Exercice :

Afin de gérer numériquement les réservations de places dans chaque wagon de ses trains, la SNCF définit la classe `Wagon` comme ci-dssous.
Attention : il ne faudra pas confondre la classe `Wagon` (qui définit les objets de type `Wagon`) et la *classe* d'un wagon (nombre entier (`int`) qui vaut 1 pour un wagon de *première classe* ou qui vaut 2 pour un wagon de *deuxième classe*) qui caractérise le niveau de confort d'un wagon.

```
class Wagon :
    def __init__(self, classe):
        """constructeur qui renvoie un wagon vide dont la classe est
        passée en paramètre"""
        self.classe = classe
        if classe == 1:
            self.nb_de_sieges = 54
        else:
            self.nb_de_sieges = 88
        self.liste_place=[] #liste des occupants de chaque siège numeroté
        for i in range(self.nb_de_sieges):
            self.liste_place.append("vide")

    def get_nb_de_sieges(self):
        return self.nb_de_sieges

    def get_classe(self):
        return self.classe

    def get_occupant(self, numero_de_place):
        """renvoie le nom de l'occupant du siege dont le numero
        est passé en paramètre (ou "vide" si le siege est libre)"""
        return self.liste_place[numero_de_place]

    def set_occupant(self, numero_de_place, nouvel_occupant):
        self.liste_place[numero_de_place] = nouvel_occupant

    def reserver_place(self, numero_de_place, nom_passager):
        if self.get_occupant(numero_de_place) == "vide":
            self.set_occupant(numero_de_place, nom_passager)
        else:
            print("Réservation impossible")

    def mystere(self):
        res = 0
        for place in self.liste_place:
            if place == "vide":
                res = res + 1
        return res
```

1- Citer les attributs de la classe `Wagon` et préciser leur type.

2- Quel est le nombre de sièges libres dans un wagon de 2ème classe ?

3- Citer au moins un *accesseur* parmi les méthodes de cette classe `Wagon`.

4- Recopier et compléter la méthode `annuler_reservation` ci-dessous qui permet de libérer le siège occupé par un passager dont le nom est `nom_passager` (à condition que ce passager ait réservé un siège à son nom).

```
    def annuler_reservation(self, nom_passager):
        if ..........................:
            for numero_de_place in range (..........):
                if self.get_occupant(..........) == ..........:
                    ...................
        else:
            print("ce passager n'est pas enregistré dans ce wagon")
```

5- Quelle information, utile pour la SNCF, renvoie la méthode `mystere()` au sujet du wagon autoréférencé ?

La SNCF définit également une classe `Train` afin de gérer, pour chaque train (instance de la classe `Train`) :

- La gare de départ
- La gare d'arrivée
- La listes des wagons (instance de la classe `Wagon`) composant le train

```
class Train:
    def __init__(self, depart, arrivee):
        self.gare_depart = depart
        self.gare_arrivee = arrivee
        self.liste_wagons=[]

    def ajouter_wagon(self, nouveau_wagon):
        self.liste_wagons.append(nouveau_wagon)

    def nb_total_de_sieges(self):
        res = 0
        for wagon in self.liste_wagons:
            res = res + wagon.get_nb_de_sieges()
        return res
```

6- Ecrire une méthode `changer_gare_arrivee` qui permet de modifier l'attribut `gare_arrivee` du train autoréférencé (on parle de *mutateur*).
Exemple d'utilisation :

```
>>> t1 = Train("Paris", "Marseille")
>>> t1.changer_gare_arrivee("Lille")
>>> t1.gare_arrivee
'Lille'
>>> t1.changer_gare_arrivee("Brest")
```

```
>>> t1.gare_arrivee
'Brest'
```

7- Ecrire une méthode `nb_places_libres_en_classe` qui renvoie le nombre total de places libres (dont la classe est passée en paramètre) dans la totalité du train.
Exemple d'utilisation :

```
>>> t1.nb_places_libres_en_classe(1)
92
>>> t1.nb_places_libres_en_classe(2)
124
```