Un serveur web (serveur HTTP) permet de répondre à une requête HTTP effectuée par un client en général un navigateur web. Nous allons travailler avec un serveur web qui est installé sur votre ordinateur. Comme tout se passe sur notre ordinateur, on n'a pas de nom de domaine et il est remplacé par localhost par convention. Il existe de nombreux serveurs web, le plus utilisé se nomme Apache. Pour des raisons pratiques, nous allons travailler avec le framework Python Flask. Ce framework va nous permettre de générer des pages web côté serveur, il possède son propre serveur web.
Par défaut, le serveur HTTP intégré à Flask fonctionne sur le port 5000, il faut donc le préciser dans l'adresse du navigateur. Ainsi si on veut accéder à la page `maPage.html`, il faudra écrire l'adresse `http://localhost:5000/maPage.html`
Nous allons travailler dans Spyder pour écrire les scripts python. Les fichier html, css et js seront écrits dans NotePad ++.

Commencer par créer un répertoire `mesFlask` dans le quel vous enregistrerez tout ce qui concerne cette partie.

Remarque : Vous pouvez aussi utiliser un jupyter notebook (le serveur utilise le port 8888)
Les notebook en ligne ne conviennent pas pour utiliser flask, le serveur se lance mais on ne peut pas se connecter.

1 Exemple 1 : première approche

1.1 Exemple 1

- Dans le répertoire `mesFlask`, créer un répertoire `essai1`
- Ouvrir Spyder et recopier le code python ci-dessous dans un script puis enregistrer votre fichier sous le nom `serveur1.py` dans le répertoire `essai1`

```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "<p>Hello World!</p>"

app.run()
```

- Exécutez le code et laisser tourner le programme. En exécutant le programme Python, le framework Flask a lancé un serveur web. Ce serveur web attend des requêtes HTTP sur le port 5000.
- Dans Firefox, saisir l'adresse suivante `http://localhost:5000/`. Si tout se passe bien vous devriez avoir une page avec écrit "Hello World !"
  En ouvrant un navigateur web (de préférence Firefox) et en tapant `"localhost:5000"`, nous faisons une requête HTTP, le serveur web fourni avec Flask répond à cette requête HTTP en envoyant une page web contenant uniquement le paragraphe `<p>Hello World!</p>`
- Pour quitter (stopper) le serveur, faites ce qui est indiqué dans la console : `Press CTRL+C to quit`
  C'est aussi dans la console que vous verrez les échanges entre le serveur et le client.

```
In [1]: runfile('/disk2/ego_docs/Nath/MesNoteBooks/Serveur/mesFlask/serveur1.py', wdir='/disk2/
ego_docs/Nath/MesNoteBooks/Serveur/mesFlask')
 * Serving Flask app "serveur1" (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)
```

1.2 Explications du code python

- on importe le module Flask.

  ```
  from flask import Flask
  ```

- On donne un nom à l'application ici app : c'est indispensable.

  ```
  app = Flask(__name__)
  ```

- `@app.route` s'appelle un décorateur et permet de préciser à quelle adresse ce qui suit va s'appliquer. La fonction qui suit ce décorateur (ici hello) sera exécutée dans le cas où le serveur web recevra une requête HTTP avec une URL correspondant à la racine du site ('/'), c'est à dire, dans notre exemple, le cas où on saisie dans la barre d'adresse `"localhost:5000/"` (ou simplement `"localhost:5000"`)

  ```
  @app.route("/")
  def hello():
      return "<p>Hello World!</p>"
  ```

- la dernière ligne `app.run()` permet de lancer le serveur

  La page renvoyée est très basique et très incomplète, il faudrait un peu de style et peut-être d'autres pages plus complexes.

2 Exemple 2 : structurer

2.1 Exemple 2

- Dans le répertoire `mesFlask`, créer un répertoire `essai2` dans lequel vous enregistrerez tout ce qui suit.
- Dans répertoire `essai2`, créer un répertoire `static` et un répertoire `templates`. Attention, les noms sont importants, Flask "sait" que c'est dans ces dossiers qu'il doit aller chercher les éléments qui l'intéressent.
- Dans un nouveau script python que vous enregistrer dans le répertoire `essai2` sous le nom `serveur2.py`, recopier le code suivant :

  ```
  from flask import Flask, render_template
  app = Flask(__name__)

  @app.route("/")
  def hello():
      return render_template("index.html")

  app.run()
  ```

- Recopier le code html ci-dessous et enregistrer le fichier dans le répertoire `templates` sous le nom `index.html`

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

- Recopier le code css ci-dessous et enregistrer le fichier dans le répertoire `static` sous le nom `style.css`.

```
p {
  color : red ;
  text-align : center;
}
```

- Exécuter le script `serveur2.py` dans Spyder.
  Dans Firefox, saisir l'adresse suivante `http://localhost:5000/` pour voir le résultat.

2.2 Explications

- Dans le fichier python, on importe la fonction `render_template` qui va permettre de faire le lien entre le fichier `.py` et la page html prédéfinie. Dans la fonction `hello`, au lieu de renvoyer le texte directement, on demande d'afficher la page `index.html`.
- Dans le fichier html, pour indiquer à flask où chercher le fichier de style, on utilise l'instruction suivante pour indiquer l'adresse de la page :
  `" url_for('static', filename='style.css') "‘`

3 Exemple 3 : une deuxième page

3.1 Exemple 3

- Stopper le serveur précédent.
- Dans le répertoire `mesFlask`, faire une copie du répertoire `essai2` et l'enregistrer sous le nom `essai3`
- Renommer le fichier `serveur2.py` en `serveur3.py`
- Recopier le code html ci-dessous et enregistrer le fichier sous le nom `page2.html` dans le répertoire `templates`(sous-répertoire de `essai3`)

```
<!doctype html>
<html lang="fr">
    <head>
         <meta charset="utf-8">
             <link rel ="stylesheet" type="text/css"
             href="{{ url_for('static', filename='style.css') }}">
    </head>
    <body>
        <p id ="bleuCentre"> Voici la deuxième page avec le message :
```

```
            {{bonjour}}</p>
    </body>
</html>
```

- Dans le fichier `style.css` (dans le répertoire `static`), rajouter le code suivant :

```
#bleuCentre{
    color : blue ;
    text-align : center ;
}
```

- Dans le fichier python `serveur3.py`, rajouter le code suivant avant l'instruction `app.run()` :

```
@app.route("/page2")
def page():
    return render_template("page2.html", bonjour = "Hello World!")
```

- Exécuter le script `serveur3.py` dans Spyder.
  Dans Firefox, saisir l'adresse suivante `http://localhost:5000/page2`
- Arrêter le serveur, modifier le message contenu dans la variable `bonjour` puis exécuter le script et rafraîchir votre page 2.

3.2 Explications

Dans la fonction page du script python, on renvoie vers le fichier `page2.html` dans le quel on a injecté le contenu de la variable `bonjour`. L'intégration de ce que python injecte dans la page se fait grâce au caractère {{}} ici on aura donc le contenu de la variable `bonjour` (ici "Hello World !") qui sera affiché dans le texte de la balise <p>.

4 Exemple 4 : lien vers une page

4.1 Exemple 4

- Stopper le serveur précédent.
- Dans le répertoire `mesFlask`, créer un répertoire `essai4`.
- Dans répertoire `essai4`, créer un répertoire `static` et un répertoire `templates`.
- Dans un nouveau script python que vous enregistrerez dans le répertoire `essai4` sous le nom `serveur4.py`, recopier le code suivant :

```
from flask import Flask, render_template
app = Flask(__name__)

@app.route("/")
def hello():
    return render_template("index.html",
    message_bienvenue = "Bienvenue sur la page d'accueil !")

@app.route("/suivante")
def suite():
    return render_template("page2.html")

app.run()
```

- Recopier le code html ci-dessous et enregistrer le fichier dans le répertoire `templates` sous le nom `index.html`

```
<!doctype html>
<html lang="fr">
    <head>
        <meta charset="utf-8">
       <link rel ="stylesheet" type="text/css"
         href="{{ url_for('static', filename='style.css') }}">
     </head>
    <body>
        <h1 > {{ message_bienvenue }}</h1>
        <a href="{{ url_for('suite') }}"> Vers la page 2 </a>
    </body>
</html>
```

- Recopier le code html ci-dessous et enregistrer le fichier dans le répertoire `templates` sous le nom `page2.html`

```
<!doctype html>
<html lang="fr">
    <head>
        <meta charset="utf-8">
       <link rel ="stylesheet" type="text/css"
         href="{{ url_for('static', filename='style.css') }}">
    </head>
    <body>
        <h1 > Vous êtes sur la deuxième page </h1>

        <a href="{{ url_for('hello') }}"> Vers la page d'accueil </a>
    </body>
</html>
```

- Recopier le code css ci-dessous et enregistrer le fichier dans le répertoire `static` sous le nom `style.css`.

```
h1{
    color : #007f99 ;
}
```

- Exécuter le script `serveur4.py` dans Spyder.
  Dans Firefox, saisir l'adresse suivante `http://localhost:5000/`.

4.2 Explications

Pour indiquer à flask, à quelle page il doit se rendre en cliquant sur le lien, il faut lui donner le **nom de la fonction** définie pour la page et non la route pour y aller. Dans l'exemple, on indique bien `url_for('suite')` et non pas `url_for('suivante')`. De même, dans `page2.html` on indique `url_for('hello')`.

5 Exemple 5 : une page avec du javascript

- Stopper le serveur précédent.
- Dans le répertoire `mesFlask`, faire une copie du répertoire `essai4` et l'enregistrer sous le nom `essai5`
- Renommer le fichier `serveur4.py` en `serveur5.py`
  Nous allons écrire un fichier js que l'on liera à `page2.html` qui fera en sorte que lorsqu'on cliquera sur le titre, il passera en rouge.
- Modifier le fichier `page2.html` qui se trouve dans le répertoire `templates` pour ajouter un identifiant à la balise h1 et un lien vers le script js. On obtient ceci :

```
<!doctype html>
<html lang="fr">
    <head>
        <meta charset="utf-8">
          <link rel ="stylesheet" type="text/css"
           href="{{ url_for('static', filename='style.css') }}">
        <script defer="defer" language="JavaScript"
        src= "{{ url_for('static', filename='script.js') }}"></script>
    </head>
    <body>
       <h1 id = "Orange" onClick = "enOrange()"> Vous êtes sur la deuxième page </h1

        <a href="{{ url_for('hello') }}"> Vers la page d'accueil </a>
    </body>
</html>
```

- Recopier le code js ci-dessous et enregistrer le fichier dans le répertoire `static` sous le nom `scrip.js`.

```
function enOrange(){
    let h1 = document.getElementById("Orange");
    h1.style.color = "#e6b63d";
}
```

  Exécuter le script `serveur5.py` dans Spyder.
  Dans Firefox, saisir l'adresse suivante `http://localhost:5000/`
  Aller sur la deuxième page et cliquer sur le texte "Vous êtes sur la deuxième page" : il doit passer en orange.

Utiliser un serveur est indispensable si vous voulez utiliser un formulaire pour enregistrer la saisie de l'utilisateur ou autoriser l'accès à d'autres pages

6 Exemple 6 : une page avec l'heure et la date

- Dans le répertoire `mesFlask`, créer un répertoire `essai6` dans lequel vous enregistrerez tout ce qui suit.
- Dans répertoire `essai6`, créer un répertoire `static` et un répertoire `templates`.
- Dans un nouveau script python que vous enregistrer dans le répertoire `essai6` sous le nom `serveur6.py`, recopier le code suivant :

```
from datetime import date
from datetime import datetime

from flask import Flask, render_template

app = Flask(__name__)

moment = date.today().strftime("%d /%m/%Y")
H = datetime.now().strftime("%H:%M:%S")
bonjour = "Hello World!"

@app.route("/")
def hello():
    return render_template("index.html", bonjour =  "Hello World!",
     moment = moment, heure = H)

app.run()
```

- Recopier le code html ci-dessous et enregistrer le fichier dans le répertoire `templates` sous le nom `index.html`

```
<!doctype html>
<html lang="fr">
    <head>
        <meta charset="utf-8">
    </head>
    <body>
        <h1>  {{bonjour}}</h1>
        <p>Nous sommes le {{moment}} et il est {{heure}}</p>
    </body>
</html>
```

- Observer dans la fonction `render_template`, les différentes façons d'indiquer les valeurs des variables apparaissant dans le code html.
- Exécuter le script `serveur6.py` dans Spyder.
  Dans Firefox, saisir l'adresse suivante `http://localhost:5000/` pour voir le résultat.

7 Exemple 7 : un formulaire simple

7.1 Exemple 7

- Stopper le serveur précédent.
- Dans le répertoire `mesFlask`, créer un répertoire `essai7`.
- Dans répertoire `essai7`, créer un répertoire `static` et un répertoire `templates`.
- Recopier le code html ci-dessous et enregistrer le fichier dans le répertoire `templates` sous le nom `index.html`

```
<!doctype html>
<html lang="fr">
    <head>
        <meta charset="utf-8">
    </head>
<body>
     <h1>Indiquer votre nom et votre prénom</h1>
        <form action = "http://localhost:5000/bonjour" method="POST">
            <p><label>Nom</label> :  <input type="text" name="Nom"></p>
            <p><label>Prénom</label> :  <input type="text" name="Prénom"></p>
            <p><input value="Valider" type="submit"></p>
        </form>
</body>
</html>
```

- Recopier le code html ci-dessous et enregistrer le fichier dans le répertoire `templates` sous le nom `bonjour.html`

```
<!doctype html>
<html lang="fr">
    <head>
        <meta charset="utf-8">
    </head>
    <body>
      <h1> Bienvenue {{ message }} </h1>
      <a href="{{ url_for('home') }}"> Retourner à la page d'accueil </a>
    </body>
</html>
```

  Dans la balise form, on indique la méthode POST pour dire qu'on va s'en servir pour que l'utilisateur puisse fournir des informations. On indique aussi que ce qui sera donné est considéré comme une string. On a aussi un bouton "OK" pour envoyer le formulaire.
- Dans le fichier py :

```
from flask import Flask, render_template, request

app = Flask(__name__)

@app.route('/')
def home():
    return render_template("index.html")

@app.route('/bonjour', methods=['POST'])
def hello():
```

```
        resultat = request.form
        nom = resultat['Nom']
        prenom = resultat['Prénom']
        nomComplet = nom + " " + prenom
        return render_template("bonjour.html" , message = nomComplet)

app.run()
```

7.2 Explications

Dans le fichier html, la balise `form`, on utilise l'attribue `action` dans lequel on renseigne l'adresse du serveur vers lequel on veut renvoyer le contenu du formulaire.
La fonction `hello` va permettre de récupérer le nom de l'utilisateur et l'afficher, ici dans une autre page. Pour récupérer les informations envoyées par l'utilisateur, on va importer une nouvelle méthode de flask, la méthode request.
Ensuite, il faut utiliser la méthode "form" de l'objet request, qui contient toutes les données du formulaire envoyé en POST.
L'objet `request.form` référencé ici par la variable `resultat` est un objet python de type :
`<class 'werkzeug.datastructures.ImmutableMultiDict'>`.
Les "clés" sont les chaînes de caractères contenues dans les attributs `name` des éléments de notre formulaire. Nous n'étudierons pas cette structure en détail.
Exécuter le script `serveur7.py` dans Spyder.
Dans Firefox, saisir l'adresse suivante `http://localhost:5000/`, remplir le formulaire et l'envoyer.
Observez les requêtes dans la console de Spyder et l'url dans la barre de navigation de firefox.

7.3 Méthode GET ou POST

Lorsqu'un formulaire envoie des données au serveur, deux méthodes de transmission sont possibles, la méthode GET et la méthode POST (la méthode GET est celle par défaut). La méthode GET communique les données du formulaire au serveur en les intégrant directement dans l'url. Pour la méthode POST, les données du formulaire sont empaquetées dans une requête HTTP - HyperText Transfer Protocol - et ne sont donc pas visibles par l'utilisateur.

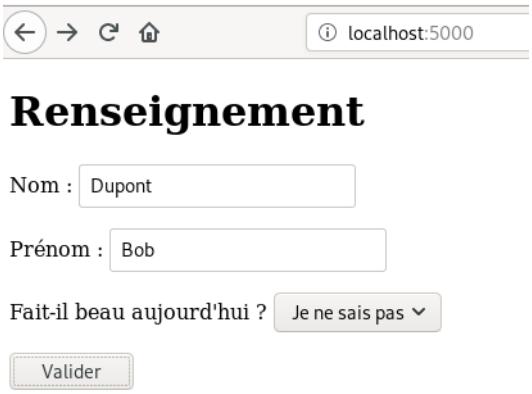
- Arrêter le serveur et modifier le code python précédent en remplaçant :
  - `methods=['POST']` par `methods=['GET']`
  - et en remplaçant `resultat = request.form` par `resultat = request.args`
- Modifier dans le fichier index.html, la methode de la balise form en remplaçant `method="POST"` par `method="GET"`
- Exécuter le script `serveur7.py` dans Spyder.
  Dans Firefox, aller à l'adresse suivante `http://localhost:5000/`, remplir le formulaire et l'envoyer.
- Observez les requêtes dans la console de Spyder et l'url dans la barre de navigation de firefox.

  Vous remarquerez que cette URL est `http://localhost:5000/?Nom=Dupont&Prénom=Bob` si vous avez saisie Dupont et Bob.
  Cette URL contient la base indiquée dans l'attribut `action` de la balise `<form>`suivie de la séquence : `?Nom=...&Prénom=`. Le "?" sépare la base de l'URL des données du formulaire qui sont séparées entre elles par le caractère "&" lorsqu'il y a plusieurs données.

Remarque :
On pourrait croire que la méthode POST est plus sûre que la méthode GET puisque les données du formulaire ne sont pas visibles dans l'url. Cependant, il est important de bien comprendre que la méthode "POST" n'offre pas non plus une sécurité absolue puisque toute personne ayant un bagage technique minimum sera capable de lire les données transmises à l'aide de la méthode "POST" en analysant la requête HTTP, même si ces données ne sont pas directement visibles dans l'URL. Seule l'utilisation du protocole sécurisé HTTPS garantit un transfert sécurisé des données entre le client et le serveur (les données sont chiffrées et donc illisibles pour une personne ne possédant pas la clé de déchiffrement).
Pour un formulaire, il faut toujours privilégier la méthode POST.

8 Exercice à rendre : un formulaire plus complet

- Stopper le serveur précédent.
- Dans le répertoire `mesFlask`, créer un répertoire `essai8`.
- Dans répertoire `essai8`, créer un répertoire `static` et un répertoire `templates`.
- Écrire un fichier html à enregistrer dans le répertoire `templates` sous le nom `index.html` et qui doit ressembler à ceci :



- On aura le choix entre oui, non ou je ne sais pas
- Écrire un fichier html à enregistrer dans le répertoire `templates` sous le nom `page2.html` et qui doit ressembler à ceci :

- Vous pouvez rajouter une feuille de style, un script javascript ...
- Vous pouvez aussi rajouter des éléments dans le formulaire.
- Vous pouvez rajouter d'autres pages ...
- Vous compresserez le dossier `essai8` et vous déposerez le fichier .zip dans le dépôt de devoir dédié sur l'ENT.