Numérique et Sciences Informatiques¶

Thème : Bases de données

Projet 1 = Gestion d'une BDD et interrogation de celle-ci en ligne.

I) Implémentation, modification et interrogation d'une BDD grâce à la bibliothèque « sqlite3 » de Python.

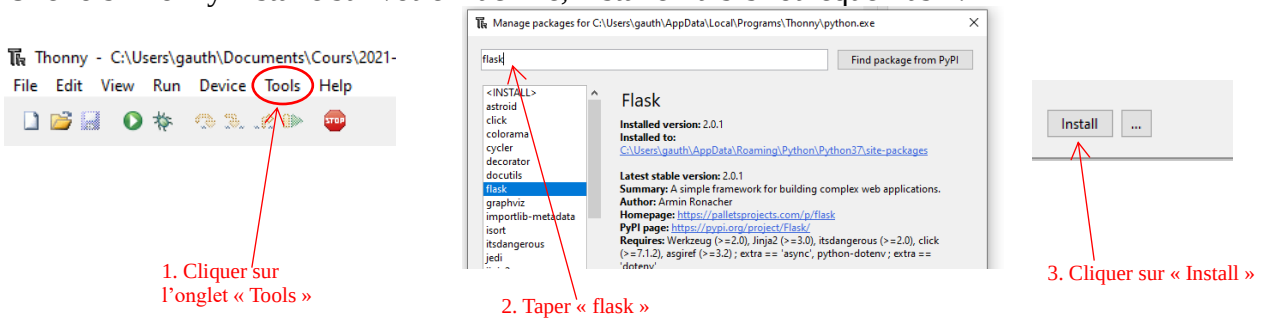
La bibliothèque **sqlite3** de Python permet de transformer une interface Python en un SGBD.
Pour comprendre son utilisation, réaliser le notebook se trouvant sur le serveur Jupyter et se nommant Chap1sqlite3.

II) Créer des pages Web dynamiques grâce à la Bibliothèque « flask » de Python.

Comme nous l'avons déjà vu en première dans la partie « web », un serveur web (aussi appelé serveur HTTP) est un ordinateur robuste qui permet de répondre à une requête HTTP effectuée par un client (très souvent un navigateur web).
Nous allons ici travailler avec notre propre ordinateur comme serveur web.
Nous utiliserons donc 2 logiciels sur le même ordinateur : le client (logiciel = navigateur web) et le serveur (logiciel = serveur web), ces 2 logiciels vont communiquer en utilisant le protocole HTTP.
Il existe de nombreux logiciels serveurs web, et le plus utilisé se nomme Apache. Nous allons ici travailler avec le logiciel serveur web Python Flask. Python Flask va nous permettre de générer des pages web côté serveur avec le langage Python. À noter qu'il est aussi possible d'utiliser d'autres langages côté serveur tels que Java, C# et surtout PHP.

1. Préparation de son espace de travail.

- Nous allons utiliser l'interface Python ***Thonny*** que vous pouvez télécharger et installer ici.
  Une fois Thonny installé sur votre machine, installer la bibliothèque flask :



- Nous allons aussi utiliser ***Sublime Text 3*** que vous pouvez télécharger et installer ici.

2. Création de pages web dynamiques.

a. Créer sur votre ordinateur un répertoire nommé "initiation_flask".
b. À l'aide de Thonny, créer un fichier Python "views.py" (sauvegardé dans le répertoire " initiation_flask ").

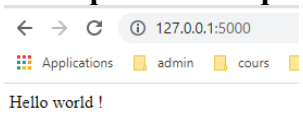
Saisir ce code suivant dans ce fichier en lisant attentivement les commentaires :

```
#importation des outils utiles de flask.
from flask import Flask , render_template , request

#création d'un outil app.
app = Flask(__name__)

#décorateur (hors programme) qui permet d'exécuter la
#fonction lorsque le serveur recevra une requête http avec
#comme URL la racine du site ('/').
@app.route('/')
def index():
 return "<p>Hello World !</p>"

#lancement du serveur
app.run()
```

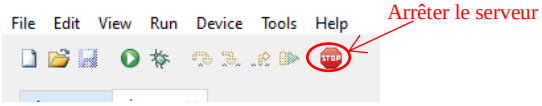
c. Le serveur est alors actif comme l'indique ceci :

```
Python 3.7.9 (bundled)
>>> %Run views.py

 * Serving Flask app 'views' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

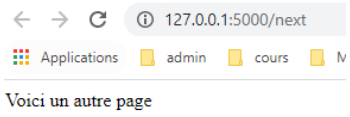
d. Ouvrir son navigateur, et taper l'URL mentionné en bleu (ou simplement cliquer sur le lien).
Dans cet exemple http://127.0.0.1:5000/ **(il faut taper le vôtre qui n'est pas le même !)**
Il doit s'afficher la page web ci-contre :



e. Arrêter le serveur sur Thonny, puis compléter le programme "views.py" comme ci-contre.



Exécuter à nouveau le programme "views.py".
Puis dans votre navigateur, compléter l'URL mentionné en rouge par « /next ».
Dans cet exemple http://127.0.0.1:5000/next
(il faut taper le vôtre !)
Il doit s'afficher la page web ci-dessous :




```
#importation des outils utiles de flask.
from flask import Flask , render_template , request

#création d'un outil app.
app = Flask(__name__)

#décorateur (hors programme) qui permet d'exécuter la
#fonction lorsque le serveur recevra une requête http avec
#comme URL la racine du site ('/').
@app.route('/')
def index():
 return "<p>Hello World !</p>"

@app.route('/next')
def autre_page():
 return "<p>Voici un autre page</p>"


#lancement du serveur
app.run()
```

g. Dans le répertoire "initiation_flask" créer un répertoire "templates" et un répertoire "static" dans lesquels on va sauvegarder respectivement "index.html" et "result.html" puis "style.css".
A l'aide de Sublime Text 3, créer les fichiers "index.html", "result.html" puis "style.css" qui seront bien sauvegardés dans les répertoires mentionnés juste avant.

"index.html" dans le répertoire "templates"

```
<!doctype html>
<html lang="fr">
        <head>
                <meta charset="utf-8">
                <title>index</title>
                <link rel="stylesheet" href="{{url_for('static', filename='style.css')}}">
        </head>
        <body>
          <h1>Mon site avec Flask</h1>
          <p>Hello World !</p>
        </body>
</html>
```

"result.html" dans le répertoire "templates"

```
<!doctype html>
<html lang="fr">
        <head>
                <meta charset="utf-8">
                <title>result</title>
                <link rel="stylesheet" href="{{url_for('static', filename='style.css')}}">
        </head>
        <body>
          <h1>Ma page dynamique</h1>
          <p>C'est ici que je dois avoir du contenu dynamique !</p>
        </body>
</html>
```

"style.css" dans le répertoire "static"

```
body{
background : orange ;
}
h1{
        text-align: center;
        color : red;
}

p,form{
        color : blue;
}
```

Arrêter le serveur sur Thonny (en cliquant sur le bouton « stop » comme à la question e.).
Dans le programme "views.py", remplacer dans la fonction index la ligne ***return " Hello World ! "***
par ***return render_template("index.html ")***.
Puis dans la fonction autre_page remplacer la ligne ***return " Voici un autre page "***
par ***return render_template("result.html")***.

Enfin dans votre navigateur, taper l'URL http://127.0.0.1:5000 puis après http://127.0.0.1:5000/next
(il faut taper les vôtres qui ne sont pas les mêmes !)

h. Pour l'instant nos pages web ne sont pas dynamiques et la navigation de l'une à l'autre n'est pas fluide. Nous allons améliorer ceci.

- Pour créer un formulaire, remplacer le contenu de la balise <body> du fichier "index.html" par :

```
<body>
  <h1>Mon site avec Flask</h1>
  <form action="http://localhost:5000/next" method="post">
    <label>Nom</label> : <input type="text" name="nom" />
    <label>Prénom</label> : <input type="text" name="prenom" />
    <input type="submit" value="Envoyer" />
  </form>
</body>
```

- Pour faire intervenir les résultats du formulaire dans l'autre page web, remplacer le contenu de la balise du fichier "result.html"

```
<p>Bonjour {{prenom}} {{nom}}, voici du dynamisme dans ta page !</p>
```

- Arrêter le serveur sur Thonny (en cliquant sur le bouton « stop » comme à la question e.).

Dans le programme "views.py", remplacer la fonction autre_page et son décorateur par :

```
@app.route('/next',methods=['POST']) #on utilise la méthode POST (et pas GET).
def autre_page():
    reponse = request.form #on obtient la réponse du formulaire sous form d'un dictionnaire.
    n = reponse['nom'] #récupération du nom = valeur de la clé 'nom' du dictionnaire
    p = reponse['prenom'] #récupération du prénom = valeur de la clé 'prenom' du dictionnaire
    return render_template("result.html", nom = n , prenom = p) #les valeurs de nom et prenom
envoyées à result.html
```

- Enfin dans votre navigateur, taper l'URL http://127.0.0.1:5000 **(il faut taper le vôtre !)** et se laisser guider.

i. Pour ajouter enfin de la fluidité dans la navigation, nous allons ajouter un lien permettant de revenir à la première page.
Pour se faire, il suffit d'ajouter la ligne ci-contre à la fin de la balise <body> de la page "result.html".

```
<p><a href="{{ url_for('index') }}">Retour au menu</a></p>
```

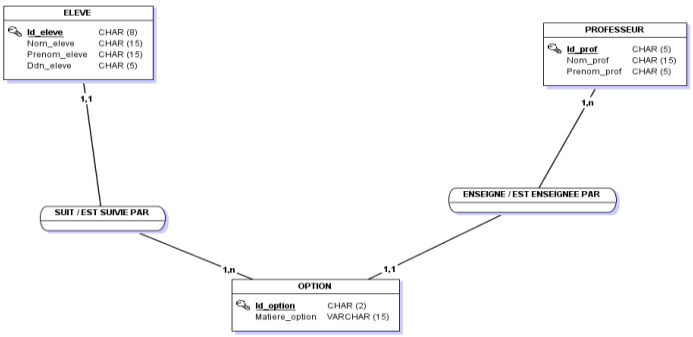
- Enfin dans votre navigateur, taper l'URL http://127.0.0.1:5000 **(il faut taper le vôtre !)** et se laisser guider.

III) Projet

Dans une université, chaque élève de première année doit suivre une option obligatoirement et une seule. De plus une même option est enseignée par un seul professeur, et un professeur peut enseigner plusieurs options.

En revanche il peut y avoir plusieurs groupes d'options de la même matière.

Vous êtes membre du service informatique de cette université et on vous met à disposition une Base de Données se nommant ***BDD_projet.db*** dont le MCD est ci-dessous :



Cette base de données qui vous est donnée regroupe au départ les données du tableur ci-dessous :

Id_eleve	*Nom_eleve*	*Prenom_eleve*	*Ddn_eleve*	*Id_prof*	*Nom_prof*	*Prenom_prof*	*Id_option*	*Matière_option*
BOWa0501	BOURGY	Wael	12/05/2001	GAJé	GAUTHIER	Jérôme	M1	MATHS
DEAl0700	DESSENS	Alexandre	05/07/2000	GAJé	GAUTHIER	Jérôme	M2	MATHS
OMLo1201	OMAIS	Loana	15/12/2001	ROFr	ROMPTEAUX	Freddy	H1	HISTOIRE
DEMa0101	DESVAUX	Margot	25/01/2001	MACa	MAZEAU	Caroline	A1	ANGLAIS
DEVi0700	DESSENS	Vincent	05/07/2000	HEJu	HERNADEZ	Juan	E1	ESPAGNOL
PADa1200	PAYEN	Damien	28/12/2000	ROFr	ROMPTEAUX	Freddy	H1	HISTOIRE
GAAx0301	GARNIER	Axelle	14/03/2001	ROFr	ROMPTEAUX	Freddy	H2	HISTOIRE

Vous commencerez par établir le MLD de cette base de données (à réaliser sur un fichier converti au format pdf), puis vous devrez mettre au point une plateforme en ligne qui permet de gérer cette base de données. Plus précisément, à l'ouverture de cette plateforme en ligne, on accèdera à une page « menu » qui nous permettra d'obtenir :

- ➔ Une page web permettant d'enregistrer dans la base le nom et le prénom d'un nouveau professeur (Id_prof devra être généré automatiquement = 2 premières lettres du nom, puis du prénom).
- ➔ Une page web permettant d'enregistrer dans la base l'identifiant, la matière et l'identifiant du professeur d'une nouvelle option.
- ➔ Une page web permettant d'enregistrer dans la base le nom, le prénom, la date de naissance et l'identifiant de l'option d'un nouvel élève (Id_eleve devra être généré automatiquement = 2 premières lettres du nom, puis du prénom, et le mois et l'année de naissance).
- ➔ Une page web permettant de supprimer un élève (en cas de départ en cours d'année scolaire).
- ➔ Une page web permettant de mettre à jour le nom et le prénom d'un professeur (en cas de changement de professeur en cours d'année scolaire), sans changer son identifiant.
- ➔ Une page web permettant d'obtenir la liste des élèves suivant une option spécifique (à partir de son Id).
- ➔ Une page web permettant d'obtenir la liste des élèves suivant une matière donnée en option.
- ➔ Une page web permettant d'obtenir le nom et le prénom du professeur de l'option d'un élève (à partir de son Id).

Voici, pour bien comprendre ce qu'il faut faire, ce à quoi doit ressembler la plateforme à créer : plateforme.

Tous les fichiers de votre projet devront être placés dans un répertoire compressé nommé « projet1 » et déposé dans votre espace personnel du groupe « NSI » de Teams.

Cahier des charges / Grille d'évaluation

MLD réalisé sur un fichier converti en pdf.	**/ 2**
Bonne organisation des différents fichiers (.py avec Thonny et .html ou .css avec sublime Text 3) dans plusieurs répertoires bien nommés, et tous placés dans un répertoire compressé, lui-même déposé sur son espace personnel de l'équipe Teams de NSI.	**/ 2**
Réalisation correcte des fichiers .html et .css en cohérence avec le cours de première, mais aussi avec l'utilisation de la bibliothèque flask de Python.	**/ 2**
Réalisation de commentaires permettant une bonne compréhension du fichier .py.	**/ 2**
Utilisation de Python avec la bibliothèque sqlite3 comme un SGBD pour modifier la BDD ou récupérer les résultats d'interrogations de la BDD.	**/ 5**
Utilisation de Python avec la bibliothèque flask pour créer des pages web dynamiques à partir de réponses d'un utilisateur à des formulaires et qui seront traitées côté serveur par Python (sqlite3) pour obtenir des réponses à des interrogations de la BDD, ou des modifications de la BDD.	**/ 5**
Rende ponctuel d'un projet abouti et fonctionnel.	**/ 2**
NOTE GLOBALE POUR LE PROJET :	**/ 20**