Ministère de l'Enseignement Supérieur
et de la Recherche Scientifique
Université de Sfax
Institut Préparatoire aux Etudes
d'Ingénieurs de Sfax

IPEIS

وزارة التعليم العالي والبحث العلمي
جامعة صفاقس
المعهد التحضيري للدراسات الهندسية
بصفاقس

Année universitaire : 2022-2023

Examen semestre 2

Epreuve d'informatique

Classe : MP2, PC2 et PT2 **Nombre de pages** : 4

Date : Mai 2023 **Durée** : 2h

Exercice : Agenda électronique

L'objectif de cet exercice est de définir des méthodes d'une classe **RDV** permettant de gérer l'ensemble des rendez-vous (instances de cette classe) formant l'agenda d'une personne.

La classe **RDV** permet d'instancier des objets représentant des rendez-vous. Le constructeur de cette classe initialisera les attributs d'instance suivants :

- la date « ***date*** », représentée par une chaîne de caractères (sous forme `'aaaa-mm-jj'`)
- l'heure « ***heure*** », représentée par un entier (on suppose que tous les rendez-vous ont lieu à une heure pile, par exemple 10h sera représenté par l'entier 10)
- le nom « ***nom*** » désignant le nom de la personne avec qui on a rendez-vous, représenté par une chaîne de caractères.

Exemple :

```
Agenda = {  RDV('2023-11-17', 10, 'Elie'),
            RDV('2023-11-17', 11, 'Nour'),
            RDV('2023-11-18', 10, 'Paul'),
            RDV('2023-11-19', 14, 'Paul'),
            RDV('2023-11-19', 10, 'Sacha') }
```

1. Définir une classe **RDV** et de sa méthode **__init__** qui initialise les attributs d'objet avec des valeurs données en argument à la fonction **__init__**.

2. Définir la méthode **__repr__** qui retourne une chaine de caractère contenant les différents champs. Par exemple, `>>> RDV('2023-11-17', 10, 'Elie')` affiche :

```
2023-11-17 10h Elie
```

3. Définir la méthode **rdv_personne** qui, étant donnés un ensemble de rendez-vous **E** et une chaîne **p** représentant le nom d'une personne, renvoie l'ensemble des rendez-vous avec cette personne.

Exemples :

```
>>> RDV.rdv_personne(Agenda , 'Laeticia')
{}
>>> RDV.rdv_personne(Agenda , 'Paul')
{2023-11-19 14h Paul, 2023-11-18 10h Paul}
```

4. Définir la méthode **rdv_restants** qui, étant donné un ensemble de rendez-vous **E**, une date **d** et une heure **h**, renvoie l'ensemble des rendez-vous de **E** qui ont lieu à la date **d** et après l'heure **h**.

   Exemple :

```
>>> RDV.rdv_restants(Agenda, '17/11/2023', 9)
{2023-11-17 11h Nour, 2023-11-17 10h Elie}
```

5. Définir la méthode **nb_rdv** qui, étant donné un ensemble de rendez-vous **E**, renvoie un dictionnaire associant à chaque date de **E** le nombre de rendez-vous ayant lieu à cette date.

   Exemples :

```
>>> RDV.nb_rdv(set())
{}
>>> RDV.nb_rdv(Agenda)
{'2023-11-17': 2, '2023-11-18': 1, '2023-11-19': 2}
```

6. Définir la méthode **dict_date** qui, étant donné un ensemble de rendez-vous **E**, renvoie un dictionnaire dont les clés sont les dates des rendez-vous de **E**. La valeur correspondante à une date **d** est l'ensemble de couples **(h,p)** pour chaque rendez-vous **(d, h, p)** de **E**.

   Exemples :

```
>>> RDV.dict_date(set())
{}
>>> RDV.dict_date(Agenda)
{'2023-11-17': {(11, 'Nour'), (10, 'Elie')}, '2023-11-18':
{(10, 'Paul')}, '2023-11-19': {(10, 'Sacha'), (14, 'Paul')}}
```

7. Définir la méthode **conflit_rdv** qui, étant donnés deux rendez-vous **r** et **s**, renvoie **True** si les deux rendez-vous ont lieu à la même date et à la même heure, et **False** sinon.

   Exemples :

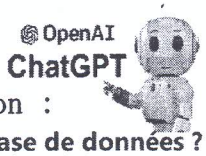
```
>>> RDV.conflit_rdv(  RDV('2023-12-18', 10, 'Sacha'),
                      RDV('2023-12-18', 10, 'Pierre') )
True
>>> RDV.conflit_rdv(  RDV('2023-12-18', 10, 'Sacha'),
                      RDV('2023-12-18', 11, 'Pierre') )
False
```

8. Définir la méthode **conflit_agenda** qui, étant donné un ensemble de rendez-vous **E**, renvoie **True** si **E** contient deux rendez-vous qui sont en conflit, et **False** sinon.

   Exemples :

```
>>> RDV.conflit_agenda(Agenda)
False
>>> RDV.conflit_agenda({RDV('2023-12-18', 10, 'Sacha'),
                        RDV('2023-12-18', 10, 'Pierre') ,
                        RDV('2023-12-18', 11, 'Nour')})
True
```

Problème : Classes et BD



Le problème suivant a été initié par ChatGPT suite à la question :
Peux-tu me formuler un problème groupant la notion de classe et la notion de base de données ?

Une entreprise de vente en ligne vend des produits à travers une plateforme sur internet. Dans la base de données de l'entreprise on stocke les informations sur les produits, les commandes et les clients. Le problème consiste à créer une application Python qui permettra de définir les classes « **Produit**, **Commande**, **Client** et **BaseDonnees** » nécessaires à la création de l'application permettant aux clients de passer des commandes, aux vendeurs de gérer les stocks de produits et aux administrateurs de gérer les comptes clients.

Le schéma relationnel de la base de données est le suivant :

*** Produit (id, nom, description, prix, quantite)**
- id : identifiant du produit de type entier clé primaire,
- nom : nom du produit de type texte,
- description : description du produit de type texte,
- prix : prix du produit de type réel,
- quantite : quantité du produit en stock de type entier.

*** Clients (id, nom, adresse, telephone)**
- id : identifiant du client de type entier clé primaire,
- nom : nom du client de type texte,
- adresse : adresse du client de type texte,
- telephone : numéro du téléphone de type entier,

*** Commandes (id, date, #id_Client, #id_Produit, prix_paye)**
- id : référence de la commande de type entier clé primaire,
- date : date de la commande de type date,
- id_Client : identifiant du client de type entier, clé étrangère qui fait référence à id de la table Client,
- id_Produit : identifiant du produit de type entier, clé étrangère qui fait référence à id de la table Produit,
- prix_paye : prix payé de type réel

La classe **BaseDonnees** contient les méthodes **creer_tables, ajouter_produit, ajouter_client, ajouter_commande, supprimer_produit, supprimer_client, supprimer_commande, modifier_produit, modifier_client.**

Ecrire le code python permettant de :

1. Définir la classe **Produit**. Les objets de cette classe doivent contenir les attributs id, nom, description, prix et quantité.
2. Définir la classe **Commande**. Les objets de cette classe doivent contenir les attributs id, date, client, produit et prix_paye.
3. Définir la classe **Client**. Les objets de cette classe doivent contenir les attributs id, nom, adresse et tetephone.

La création des tables est réalisée par la méthode **creer_tables**.

L'instanciation de la classe **BaseDonnees** avec un nom de fichier comme paramètre permet de :

- créer un attribut connection utilisé comme nom logique de connection à la base,
- effectuer un appel de la méthode creer_tables de la classe BaseDonnees.

4. Définir la classe **BaseDonnees**.
5. Ecrire la méthode **creer_tables** qui permet de définir le curseur et créer la table **Commandes** selon les commandes SQLite. (les tables **Produits** et **Clients** ne sont pas demandées)
6. Définir les méthodes : **ajouter_produit(*self, produit*), supprimer_produit(*self, id_produit*) , modifier_produit(*self, id_produit*).** L'argument ***produit*** est un objet de la classe **Produit** à ajouter, et ***id_produit*** est l'identifiant du produit à supprimer ou modifier.
7. **a.** Définir une méthode **requete(*self,req*)** permettant d'une part d'exécuter la requête SQL ***req*** et d'autre part d'afficher le résultat de la requête.

   b. En supposant que la commande **db = BaseDonnees("oo.db")** a été exécuté et que différents enregistrements ont été effectués dans la base, écrire l'appel permettant de donner les produits non disponibles.

Donner les commandes SQL permettant de :

8. Calculer le montant total des commandes passées par chaque client.
9. Donner le nom des produits commandés par "monsieur X"
10. Donner le nom des clients n'ayant pas effectué des commandes en 2023.
11. Donner le nombre des clients ayant effectué des commandes dont le montant total dépasse 1000 dinars.
12. Donner les noms des clients ayant effectué au moins deux commandes supérieures chacune à 400 dinars.