

13/10/2024

COMPTE RENDU PYTHON TP-2

réalisé par

El Ouardi Mohamed

Encadré par :

Prof.Hain



• ACTIVITE 10

• Étapes réalisées :

- Création d'un projet Python nommé "tp_DB" avec un fichier "test1.py".
- Connexion à une base de données SQLite nommée "gestion.db".
- Création d'une table "person" avec deux colonnes : "code" (entier) et "name" (texte).
- Insertion de données dans la table :
 - (10, "Karim")
 - (11, "Amal")
 - (12, "Reda")
- Affichage du contenu de la table.
- Suppression de la ligne avec le code 10 (Karim).
- Modification du nom "Reda" en "Radi" pour le code 12.
- Affichage du contenu final de la table.

Code:

```
import sqlite3
db = sqlite3.connect("gestion.db")
db.execute("create table if not exists person (code integer, name text)")
db.execute("insert into person (code, name) values (?, ?)", (10, "Karim"))
db.execute("insert into person (code, name) values (?, ?)", (11, "Amal"))
db.execute("insert into person (code, name) values (?, ?)", (12, "Reda"))
db.row_factory = sqlite3.Row
cursor = db.execute("select * from person")
for row in cursor:
    print(row["code"], row["name"])
print("Supprimer la ligne de Karim")
db.execute("delete from person where code=10")
cursor = db.execute("select * from person")
for row in cursor:
    print(row["code"], row["name"])
```

```
print("Modifier le nom de Reda a Radi")
db.execute("update person set name='Radi' where code=12")
cursor = db.execute("select * from person")
for row in cursor:
    print(row["code"], row["name"])
db.close()
```

• Observations :

- La base de données "gestion.db" a été créée avec succès.
- La table "person" a été créée et les données ont été insérées correctement.
- Les opérations de suppression et de mise à jour ont fonctionné comme prévu.



Name	Type	Schema
Tables (1)		
person		CREATE TABLE person (code integer, name text)
code	integer	"code" integer
name	text	"name" text
Indices (0)		

• Résultat final :

- La table "person" contient maintenant deux entrées :
- (11, "Amal")
- (12, "Radi")

```
C:\Users\MED\AppData\Local\Programs\Python\Python312\python.exe
10 Karim
11 Amal
12 Reda
Supprimer la ligne de Karim
11 Amal
12 Reda
Modifier le nom de Reda a Radi
11 Amal
12 Radi
```

• ACTIVITE 2

• Étapes réalisées :

- Création d'une base de données nommée "MY_BD.db".
- Création de deux tables :
 - "formation" (id, titre)
 - "etudiant" (code, nom, email)
- Insertion de données initiales dans les deux tables.
- Implémentation de fonctions CRUD :
 - ins(v1, v2) : Insérer une nouvelle formation
 - modifier(id, titre_modif) : Modifier le titre d'une formation
 - supprimer(id) : Supprimer une formation
 - afficher(tab) : Afficher le contenu d'une table
- Test des fonctions implémentées :
 - Insertion de nouvelles formations
 - Modification d'une formation existante
 - Suppression d'une formation

Code:

```
import sqlite3
db = sqlite3.connect("MY_BD.db")
db.execute("create table if not exists formation(id integer, titre text)")
db.execute("create table if not exists etudiant(code integer, nom text, email text)")
db.execute("insert into formation values(?,?)", (1, "python"))
db.execute("insert into formation values(?,?)", (2, "html"))
db.execute("insert into etudiant values(?,?,?)", (1, "youssef", "youssef@gmail.com"))
db.execute("insert into etudiant values(?,?,?)", (2, "sara", "sara@gmail.com"))
db.commit()
def ins(v1, v2):
    db.execute("insert into formation values(?,?)", (v1, v2))
    db.commit()
```

```
def modifier(id, titre_modif):
    db.execute("update formation set titre=? where id=?", (titre_modif, id))
    db.commit()
def supprimer(id): # Correction de l'orthographe ici
    db.execute("delete from formation where id=?", (id,))
    db.commit()
def afficher(tab):
    c = db.execute(f"select * from {tab}")
    for r in c:
        print(r)
print("Affichage du contenu de chaque table Avant Modification")
afficher("formation")
afficher("etudiant")
print("Affichage du contenu de chaque table Après Modification")
ins(3, "JAVA")
ins(4, "C")
ins(5, "PYTHON")
modifier(1, "C++")
supprimer(2)

print("Contenu après les modifications :)")
afficher("formation")

db.close()
```

- **Observations :**

- Les tables "formation" et "etudiant" ont été créées avec succès.
- Les données initiales ont été insérées correctement.
- Les fonctions CRUD fonctionnent comme prévu :
- Nouvelles formations ajoutées (JAVA, C, PYTHON)
- Modification du titre de la formation avec id=1 de "python" à "C++"
- Suppression de la formation avec id=2

- **Résultat final :**

- La table "formation" a été modifiée et contient maintenant :
- (1, "C++")
- (3, "JAVA")
- (4, "C")
- (5, "PYTHON")

Database Structure Browse Data Edit Pragmas Execute SQL			
Table: etudiant			
	code	nom	email
	Filter	Filter	Filter
1	1	youssef	youssef@gmail.com
2	2	sara	sara@gmail.com

Table: formation		
	id	titre
	Fil...	Filter
1	1	C++
2	3	JAVA
3	4	C
4	5	PYTHON

• ACTIVITE 7

• Étapes réalisées :

- Création d'un projet nommé "Projet2_3" avec un package "gestion".
- Implémentation de la classe de base Person avec ses attributs et méthodes.
- Création des classes dérivées Student et Employe, héritant de Person.
- Implémentation des constructeurs, getters, setters et méthodes `__str__` pour chaque classe.
- Création d'une classe MyClass pour tester les classes implémentées.

• Observations :

- Les classes Student et Employe héritent correctement des attributs et méthodes de la classe Person.
- La surcharge de la méthode `__str__` dans les classes dérivées permet d'afficher les informations spécifiques à chaque type d'objet.
- L'utilisation de `super().__init__()` ou `Person.__init__()` dans les constructeurs des classes dérivées assure l'initialisation correcte des attributs de la classe de base.

• Résultat final :

- Lorsque `MyClass()` est instancié, il crée et affiche :
- Deux objets Student avec leurs attributs spécifiques.
- Un objet Employe avec ses attributs spécifiques.
- L'affichage montre clairement la différence entre les objets et l'inclusion des attributs hérités et spécifiques.

```
C:\Users\MED\AppData\Local\Programs\Python\Python312\python.exe D:\Apy\Projet2_4\main.py
Student[1]
id: 1 | name: MED | city: casa | formation:Engineering | level:Master
Student[2]
id: 2 | name: Hamza | city: fes | formation:Science | level:Bachelor
Employe[1]
id: 1 | name: Sara | city: rabat | job:Developer | sal:30000

Process finished with exit code 0
```

• ACTIVITE 8

• Étapes réalisées :

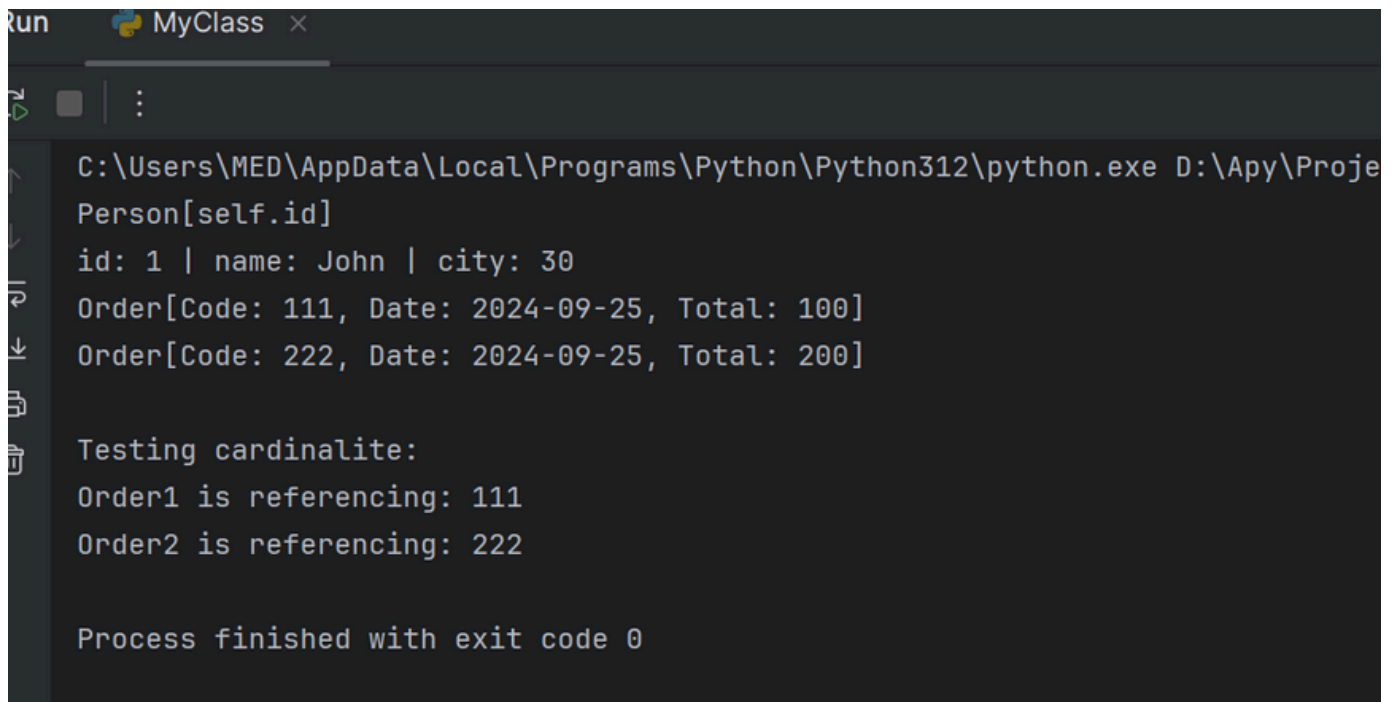
- Création d'un projet nommé "Projet2_4" avec un package "com".
- Implémentation de la classe Person avec ses attributs et méthodes.
- Implémentation de la classe Order avec ses attributs et méthodes.
- Création d'une classe MyClass pour tester les classes implémentées.
- Test des cardinalités entre les objets Person et Order.

• Observations :

- Les classes Person et Order sont correctement implémentées avec leurs attributs, constructeurs, getters, setters et méthodes `__str__`.
- Dans la classe MyClass, un objet Person et deux objets Order sont créés.
- Les objets Order ne sont pas directement liés à l'objet Person dans l'implémentation actuelle.
- La méthode `display()` de MyClass affiche les informations de chaque objet séparément.

- **Résultat final :**

- Lorsque MyClass() est instancié et que display() est appelé, on obtient :
- L'affichage des informations de l'objet Person.
- L'affichage des informations des deux objets Order.
- Un test de cardinalité qui montre les codes des deux commandes.



```
C:\Users\MED\AppData\Local\Programs\Python\Python312\python.exe D:\Apy\Proje
Person[self.id]
id: 1 | name: John | city: 30
Order[Code: 111, Date: 2024-09-25, Total: 100]
Order[Code: 222, Date: 2024-09-25, Total: 200]

Testing cardinalite:
Order1 is referencing: 111
Order2 is referencing: 222

Process finished with exit code 0
```