Python

Licence 3

Séance 4

Souleymane BAH

bsouleymane@gmail.com

Plan de la séance

- Révision cours passé
- Contrôle du devoir de maison
- SQL avec Python (suite et fin)
- Fonctions
- Projet final
- Discussion
- Ressources pour aller plus loin

Révision cours passé

1. Fichiers et persistance ?
2. Ouvrir fichier ?
3. Modes ouverture ?
   - 'w' vs 'a'
4. De quoi Python à besoin pour interagir avec le SGBD MySQL ?

Contrôle du devoir de maison

1. Ecrire un code qui demande un nom d’un fichier existant
2. Puis ouvre le fichier
3. Puis affiche ses lignes

SQL avec Python 1/13

- **WampServer**
  - Windows Apache MySQL PHP
  - Télécharger :
    - https://www.wampserver.com/#download-wrapper
  - Installer
    - Choisir le navigateur et l'éditeur de texte par défaut
  - Démarrer WampServer

SQL avec Python 2/13

- **Bonus : PORT OCCUPÉ PAR PROCESSUS + ARRÊT PROCESSUS**
  - Ouvrir CMD **en tant qu'administrateur**
  - > netstat -bano | more
  - Rechercher le PID du processus écoutant sur le port 3306
  - Ouvrir le Gestionnaire de tâches
  - Ajouter PID au Gestionnaire de tâches
  - Clic droit → PID → Classer par PID → Tuer processus

SQL avec Python 3/13

- **Connecteur Python**
  - Télécharger puis Installer Microsoft Visual C++ 2015
    - https://www.microsoft.com/fr-FR/download/details.aspx?id=52685
  - Télécharger puis Installer le connecteur
    - https://dev.mysql.com/downloads/connector/python/

SQL avec Python 4/13

- Connexion : connexion.py

```
import mysql.connector


ma_base_de_donnee = mysql.connector.connect(
    host = "localhost",
    user = "root", # en production autres conptes
    password = "" # en production changer de mot de passe
)

print(ma_base_de_donnee)
```

SQL avec Python 5/13

- Création base de données : create.py

```
# Connexion d'abord

# Suite
mon_curseur = ma_base_de_donnee.cursor()

mon_curseur.execute("CREATE DATABASE ma_base")
```

SQL avec Python 6/13

- Voir bases de données existantes : view_db.py
  - **phpMyAdmin : http://localhost/phpmyadmin/**
    - Utilisateur : root
    - Mot de passe : VIDE (pas de mot de passe)
  - **Python :**

```
# Connexion d'abord

# Suite
mon_curseur = ma_base_de_donnee.cursor()

mon_curseur.execute("SHOW DATABASES")

for base in mon_curseur:
    print(base)
```

SQL avec Python 7/13

- Création table : create_table.py

```
import mysql.connector

ma_base_de_donnee = mysql.connector.connect(
    host = "localhost",
    user = "root", # en production autres conptes
    password = "", # en production changer de mot de passe
    database = "ma_base" # une base en particulier
)

mon_curseur = ma_base_de_donnee.cursor()

mon_curseur.execute("CREATE TABLE utilisateur(login VARCHAR(50), nom VARCHAR(100), prenom VARCHAR(150))")
```

SQL avec Python 8/13

- Création table : view_tables.py
  - **phpMyAdmin : http://localhost/phpmyadmin/**
    - Utilisateur : root
    - Mot de passe : VIDE (pas de mot de passe)
  - **Python :**

```
# Connexion à une base d'abord

# Suite
mon_curseur = ma_base_de_donnee.cursor()

mon_curseur.execute("SHOW TABLES")

for table in mon_curseur:
    print(table)
```

SQL avec Python 9/13

- Insérer données : insert.py

```
# Connexion à une base d'abord

# Suite
mon_curseur = ma_base_de_donnee.cursor()

requete = "INSERT INTO utilisateur (login, nom, prenom) VALUES (%s, %s, %s)"
valeurs = [# liste de tuples
    ('sbah', 'BAH', 'Souleymane'),
    ('bsuzanne', 'BAGUIDI', 'Suzanne'),
    ('mtoto', 'MAMA', 'Toto')
]

mon_curseur.executemany(requete, valeurs)

ma_base_de_donnee.commit()

print(mon_curseur.rowcount, " lignes insérées")
```

SQL avec Python 10/13

- Voir données : view_data.py
  - **phpMyAdmin : http://localhost/phpmyadmin/**
    - Utilisateur : root
    - Mot de passe : VIDE (pas de mot de passe)
  - **Python :**

```
# Connexion à une base d'abord

# Suite
mon_curseur = ma_base_de_donnee.cursor()

mon_curseur.execute("SELECT * FROM utilisateur")

for ligne in mon_curseur:
    print(ligne)
```

SQL avec Python 11/13

- Filtrer les données : filter.py
  - **Python :**

```
# Connexion à une base d'abord

# Suite
mon_curseur = ma_base_de_donnee.cursor()

mon_curseur.execute("SELECT * FROM utilisateur WHERE login = 'sbah'")

for ligne in mon_curseur:
    print(ligne)
```

SQL avec Python 12/13

- Mise à jour de données : update.py

```
# Connexion à une base d'abord

# # Suite
mon_curseur = ma_base_de_donnee.cursor()

mon_curseur.execute("UPDATE utilisateur SET prenom = 'alain' WHERE login = 'sbah'")

ma_base_de_donnee.commit()

print(mon_curseur.rowcount, " lignes mises à jour")
```

- **Vérifier (phpMyAdmin et Python)**

SQL avec Python 13/13

- Filtrer les données : delete.py

```
# Connexion à une base d'abord

# # Suite
mon_curseur = ma_base_de_donnee.cursor()

mon_curseur.execute("DELETE FROM utilisateur WHERE login = 'sbah'")

ma_base_de_donnee.commit()

print(mon_curseur.rowcount, " lignes supprimées")
```

- **Vérifier (phpMyAdmin et Python)**

Fonctions 1/3

- Répétitions dans code
- Fonctions : exécuter le même code sur différentes valeurs
- Définition function
  - def *nom_function(param_1, param_2, …, param_n)*:

    # code

    return *valeur_de_retour*
- Appel function : les arguments effectifs
  - *ma_variable = ma_function(arg_1, arg_2, …, arg_n)*

Fonctions 2/3

TP 7 : ma calculatrice modulaire

- Ecrire les fonctions **somme(), diff(), produit(), division(), modulo(), plus_grand()** qui prennent deux nombres comme paramètres et qui effectuent respectivement leurs somme, difference, produit, division, modulo et comparaison (renvoie le plus grand)
- Ecrire du code pour tester votre programme :
  - entrez deux nombres **(réels)**
  - effectuez l'ensemble des opérations

Fonctions 3/3

TP 8 : mon traitement de texte modulaire

- Ecrire la fonction **ouvrir_fichier()** qui prend en paramètre le nom d'un fichier et le mode d'ouverture et renvoie l'identifiant de fichier
- Ecrire du code pour tester votre programme
- Ecrire la fonction **ecrire_ligne_fichier()** qui prend deux paramètres : le nom d'un fichier et une ligne de texte. La fonction **ajoute** ensuite la ligne dans le fichier (**en utilisant ouvrir_fichier()**)
- Ecrire du code pour tester votre programme
- Ecrire la fonction **lire_fichier()** qui prend en paramètre le nom d'un fichier et qui affiche son contenu (**en utilisant ouvrir_fichier()**)
- Ecrire du code pour tester votre programme

Projet final : Application de gestion des notes 1/4

- Vous avez été retenu pour réaliser l'application de gestion des notes de votre école. Le cahier de charges parle des fonctionnalités ci-après :
  - Gestion des étudiants : ajout, modification, recherche et suppression. Il faut noter que chaque étudiant a un matricule unique
  - Gestion des matières : ajout, modification, recherche et suppression. Chaque matière a un coefficient
  - Gestion des notes : ajout, modification, recherche et suppression. Il existe trois évaluations : devoir, examen et projet. L'examen a un coefficient de 2 par rapport aux autres : moyenne = (devoir + examen*2 + projet) / 4
  - Génération de relevés PDF : l'application doit pouvoir générer le relevé de notes de chaque élève **en version PDF** (1 fichier PDF par élève)

Projet final : Application de gestion des notes 2/4

- Vous avez été retenu pour réaliser l'application de gestion des notes de votre école. Le cahier de charges parle des contraintes techniques suivantes :
  - **Utilisation de fonctions**
  - **Stockage des données dans une base de données MySQL**
  - **Code lisible, élégant et maintenable**

Projet final : Application de gestion des notes 3/4

- **Travail à rendre**
  - Rapport écrit contenant les noms des membres du groupe (**4 membres max**) et présentant la démarche utilisée.
  - Joindre le code et le schéma de la base SQL (code Python de création de la base et des tables)
  - **ÇA DOIT S'EXÉCUTER TOUT SEUL**
- **Envoyer à :**
  - bsouleymane@gmail.com
  - **Objet du mail : PROJET PYTHON L3 UATM**
- **Date limite** : le dimanche **25 décembre 2022 à 23h59**

Projet final : Application de gestion des notes 4/4

- **Ressources**
  - Générer un PDF depuis Python : utiliser la librairie **ReportLab**
    - **Installation** : en ligne de commande (CMD) **en mode administrateur**
      - **python -m pip install reportlab**
    - **Utilisation** : https://www.blog.pythonlibrary.org/2021/09/28/python-101-how-to-generate-a-pdf/

```
import os
from reportlab.pdfgen import canvas

chemin_absolu = os.path.dirname(__file__)
os.chdir(chemin_absolu)

my_canvas = canvas.Canvas("hello.pdf")
my_canvas.drawString(100, 750, "Welcome to Reportlab!")
my_canvas.save()
```

Projet final : Application de gestion des notes

QUESTIONS ?

Discussion

- Examen final
  - Reprendre et comprendre **totalement** le cours
  - Reprendre **tous les codes** (devoirs de maison compris)
  - **Sinon ~= 0**

Ressources pour aller plus loin

- **Graphical User Interface :** https://realpython.com/python-gui-tkinter/

M E R C I