EXERCICE 2 (4 points)

Cet exercice aborde les notions de classes, itération et récursivité

Une petite société immobilière ne voulant pas investir dans une base de données nécessitant une mise en place longue et fastidieuse a créé un fichier .csv pour stocker ses annonces. La consultation et la mise à jour de ce fichier ne seront pas étudiées ici. Pour limiter notre étude, nous considérerons que les données sont stockées temporairement dans une liste v dont voici la structure simplifiée :

```
# pieces
class Piece:
    def __init__(self, a, b):
        self.nom = a   # nom
        self.sup = b   #superficie de la piece
    def sup(self):
        return self.sup
#villas
class Villa:
    def __init__(self, a, b, c, d, e):
        self.nom = a # nom de la villa
        self.sejour = b  # caracteristiques sejour
        self.ch1   = c # caracteristiques de la 1ere chambre
        self.ch2   = d # caracteristiques de la 2eme chambre
        self.eqCuis = e  # equipement de la cuisine "eq" ou "noneq"
    def nom(self):
        return self.nom
    def surface(self):
        return  … …
    def equip(self):
        return self.eqCuis

# Programme principal
v=[]
v.append( Villa("Les quatre vents", Piece("séjour",40),
Piece("ch1",10), Piece("ch2",20),"eq"))
v.append( Villa("Les goélands", Piece("séjour",50),
Piece("ch1",15), Piece("ch2",15), "eq"))
v.append( Villa("Rêve d'été", Piece("séjour",30), Piece("ch1",15),
Piece("ch2",20), "non eq"))
v.append( Villa("Les oliviers", Piece("séjour",30),
Piece("ch1",10), Piece("ch2",20), "eq"))
v.append( Villa("Bellevue", Piece("séjour",30), Piece("ch1",10),
Piece("ch2",20), "non eq"))
```

La structure de données retenue pour l'exercice est définie par la classe Villa.

Partie A : Analyse du code et complétion

1.a) Combien d'éléments contient la liste `v` ?

1.b) Que retourne l'instruction `v[1].nom()` ?

Pour accéder à l'information de la surface habitable du logement, le développeur souhaite ajouter la méthode `surface()` qui renvoie cette surface.

1.c) Compléter sur votre copie la méthode `surface()`.

```
def surface(self):
    return  ... ...
```

L'agent immobilier veut pouvoir consulter les villas qui disposent d'une cuisine équipée. Pour cela vous devez écrire la portion du programme qui affichera la liste des villas équipées. Elle devra parcourir séquentiellement la liste et afficher à l'écran le nom de chaque villa équipée.

2. Rédigez sur votre copie la portion de programme réalisant cette sélection.

Partie B : Récursivité

L'agent immobilier veut répondre à une demande d'un client : " Quelle est votre villa la plus grande ?" Pour cela, il souhaite disposer d'une fonction `max_surface()` qui va extraire de la liste `v`, la villa désirée. Nous avons décidé d'écrire cette fonction de façon récursive.

3. Recopiez parmi les propositions suivantes celle qui caractérise un appel récursif.

- appel d'une fonction par elle-même.
- appel dont l'exécution est un processus itératif.
- appel d'une fonction comportant une boucle.

L'algorithme suivant a été choisi :

Il faut partir d'une liste de villas :

- si cette liste contient un seul élément, c'est le résultat ;
- si la liste en contient plusieurs, il faut analyser les deux premiers éléments, éliminer la villa de plus petite surface, et recommencer avec la liste tronquée.

A la fin du processus, une seule villa est renvoyée.

Pour écrire cette fonction, dans un premier temps, nous allons donc distinguer deux cas:

- celui où la liste des villas ne contient qu'une villa :

    il faut renvoyer la villa

- celui où la liste en contient au moins deux :

    si la surface de la villa `v[0]` est inférieure à celle de la villa `v[1]`

        il faut supprimer `v[0]` de la liste

    sinon

        il faut supprimer `v[1]` de la liste.

4. Ecrivez sur votre copie le code de cette fonction en Python.

```
def max_surface(v):
....
```