EXERCICE 1 (4 points)

Écrire une fonction `moyenne` prenant en paramètres une liste d'entiers et qui renvoie la moyenne des valeurs de cette liste.

Exemple :

```
>>> moyenne([10,20,30,40,60,110])
45.0
```

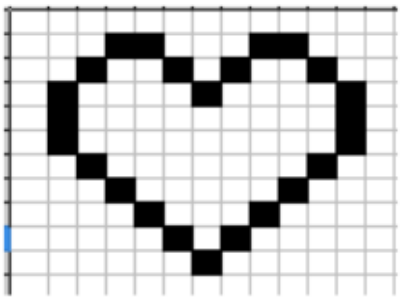
EXERCICE 2 (4 points)

On travaille sur des dessins en noir et blanc obtenu à partir de pixels noirs et blancs :

La figure « cœur » ci-contre va servir d'exemple.

On la représente par une grille de nombres, c'est-à-dire par une liste composée de sous-listes de mêmes longueurs.

Chaque sous-liste représentera donc une ligne du dessin.



Dans le code ci-dessous, la fonction `affiche` permet d'afficher le dessin. Les pixels noirs (1 dans la grille) seront représentés par le caractère `"*"` et les blancs (0 dans la grille) par deux espaces.

La fonction `zoomListe` prend en argument une liste `liste_depart` et un entier `k`. Elle renvoie une liste où chaque élément de `liste_depart` est dupliqué `k` fois.

La fonction `zoomDessin` prend en argument la grille `dessin` et renvoie une grille où toutes les lignes de `dessin` sont zoomées `k` fois et répétées `k` fois.
Soit le code ci-dessous :

```
coeur = [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], \
         [0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0], \
         [0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0], \
         [0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0], \
         [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0], \
         [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0], \
         [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0], \
         [0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0], \
         [0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0], \
         [0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0], \
         [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0], \
         [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]

def affiche(dessin):
    ''' affichage d'une grille : les 1 sont représentés par
```

```
        des "*" , les 0 par deux espaces "  " '''
    for ligne in dessin:
        for col in ligne:
            if col == 1:
                print(" *",end="")
            else:
                print("  ",end="")
        print()


def zoomListe(liste_depart,k):
    '''renvoie une liste contenant k fois chaque
       élément de liste_depart'''
    liste_zoom = ...
    for elt in ... :
        for i in range(k):
            ...
    return liste_zoom

def zoomDessin(grille,k):
    '''renvoie une grille où les lignes sont zoomées k fois
       ET répétées k fois'''
    grille_zoom=[]
    for elt in grille:
        liste_zoom = ...
        for i in range(k):
            ... .append(...)
    return grille_zoom
```

Résultats à obtenir :

```
>>> affiche(coeur)
```



```
>>> affiche(zoomDessin(coeur,3))
```

EXERCICE 2 (4 points)

On définit une classe gérant une adresse IPv4.

On rappelle qu'une adresse IPv4 est une adresse de longueur 4 octets, notée en décimale à point, en séparant chacun des octets par un point. On considère un réseau privé avec une plage d'adresses IP de 192.168.0.0 à 192.168.0.255.

On considère que les adresses IP saisies sont valides.

Les adresses IP 192.168.0.0 et 192.168.0.255 sont des adresses réservées.

Le code ci-dessous implémente la classe `AdresseIP`.

```
class AdresseIP:

    def __init__(self, adresse):
        self.adresse = ...

    def liste_octet(self):
        """renvoie une liste de nombres entiers,
           la liste des octets de l'adresse IP"""
        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"""
        return ... or ...

    def adresse_suivante(self):
        """renvoie un objet de AdresseIP avec l'adresse
           IP qui suit l'adresse self
           si elle existe et False sinon"""
        if ... < 254:
            octet_nouveau = ... + ...
            return AdresseIP('192.168.0.' + ...)
        else:
            return False
```

Compléter le code ci-dessus et instancier trois objets : `adresse1, adresse2, adresse3` avec respectivement les arguments suivants :
`'192.168.0.1', '192.168.0.2', '192.168.0.0'`

Vérifier que :

```
>>> adresse1.est_reservee()
False
>>> adresse3.est_reservee()
True
>>> adresse2.adresse_suivante().adresse
'192.168.0.3'
```