MENERAT Emy

BOUCHAMA Wallys

Projet 3 - Gestion d'un parking

Introduction

Dans ce projet, nous devions créer une application de gestion d'un parking. Celui-ci devais posséder 5 niveaux, avec par niveaux, 80 places toutes numérotées de 1 à 480, en sachant que le chiffre des centaines est le numéro du niveau. Pour cela, nous devions créer une classe Parking et une classe Voiture.

Les conditions à respecter

L'objet Voiture contient le numéro d'immatriculation de la voiture, sa marque, le nom de son propriétaire et un booléen qui indique si le propriétaire est abonné, auquel cas un numéro de place de parking lui est attribué.

L'objet Parking contient la liste des personnes abonnées et la liste des places. Ainsi que des méthodes qui permettent de :

- Récupérer les différentes information d'une voiture comme l'immatriculation, la marque ou encore le propriétaire
- Pouvoir s'abonner ou annuler son abonnement
- Affecter une voiture à une place
- Savoir si la place donné est libre ou qui l'occupe
- Renvoyer la liste des places d'abonnées occupées par d'autres voitures
- Connaître le nombre de places libres sans compter les places réservées aux abonnées

I. Répartition des tâches

Pour se partager notre travail, nous avons utilisé *Discord* et *GitHub*.

Wallys : Créer la classe *Parking*, la fonction pour créer les places (*place*), la fonction *ajout_place*, la fonction *si_voiture*, la fonction *places_abonnes_occupes* et la fonction *nombre_place*.

Emy : Créer la classe *Voiture*, la fonction *abonnement* et la fonction *desabonner*, la fonction *information*, le typst et la doc du code.

II. Explication du code

Si vous comptez réaliser des tests, veuillez suivre le même ordre que dans le code déjà présent.

Pour la classe Parking

Ce code créer la classe Parking et les étages et les places du parking de 1 à 80:

```
class Parking:

    def __init__(self):
        self.niv = 1
        self.place_niv1 = {}
        self.place_niv2 = {}
        self.place_niv3 = {}
        self.place_niv4 = {}
        self.place_niv5 = {}
        self.abonnes = []

    def place(self):
        for i in range(80):
            self.place_niv1[i] = False
            self.place_niv2[i] = False
            self.place_niv3[i] = False
            self.place_niv4[i] = False
            self.place_niv5[i] = False
```

Ce code permet de s'abonner ou de se désabonner :

```
def abonnement(self, voiture):
        if voiture.ab == False:
            a = str(input("voulez-vous vous abonnez : "))
            if a == "oui" or a == "Oui":
                voiture.ab = True
                self.abonnes.append(voiture)
        else:
            print("vous êtes déjà abonné")

    def desabonner(self, voiture):
        if voiture.ab == True:
                a = str(input("voulez-vous vous désabonner ? : "))
```

```
                if a == "oui" or a == "Oui":
                    voiture.ab = False
                    if voiture in self.abonnes:
                        self.abonnes.remove(voiture)
```

Ce code permet de savoir si une place est libre ou occupée:

```
def si_voiture(self, niv, place):
        if niv == 1:
            if place in self.place_niv1:
                if self.place_niv1[place] != False:
                    print("place non libre")
                    a = str(input("Voulez-vous savoir quelle voiture est dessus ? : "))
                    if a == "oui" or a == "Oui":
                        print(self.place_niv1[place])
                    return False
                else:
                    print("place libre")
                    return True
        if niv == 2:
            if place in self.place_niv2:
                if self.place_niv2[place] != False:
                    print("place non libre")
                    return False
                else:
                    print("place libre")
                    return True
        if niv == 3:
            if place in self.place_niv3:
                if self.place_niv3[place] != False:
                    print("place non libre")
                    return False
                else:
                    print("place libre")
                    return True
        if niv == 4:
            if place in self.place_niv4:
                if self.place_niv4[place] != False:
                    print("place non libre")
                    return False
                else:
                    print("place libre")
                    return True
        if niv == 5:
            if place in self.place_niv5:
                if self.place_niv5[place] != False:
                    print("place non libre")
                    return False
                else:
                    print("place libre")
                    return True
```

Ce code permet de savoir si la place est libre, si elle l'est, la voiture se gare, sinon, cela renvoie que la place est prise :

```
def ajout_place(self, niv, place, voiture):
        if parking.si_voiture(niv, place) == True:
```

```
            print("vous avez pris la place")
            if niv == 1:
                self.place_niv1[place] = voiture.information()
            if niv == 2:
                self.place_niv2[place] = voiture.information()
            if niv == 3:
                self.place_niv3[place] = voiture.information()
            if niv == 4:
                self.place_niv4[place] = voiture.information()
            if niv == 5:
                self.place_niv5[place] = voiture.information()
        else:
            print("vous pouvez pas prendre la place")
```

Ce code permet de voir si une place réservée est occupée ou non :

```
def places_abonnes_occupees(self):
        liste_places = []
        for voiture in self.abonnes:
            if voiture.num_place != None:
                niv = (voiture.num_place + 100) // 100
                place = (voiture.num_place + 100) % 100
                if niv == 1:
                    if place in self.place_niv1:
                        if self.place_niv1[place] != False:
                            if self.place_niv1[place] != voiture.information():
                                liste_places.append(voiture.num_place)
                if niv == 2:
                    if place in self.place_niv2:
                        if self.place_niv2[place] != False:
                            if self.place_niv2[place] != voiture.information():
                                liste_places.append(voiture.num_place)
                if niv == 3:
                    if place in self.place_niv3:
                        if self.place_niv3[place] != False:
                            if self.place_niv3[place] != voiture.information():
                                liste_places.append(voiture.num_place)
                if niv == 4:
                    if place in self.place_niv4:
                        if self.place_niv4[place] != False:
                            if self.place_niv4[place] != voiture.information():
                                liste_places.append(voiture.num_place)
                if niv == 5:
                    if place in self.place_niv5:
                        if self.place_niv5[place] != False:
                            if self.place_niv5[place] != voiture.information():
                                liste_places.append(voiture.num_place)
        return liste_places
```

Ce code permet d'afficher le nombre de places libres :

```
def nombre_places_libres(self):
        nb = 0
        places_abonnes = []
        for voiture in self.abonnes:
            if voiture.num_place != None:
```

```
                places_abonnes.append(voiture.num_place)

        for i in range(80):
            if self.place_niv1[i] == False:
                if (0 + i) not in places_abonnes:
                    nb = nb + 1
            if self.place_niv2[i] == False:
                if (100 + i) not in places_abonnes:
                    nb = nb + 1
            if self.place_niv3[i] == False:
                if (200 + i) not in places_abonnes:
                    nb = nb + 1
            if self.place_niv4[i] == False:
                if (300 + i) not in places_abonnes:
                    nb = nb + 1
            if self.place_niv5[i] == False:
                if (400 + i) not in places_abonnes:
                    nb = nb + 1
        return nb
```

Ce code permet d'afficher les différents niveaux du parking :

```
def afficher_niveau(self, niv):
        if niv == 1:
            print(f"Niveau {niv}:")
            for i in range(80):
                if self.place_niv1[i] == False:
                    print(f"Place {0+i}: libre")
                else:
                    print(f"Place {0+i}: {self.place_niv1[i]}")
        if niv == 2:
            print(f"Niveau {niv}:")
            for i in range(80):
                if self.place_niv2[i] == False:
                    print(f"Place {100+i}: libre")
                else:
                    print(f"Place {100+i}: {self.place_niv2[i]}")
        if niv == 3:
            print(f"Niveau {niv}:")
            for i in range(80):
                if self.place_niv3[i] == False:
                    print(f"Place {200+i}: libre")
                else:
                    print(f"Place {200+i}: {self.place_niv3[i]}")
        if niv == 4:
            print(f"Niveau {niv}:")
            for i in range(80):
                if self.place_niv4[i] == False:
                    print(f"Place {300+i}: libre")
                else:
                    print(f"Place {300+i}: {self.place_niv4[i]}")
        if niv == 5:
            print(f"Niveau {niv}:")
            for i in range(80):
                if self.place_niv5[i] == False:
                    print(f"Place {400+i}: libre")
```

```
                else:
                    print(f"Place {400+i}: {self.place_niv5[i]}")
```

Ce code affiche les chaines de caractères, ici, les places de parking :

```
def __str__(self):
        return (f"{self.place_niv1}"
                f"{self.place_niv2}"
                f"{self.place_niv3}"
                f"{self.place_niv4}"
                f"{self.place_niv5}")
```

Pour la classe Voiture

Ce code créer la classe Voiture et la fonction qui renvoie les informations d'une voiture :

```
class Voiture:

    def __init__(self, immatriculation , marque , proprietaire , abonnement):
        self.im = immatriculation
        self.ma = marque
        self.pr = proprietaire
        self.ab = abonnement
        self.num_place = None

    def information(self):
        return (f"(immatriculation : {self.im})",
                f"(marque : {self.ma})",
                f"(proprietaire : {self.pr})",
                f"(abonnement : {self.ab})")
```

Ces lignes de code permettent d'avoir une information seule, sans avoir à afficher le reste :

```
def get_immatriculation(self):
        return self.im

    def get_marque(self):
        return self.ma

    def get_proprietaire(self):
        return self.pr

    def get_abonnement(self):
        return self.ab
```

Ce code permet d'initialiser le parking et ajoute des voitures pré-faite, avec certaine voitures abonnées :

```
parking = Parking()
parking.place()

voiture1 = Voiture("AA-123-BB", "BMW", "Jean", False)
```

```
voiture2 = Voiture("CC-456-DD", "Renault", "Marie", False)
voiture3 = Voiture("EE-789-FF", "Peugeot", "Paul", False)
voiture4 = Voiture("GG-012-HH", "Audi", "Sophie", False)

print(voiture1.information()) # test les informations d'une voiture

parking.abonnement(voiture1)
voiture1.num_place = 50
parking.abonnement(voiture2)
voiture2.num_place = 150
```

Ce code ajoute les voitures dans le parking et montre le nombres de place encore disponible :

```
parking.ajout_place(1, 50, voiture1)
parking.ajout_place(2, 50, voiture3)
parking.ajout_place(1, 10, voiture4)
parking.ajout_place(1, 10, voiture2)

print(parking.nombre_places_libres())
```

Cette commande permet d'afficher un niveau choisi (ici, le niveau 1) :

```
parking.afficher_niveau(2)
```

Cette commande donne le nombre de places d'abonnées actuellement occupées :

```
print(parking.places_abonnes_occupees())
```

III. Améliorations possibles

Les points que nous aurions pu améliorer sont :

- une représentation en texte du parking
- un code plus clair, mieux structuré et plus compréhensible
- une meilleure gestion du travail (meilleure répartition)