

Documentation et carnet de bord MP3

by Esmard Liz et Vincent Anaël

1. Introduction

1.1. Objectif

Nous devions réaliser une application de gestion d'un parking composé de 5 étages contenant 80 places par étage, numérotées de 1 à 480. On utilisera pour ça, une classe Parking et une classe Voiture.

1.2. Détails du contenue attendu

Un objet voiture doit contenir:

- le numéro d'immatriculation
- sa marque
- le nom du propriétaire
- un booléen indiquant si le propriétaire est abonné. Si oui, un numéro de place lui est attribué

L'objet parking contient:

- la liste des abonnées
- la liste des places.

Des méthodes permettent:

- de récupérer les différentes informations relatives à une voiture (immatriculation, marque, propriétaire, abonnement)
- d'abonner ou d'annuler un abonnement
- d'affecter une voiture à une place
- de savoir si une place donnée est libre, ou quelle voiture l'occupe
- de renvoyer la liste des places d'abonnées occupées par d'autres voitures
- de connaître le nombre de places libres sans compter les places réservées aux abonnées
- de représenter un niveau (en mode texte).

2. Carnet de Bord

La première semaine, nous avons établie point par point tous ce que nous devions faire. Puis, nous avons réaliser la class Voiture, et nous l'avons développer avec plusieurs fonctions que nous avons détaillés dans la documentation de notre code.

La deuxième semaine nous avons poursuivie notre code par la class Parking, qui s'est avérée plus complexe que la class précédente de part ses attentes, sa taille et son codage.

Enfin, la troisième semaine, nous avons finaliser le code et représentés visuellement le parking.

Nous avons travailler à deux sur tous car les classe ne sont pas équitable (classe voiture plus simple que la classe parking).

Puis pour finir, nous avons commencer de rédiger la documentation et le carnet de bord à deux, et par la suite Liz a terminée le document et Anaël a terminé la représentation graphique du parking.

3. Explication du code Python : Gestion d'un Parking

Ce document explique le fonctionnement du fichier MP3.py, qui modélise un système simple de gestion de parking à l'aide de deux classes : Voiture et Parking.

3.1. Classe Voiture

La classe Voiture représente une voiture entrant dans le parking.

Les attributs principaux sont :

- `immat` : l'immatriculation du véhicule
- `marque` : la marque du véhicule
- `proprio` : le nom du propriétaire
- `sub` : indique si la voiture est abonnée (« oui » / « non »)

Les méthodes :

- `__str__()` : retourne une phrase descriptive de la voiture.
- `releve_immatriculation()` : renvoie l'immatriculation.
- `nom_proprio()` : renvoie le nom du propriétaire.
- `constructeur()` : renvoie la marque.
- `abonner()` : renvoie `true` si le véhicule est déclaré comme abonné.

3.2. Classe Parking

Le parking possède :

- `place` : nombre de places par étage
- `etage` : nombre total d'étages
- `reserver_sub` : nombre de places réservées aux abonnés

Lors de l'initialisation :

- `liste_places` : liste des places sous la forme “étage + numéro”, par exemple “003”.
- `occupe` : dictionnaire indiquant si une place est libre (`none`) ou occupée par une voiture.
- `abonnes` : ensemble des immatriculations des véhicules abonnés.

3.2.1. Méthodes importantes

3.2.2. `creation_place()`

Construit la liste de toutes les places du parking sous la forme :

3.2.3. `initialiser_occupation()`

Crée un dictionnaire où chaque place est initialement vide.

3.2.4. `places_dispo()`

Retourne la liste des places non occupées.

3.2.5. `place_libre_occuper(voiture)`

Indique si une voiture occupe déjà une place et, si oui, laquelle.

3.2.6. `attribuer_place(voiture)`

Attribue à la première place libre la voiture donnée. Si aucune place n'est disponible, un message l'indique.

3.2.7. `abonner_voiture(voiture)`

Ajoute la voiture à la liste des abonnés si elle est déclarée comme abonnée.

3.2.8. `desabonner_voiture(voiture)`

Retire une voiture de la liste des abonnés.

3.2.9. `place_reservee_abonne(voiture)`

Indique si la voiture a accès aux places réservées.

3.2.10. nombre_places_libres_sans_compter_place_abonnees()

Compte les places libres sans tenir compte des places réservées aux abonnés.

3.2.11. representation_niveau_du_parking_texte()

Affiche le parking étage par étage sous forme textuelle avec :

- [] pour une place libre
- [X] pour une place occupée

3.3. Partie test du programme

La fin du fichier crée :

- Trois voitures (voiture1, voiture2, voiture3)
- Un petit parking pour les tests (place = 5, etage = 2)

Le script :

1. Affiche le parking vide
2. Attribue des places aux voitures
3. Affiche le parking rempli
4. Vérifie les abonnements
5. Indique quelles places sont occupées par quelle voiture
6. Compte les places libres hors réservées
7. Teste la fonction de désabonnement
8. Vérifie l'accès aux places réservées