Frédéric
LURET

POO
sujet
2025 2026

Version du 13 février 2026

Université de TOURS

Table des matières

I Système de Location de Véhicules

A Objectif

Développer en Python un programme console pour gérer une agence de location de véhicules, en respectant une architecture modulaire multi-fichiers et en appliquant de bonnes pratiques de POO.

Important ⚠

Vous devez rendre votre projet sous la forme d'un lien gitHub.

B Historique des versions

Le dépôt Git fera partie de l'évaluation (qualité de l'historique, granularité des commits, cohérence des messages). Chaque étape clé du développement doit correspondre à, au moins, un commit distinct.

Les commits suivants sont **obligatoires**. Les messages doivent commencer par votre nom. (En abrégé s'il est long, par exemple "Luret - 01-init-projet").

1. `01-init-projet`
   Initialisation du dépôt, création de la structure minimale (`models/`, `data_manager.py`, `ui.py`, `main.py`, `.gitignore`, `README.md`).
2. `02-specification-modeles`
   Ajout dans le `README` d'une description textuelle des classes `Client`, `Vehicule`, `Reservation` (attributs, types, rôle) *sans* encore écrire le code complet.
3. `03-impl-client-vehicule-v1`
   Implémentation des classes `Client` et `Vehicule` (attributs, `__init__`, `__str__` minimal).
4. `04-serialisation-client-vehicule`
   Ajout de `to_dict()` et `from_dict()` pour `Client` et `Vehicule` dans `models/client.py` et `models/vehicule.py`.
5. `05-tarifs-manager-structure`
   Création de `models/tarifs.py` avec la classe `TarifsManager`, le dictionnaire `TARIFS` (même partiel) et la signature de `obtenir_tarif` / `afficher_grille` sans logique complète.
6. `06-tarifs-manager-complet`
   Complétion de la grille tarifaire et implémentation de `obtenir_tarif` et `afficher_grille` pour toutes les cylindrées et forfaits.
7. `07-data-manager-chargement-json`
   Implémentation de `charger_clients()` et `charger_vehicules()` dans `data_manager.py`, lecture des fichiers JSON fournis.

8. `08-ui-menu-minimal`
   Implémentation d'un premier menu texte minimal dans `ui.py` et appel depuis `main.py` (affichage du menu, choix utilisateur, option "Quitter").

9. `09-ui-affichage-clients-vehicules`
   Implémentation de `afficher_clients()` et `afficher_vehicules()` dans `ui.py` en s'appuyant sur les données chargées par `data_manager.py`.

10. `10-modele-reservation-v1`
    Création de la classe `Reservation` dans `models/reservation.py` avec les attributs et le constructeur, sans encore calculer `cout_estime`.

11. `11-reservation-calcul-cout-estime`
    Ajout de la méthode privée `_calculer_cout_estime()` et du calcul automatique de `cout_estime` dans `Reservation`.

12. `12-data-manager-reservations`
    Implémentation dans `data_manager.py` de `charger_reservations()`, `sauvegarder_reservation()`, `generer_id_reservation()`, `filtrer_reservations_par_client()`.

13. `13-ui-creation-reservation`
    Implémentation du flux de création d'une réservation dans `ui.py` (`demander_reservation`, appel à `TarifsManager`, construction d'une `Reservation`, affichage du récapitulatif).

14. `14-ui-affichage-reservations`
    Implémentation de l'affichage de toutes les réservations (`afficher_reservations`) et des réservations d'un client (`afficher_reservations_client`).

15. `15-nettoyage-terminal`
    Implémentation de `nettoyer_terminal()` dans `ui.py` et intégration de son appel au début de chaque action (menu, affichages, création de réservation, etc.).

16. `16-gestion-erreurs-io`
    Gestion d'erreurs de base dans `data_manager.py` (fichiers manquants, JSON invalide) avec messages d'erreur lisibles pour l'utilisateur.

17. `17-gestion-erreurs-domaine` (facultatif)
    Gestion d'erreurs métier : dates invalides ou incohérentes, `id_client` ou `id_vehicule` inexistants, forfait kilométrique invalide, etc.

18. `18-typing-docstrings` (facultatif)
    Ajout d'annotations de type et de docstrings sur les fonctions publiques principales (`data_manager`, `ui`, classes du dossier `models/`).

19. `19-refactorisation-architecture` (facultatif)
    Refactorisation légère pour améliorer la séparation des responsabilités entre `models/`, `data_manager.py`, `ui.py`, `main.py` (sans ajouter de nouvelles fonctionnalités).

20. `20-tests-manuels-et-corrections-finales`
    Série de tests manuels de tous les cas d'usage décrits dans le sujet (affichage des listes, création de réservation, filtrage par client, grille tarifaire), corrections finales avant rendu.

Tout dépôt ne respectant pas cette structure minimale de commits (commits manquants, regroupés ou manifeste-

ment artificiels) pourra être pénalisé.

C Contrainte d'architecture (obligatoire)

Votre projet doit être découpé en plusieurs modules avec un dossier "models/" qui contient :

- `client.py` : classe "Client"
- `vehicule.py` : classe "Vehicule"
- `reservation.py` : classe "Reservation"
- `tarifs.py` : gestionnaire des tarifs ("TarifsManager")
- `__init__.py` : export des classes

En plus du dossier "models/", créez les fichiers suivants à la racine du projet :

- `data_manager.py` : gestion des fichiers (JSON)
- `ui.py` : interface console (menu, affichages, saisies, nettoyage terminal)
- `main.py` : point d'entrée, orchestration générale

Le code doit être organisé pour que chaque fichier ait une responsabilité unique (Single Responsibility Principle). Les classes du dossier "models/" ne doivent pas connaître ni manipuler directement les fichiers ou les interactions console.

D Outils et exécution

- Gestion des paquets et exécution avec UV.
- Aucune dépendance externe n'est requise (librairie standard Python).

E Données et classes à implémenter

- Classe "Client"
  - "id_client" (chaîne)
  - "nom"
  - "prenom"
  - "mail"
  - "telephone"
  - "adresse"
- Classe "Vehicule"
  - "id_vehicule" (chaîne)
  - "marque"
  - "modele"
  - "cylindree" (entier : 4, 5 ou 6)

- "kilometrage_actuel" (nombre)
- "date_mise_en_circulation" (chaîne ou date)

- Classe "Reservation"
  - "id_reservation"
  - "id_client"
  - "id_vehicule"
  - "date_depart" (format "AAAA-MM-JJ")
  - "date_retour" (format "AAAA-MM-JJ")
  - "forfait_km" (100, 200, 300 ou +300)
  - "cout_journalier"
  - "prix_km_supp"
  - "cout_estime"

F Tarifs et calculs

Les tarifs dépendent de la "cylindree" du véhicule (4, 5, 6) et du "forfait_km" (100, 200, 300, +300). Ils peuvent être codés dans "models/tarifs.py" sous forme d'un dictionnaire.

Pour une réservation, calculez :

- Le nombre de jours entre "date_depart" et "date_retour" (minimum 1 jour).
- Le coût estimé : $cout_{estime} = cout_{journalier} \times nb_{jours}$.

Le dépassement kilométrique (et son coût final) peut être traité en bonus.

G Fichiers de données

Stockage JSON (ou CSV) attendu. Par défaut, utilisez JSON :

- "clients.json"
- "vehicules.json"
- "reservations.json"

Les fichiers JSON vous sont fournis avec des données de démonstration. Vous pouvez les modifier ou les compléter.

H Fonctionnalités minimales (obligatoires)

1. Charger la liste des clients et des véhicules depuis des fichiers JSON.
2. Afficher la liste des clients.
3. Afficher la liste des véhicules.
4. Créer une réservation :

- Saisies : "id_client", "id_vehicule", "date_depart", "date_retour", "forfait_km".
- Déterminer "cout_journalier" et "prix_km_supp" en fonction "cylindree" × "forfait_km".
- Calculer "cout_estime" et afficher un récapitulatif.
- Enregistrer la réservation dans "reservations.json".

5. Afficher la liste de toutes les réservations (format condensé : ID, client, véhicule, dates, forfait, coût).
6. Afficher la liste des réservations pour un client donné ("id_client").
7. Nettoyer le terminal avant chaque affichage d'une action (menu, listes, grille, récapitulatif, etc.).

I Architecture attendue (détails)

- "models/client.py" :
  - Classe "Client" avec `to_dict()` et `from_dict()`.
  - `__str__()` pour un affichage formaté.
- "models/vehicule.py" :
  - Classe "Vehicule" avec `to_dict()` et `from_dict()`.
  - `__str__()` pour un affichage formaté.
- "models/tarifs.py" :
  - Classe "TarifsManager" avec :
  - "TARIFS" (dict) : `cylindree: {forfait\_km: (cout\_journalier, prix\_km\_supp)}`.
  - `obtenir_tarif(cylindree, forfait_km)` renvoie `(cout\_journalier, prix\_km\_supp)`.
  - `afficher_grille()` : affichage formaté de la grille.
- "models/reservation.py" :
  - Classe "Reservation" qui calcule `cout_estime` automatiquement.
  - Méthode privée `_calculer_cout_estime()`.
  - `to_dict()` et `__str__()`.
- "data_manager.py" :
  - Lecture/écriture JSON : `charger_clients()`, `charger_vehicules()`, `charger_reservations()`.
  - `sauvegarder_reservation(reservation)`.
  - `generer_id_reservation()` ➔ "R0001", "R0002", ...
  - `filtrer_reservations_par_client(id_client)` : retourne une liste filtrée.
- "ui.py" :
  - `afficher_menu()` et `demander_choix_menu()`.
  - `afficher_clients()`, `afficher_vehicules()`.
  - `demander_reservation()` : orchestre les saisies (client, véhicule, dates, forfait).

- `afficher_recapitulatif(reservation)`.
- `afficher_reservations(reservations)` et `afficher_reservations_client(reservations, id_client)`.
- `demander_id_client()` : saisie d'un ID client.
- `nettoyer_terminal()` : "cls" sous Windows, "clear" sous Unix.

- "main.py" :
  - Orchestration (boucle principale) : menu ➔ choix ➔ action.
  - Appeler `UI.nettoyer_terminal()` au début de chaque action.

J Guide d'implémentation (suggestion de démarche)

1. Créez le package "models/" et implémentez "Client", "Vehicule" et "TarifsManager".
2. Implémentez "Reservation" avec le calcul du coût estimé.
3. Créez "data_manager.py" pour charger/sauvegarder les données et générer les ID.
4. Créez "ui.py" pour l'affichage (menu, listes, récapitulatifs), saisies et nettoyage du terminal.
5. Créez "main.py" comme point d'entrée et reliez toutes les parties.
6. Testez progressivement :
   - Affichez les clients/véhicules.
   - Créez une réservation.
   - Affichez toutes les réservations.
   - Affichez les réservations pour un client précis.

K Bonnes pratiques attendues

- POO propre : encapsulation, responsabilités claires, classes cohérentes.
- Types ("typing") et docstrings : annotations de type et documentation des fonctions.
- Gestion d'erreurs (fichiers absents, JSON invalide, dates au mauvais format).
- Code lisible et structuré (noms explicites, découpage logique).
- Séparation stricte des couches : modèles / persistance / interface / orchestration.

L Annexes sorties console attendues

1 Menu principal

Sortie console

```
============================================================
Chargement des données...
✓ 4 client(s) chargé(s)
✓ 5 véhicule(s) chargé(s)
============================================================

============================================================
SYSTÈME DE LOCATION DE VÉHICULES
============================================================
1. Afficher les clients
2. Afficher les véhicules
3. Créer une réservation
4. Afficher la grille tarifaire
5. Afficher toutes les réservations
6. Afficher les réservations d'un client
7. Quitter
============================================================

Votre choix :
```

menu-principal.txt

2 Affichage des clients : choix 1

Sortie console

```
============================================================
LISTE DES CLIENTS
============================================================
C001 - Jean Dupont (jean.dupont@email.fr)
C002 - Sophie Martin (sophie.martin@email.fr)
C003 - Pierre Dubois (pierre.dubois@email.fr)
C004 - Marie Leroy (marie.leroy@email.fr)


============================================================
SYSTÈME DE LOCATION DE VÉHICULES
============================================================
```

...

```
1. Afficher les clients
2. Afficher les véhicules
3. Créer une réservation
4. Afficher la grille tarifaire
5. Afficher toutes les réservations
6. Afficher les réservations d'un client
7. Quitter
============================================================

Votre choix :
```

affichage-clients.txt

3 Affichage des véhicules : choix 2

Sortie console

```
============================================================
LISTE DES VÉHICULES
============================================================
V001 - Renault Clio (4 cyl., 45000 km)
V002 - Peugeot 308 (5 cyl., 32000 km)
V003 - Citroën C3 (4 cyl., 28000 km)
V004 - BMW Série 3 (6 cyl., 18000 km)
V005 - Volkswagen Golf (5 cyl., 52000 km)


============================================================
SYSTÈME DE LOCATION DE VÉHICULES
============================================================
1. Afficher les clients
2. Afficher les véhicules
3. Créer une réservation
4. Afficher la grille tarifaire
5. Afficher toutes les réservations
6. Afficher les réservations d'un client
7. Quitter
============================================================

Votre choix :
```

affichage-vehicules.txt

4 Création d'une réservation : choix 3

Sortie console

```
============================================================
CRÉER UNE NOUVELLE RÉSERVATION
============================================================

Clients disponibles :
  - C001 - Jean Dupont (jean.dupont@email.fr)
  - C002 - Sophie Martin (sophie.martin@email.fr)
  - C003 - Pierre Dubois (pierre.dubois@email.fr)
  - C004 - Marie Leroy (marie.leroy@email.fr)

ID du client : C004

Véhicules disponibles :
  - V001 - Renault Clio (4 cyl., 45000 km)
  - V002 - Peugeot 308 (5 cyl., 32000 km)
  - V003 - Citroën C3 (4 cyl., 28000 km)
  - V004 - BMW Série 3 (6 cyl., 18000 km)
  - V005 - Volkswagen Golf (5 cyl., 52000 km)

ID du véhicule : V004

Date de départ (AAAA-MM-JJ) : 2026-06-10
Date de retour (AAAA-MM-JJ) : 2026-06-12

Forfaits disponibles : 100, 200, 300, +300
Forfait kilométrique : 200

============================================================
RÉCAPITULATIF DE LA RÉSERVATION
============================================================
Réservation R0003
  Client: C004
  Véhicule: V004
  Du 2026-06-10 au 2026-06-12
  Forfait: 200 km
  Coût journalier: 80.0€
  Prix km supp.: 0.35€/km
  Coût estimé: 160.00€
============================================================

Sauvegarder cette réservation ? (o/n) : o
✓ Réservation sauvegardée dans reservations.json
✓ Réservation enregistrée avec succès !
```

creation-reservation.txt

5 Affichage de la grille tarifaire : choix 4

Sortie console

```
======================================================================
GRILLE TARIFAIRE
======================================================================
Cylindrée    Forfait       Coût/jour  Prix km supp.
----------------------------------------------------------------------
4 cylindres  100             35.00€        0.25€/km
4 cylindres  200             50.00€        0.20€/km
4 cylindres  300             65.00€        0.15€/km
4 cylindres  +300            80.00€        0.10€/km
----------------------------------------------------------------------
5 cylindres  100             45.00€        0.30€/km
5 cylindres  200             60.00€        0.25€/km
5 cylindres  300             75.00€        0.20€/km
5 cylindres  +300            95.00€        0.15€/km
----------------------------------------------------------------------
6 cylindres  100             60.00€        0.40€/km
6 cylindres  200             80.00€        0.35€/km
6 cylindres  300            100.00€        0.30€/km
6 cylindres  +300           120.00€        0.25€/km
----------------------------------------------------------------------
```

grille-tarifaire.txt

6 Affichage de toutes les réservations : choix 5

Sortie console

```
============================================================
LISTE DES RÉSERVATIONS
============================================================
R0001 | Client: C001 | Véhicule: V002 | 2026-01-12 ➔ 2026-01-25 | Forfait: 200 km | Coût estimé:
↪  780.00€
R0002 | Client: C003 | Véhicule: V005 | 2026-05-12 ➔ 2026-05-14 | Forfait: 100 km | Coût estimé: 90.00€
R0003 | Client: C004 | Véhicule: V004 | 2026-06-10 ➔ 2026-06-12 | Forfait: 200 km | Coût estimé:
↪  160.00€


============================================================
SYSTÈME DE LOCATION DE VÉHICULES
============================================================
1. Afficher les clients
2. Afficher les véhicules
```

```
3. Créer une réservation
4. Afficher la grille tarifaire
5. Afficher toutes les réservations
6. Afficher les réservations d'un client
7. Quitter
============================================================

Votre choix :
```

affichage-reservations.txt

7 Affichage des réservations pour un client : choix 6

Sortie console

```
ID du client à rechercher : C001

============================================================
RÉSERVATIONS DU CLIENT C001
============================================================
R0001 | Véhicule: V002 | 2026-01-12 ➔ 2026-01-25 | Forfait: 200 km | Coût estimé: 780.00€
```

affichage-reservations-client.txt