom du plat à supprimer (chaîne de caractères)
        """
        self.plats = [plat for plat in self.plats if plat.nom != nom_plat]

    def calculer_total(self):
        """
        Calcule le total de la commande en additionnant les sous-totaux des plats.
        :return: Le total de la commande (float)
        """
        total = 0
        for plat in self.plats:
            total += plat.calculer_sous_total()
        return total
```