Centres étrangers – 2025 – sujet2

Exercice 1 (6 points)

Cet exercice porte sur la programmation orienté objet, la sécurité des communications.

Une compagnie de taxis est gérée par une application qui a été programmée en langage Python. On s'intéresse à quelques classes programmées dont on donne toutes les propriétés initialisées par le constructeur et les méthodes qui seront utilisées dans cet exercice.

La classe **Chauffeur** Proprietes (Attributs)

Nom	Description	Type	Initialisation
`nom`	Nom du chauffeur.	`str`	Premier paramètre passé au constructeur.
`prenom`	Prénom du chauffeur.	`str`	Deuxième paramètre passé au constructeur.
`neph`	Numéro d'enregistrement préfectoral harmonisé, composé de 12 chiffres. Numéro unique attribué à vie depuis 2013.	`str`	Troisième paramètre passé au constructeur.

La classe **Client** Proprietes (Attributs)

Nom	Description	Type	Initialisation
`nom`	Nom du client.	`str`	Premier paramètre passé au constructeur.
`prenom`	Prénom du client.	`str`	Deuxième paramètre passé au constructeur.
`adresse`	Adresse du client	`str`	Troisième paramètre passé au constructeur.
`nb_personne`	Nombre de personnes accompagnant le client.	`int`	Quatrième paramètre passé au constructeur.

La classe Taxi Proprietes (Attributs)

Nom	Description	Type	Initialisation
`immatriculation`	Immatriculation du véhicule.	`str`	Premier paramètre passé au constructeur.
`type_vehicule`	Le type du véhicule : `'standard', 'monospace', 'minibus'`	`str`	Deuxième paramètre passé au constructeur.
`energie`	Hybride, électrique ou thermique.	`str`	Troisième paramètre passé au constructeur.
`adaptation`	Adapté pour les personnes à mobilité réduite.	`bool`	Quatrième paramètre passé au constructeur.
`libre`	La voiture peut-elle prendre des clients?	`bool`	Initialisé à `True` dans le constructeur.
`chauffeur`	Le chauffeur de la compagnie qui conduit le véhicule.	`Chauffeur`	Initialisé à `None` dans le constructeur.

Methodes

Nom	Description	Paramètres	Type de renvoi
`est_libre`	Renvoie la valeur de la propriété `libre`.	Aucun	`bool`
`modifier_libre`	Modifie la propriété `libre`.	`statut` de type `bool`	Aucun
`choix_chauffeur`	Met à jour la propriété `chauffeur`.	`chauffeur` (de type `Chauffeur`)	Aucun

La classe Course Proprietes (Attributs)

Nom	Description	Type	Initialisation
`client`	Le client qui a commandé la course.	`Client`	Premier paramètre passé au constructeur.
`taxi`	Le véhicule qui prendra en charge le client et les accompagnants.	`Taxi`	Deuxième paramètre passé au constructeur.
`depart`	Adresse de départ au format : `'42 rue de l'informatique, Octetsur-Mer, France'`.	`str`	Troisième paramètre passé au constructeur.
`destination`	Adresse de destination au format : `'42 rue de l'Informatique, Octetsur-Mer, France'`.	`str`	Quatrième paramètre passé au constructeur.

`date_depart`	Date de départ au format : `datetime(année, mois, jour, heure, minutes)`.	`datetime`	Initialisée à `None` dans le constructeur.
`date_arrivee`	Date d'arrivée au format : `datetime(année, mois, jour, heure, minutes)`.	`datetime`	Initialisée à `None` dans le constructeur.

METHODES

Nom	Description	Paramètres	Type de renvoie
`distance`	Renvoie la distance du trajet en km.	Aucun	`float`
`temps`	Renvoie le temps du trajet en minutes.	Aucun	`float`

On dispose des variables suivantes.

- Un dictionnaire `vehicules` qui a pour clés le type du véhicule et pour valeurs un dictionnaire regroupant les informations suivantes.
  - `capacite`: le nombre de passagers pris en charge en plus du conducteur.
  - `prix_km` : le prix du km appliqué par la compagnie en euros.
  - `prise_en_charge` : le coup fixe de la prise en charge en euros.
  - `tarif_horaire` : le tarif horaire appliqué en euros.

```
vehicules = {
    'standard': {'capacite': 4, 'prix_km': 1.10,
'prise_en_charge': 3, 'tarif_horaire': 38},
    'monospace': {'capacite': 8, 'prix_km': 1.80,
'prise_en_charge': 9, 'tarif_horaire': 60},
    'minibus': {'capacite': 19, 'prix_km': 3.00,
'prise_en_charge': 50, 'tarif_horaire': 100}
    }
```

- Une liste `energie` contenant les éléments d'information sur l'énergie utilisée par les véhicules de la flotte de taxis de la compagnie.

```
energie = ['hybride', 'électrique', 'thermique']
```

PARTIE A

Chaque classe est définie dans un script Python qui porte le nom de la classe, `taxi.py` pour la classe `Taxi`, `client.py` pour la classe `Client`, etc., et les variables `vehicules` et `energie` sont déclarées dans un script Python nommé `donnees.py`. L'application est programmée dans un script Python nommé `application.py`.

Tous les fichiers sont dans le même répertoire.

1. Donner les instructions à ajouter au script `application.py` pour que l'on puisse y utiliser les variables `vehicules` et `energie` du script `donnees.py` et les classes `Chauffeur`, `Taxi`, `Client` et `Course`.

2. Écrire une commande Python permettant d'instancier le chauffeur de taxi de prénom John et de nom Doe, dont le numéro neph est 140159320012.

Le chauffeur John Doe conduit le véhicule de type standard à énergie électrique d'immatriculation HG-818-AV, qui n'est pas adapté pour les personnes à mobilités réduites.

3. Écrire les instructions qui permettent de créer le véhicule d'immatriculation HG-818-AV et de l'attribuer au chauffeur John Doe.

La centrale de la compagnie de taxi envoie une demande au chauffeur John Doe pour qu'il prenne en charge la cliente Jeanne Doe qui est accompagnée de ses deux enfants.

Elle désire, de son domicile au 6 rue des ordinateurs à Paris, se rendre au 211 avenue Jean Jaurès à Paris (Le parc de la Villette).

4. Écrire une assertion, qui placée au début du constructeur de la classe `Course`, permet d'interdire la création de la course si le taxi n'est pas libre.

5. Ajouter une instruction à la classe `Course` afin que le taxi ne soit plus marqué comme libre lors de la création d'une course.

6. Écrire la ou les instructions qui permettent de créer la course de la cliente Jeanne Doe.

À l'aide de la bibliothèque Python `datetime` on peut calculer la différence de temps en secondes entre deux dates à l'aide de la fonction du même nom `datetime`.

```
>>> from datetime import datetime
>>> depart = datetime(2025, 7, 12, 9, 12)
>>> arrivee = datetime(2025, 7, 12, 17, 12)
>>> (arrivee - depart).total_seconds()
28800.0
```

C'est-à-dire qu'entre le 12 juillet 2025 à 9h12 et le 12 juillet 2025 à 17h12 il s'est écoulé 28800 secondes.

7. Écrire un code Python pour la méthode `temps` de la classe `Course` qui renvoie le temps du trajet en minutes.

On veut ajouter la méthode `tarif` à la classe `Course` qui permet de renvoyer le montant à payer en euros par le client pour la course réalisée. Le chauffeur John Doe va réaliser un trajet de 9,1 km en 19 minutes pour amener sa cliente Jeanne Doe à destination. Pour un véhicule standard la prise en charge est de 3 euros, le prix au km est de 1,10 euros et le tarif horaire est de 38 euros.

Le calcul du coût en euros de la course est le suivant :

$$3 + 1{,}10 \times 9{,}1 + 38 \times \frac{19}{60} \approx 25{,}04$$

8. Recopier et compléter le code Python pour la méthode `tarif` de la classe `Course` suivant.

```
    def tarif(self):
        type_vehicule = ...
        donnees_tarifaire = vehicules[type_vehicule]
        prise_en_charge = ...
        tarif_h = ...
        prix_km = ...
        tarif = ...
        tarif = tarif + prix_km*...
        tarif = tarif + tarif_h*.../60
        return tarif
```

PARTIE B

L'application permet de gérer les échanges entre la centrale de la compagnie de taxi et les différents chauffeurs. Le système permet de communiquer de façon sécurisée avec les chauffeurs.

9. Expliquer la différence de fonctionnement entre les chiffrements symétrique et asymétrique.

Pour initier une communication avec le chauffeur John Doe, l'application va générer aléatoirement une clé de session symétrique avec laquelle elle va chiffrer les messages à transmettre. Le chauffeur doit posséder cette clé pour déchiffrer les messages. L'application transmet la clé au chauffeur sans traitement particulier, avant d'envoyer les messages chiffrés.

10. Expliquer le problème de sécurité qui peut intervenir en transmettant la clé de session directement au chauffeur.

Il est établi que le chiffrement symétrique est très rapide et efficace pour chiffrer de grandes quantités de données. La compagnie de taxi souhaite que les communications soient sécurisées, sans pour autant ralentir la transmission des échanges entre la centrale et les chauffeurs.

11. Proposer une stratégie qui permettrait de transmettre cette clé de session symétrique de façon sécurisée en détaillant les étapes.