Flask - Les routes | Exercices

Voici quatre exercices progressifs et complets pour apprendre à utiliser les routes dans Flask. Chaque exercice introduit un nouveau concept, et vous aidera à comprendre comment gérer les routes et les requêtes HTTP dans une application web Flask.

5TTR 6TTR Intérmédiaire

Exercice 1 : Création de routes de base

Objectif

Dans cet exercice, vous allez créer une application Flask avec plusieurs routes simples qui affichent des messages différents selon l'URL visitée.

Instructions

1. Créez un projet Flask.
2. Ajoutez trois routes distinctes dans l'application Flask :
   - Une route pour la page d'accueil (`/`) qui affiche le message "Bienvenue sur la page d'accueil".
   - Une route pour une page "contact" (`/contact`) qui affiche le message "Page de contact".
   - Une route pour une page "à propos" (`/about`) qui affiche le message "À propos de nous".

Code de base

```python
from flask import Flask

app = Flask(__name__)

@app.route('/')
def home():
    return "Bienvenue sur la page d'accueil"

@app.route('/contact')
def contact():
    return "Page de contact"

@app.route('/about')
def about():
    return "À propos de nous"
```

```
16
17  if __name__ == '__main__':
18      app.run(debug=True)
```

Objectifs d'apprentissage

- Comprendre la création de routes avec le décorateur `@app.route()`.
- Gérer plusieurs routes dans une application Flask.
- Exécuter et tester l'application dans un navigateur.

Exercice 2 : Routes avec paramètres dynamiques

Objectif

Dans cet exercice, vous allez ajouter une route dynamique qui accepte un paramètre dans l'URL et renvoie une réponse personnalisée.

Instructions

1. Ajoutez une nouvelle route qui accepte un paramètre `nom` dans l'URL. Par exemple, si l'utilisateur visite `/bonjour/John`, la page affichera "Bonjour, John".
2. Faites en sorte que cette route fonctionne avec n'importe quel nom.

Code de base

```python
from flask import Flask

app = Flask(__name__)

@app.route('/')
def home():
    return "Bienvenue sur la page d'accueil"

@app.route('/bonjour/<nom>')
def bonjour(nom):
    return f"Bonjour, {nom}"

if __name__ == '__main__':
    app.run(debug=True)
```

Test

- Visitez l'URL `/bonjour/John` et vérifiez que le message est bien personnalisé.
- Essayez d'autres noms comme `/bonjour/Marie` ou `/bonjour/Pierre`.

Objectifs d'apprentissage

- Utiliser des paramètres dynamiques dans les routes.
- Générer des réponses personnalisées en fonction de l'URL visitée.

Exercice 3 : Routes avec des types de paramètres

Objectif

Dans cet exercice, vous allez créer une route qui accepte des paramètres de types différents (entiers et chaînes de caractères) et gérez le type des données dans l'URL.

Instructions

1. Créez une nouvelle route qui accepte un **ID** utilisateur (un entier) et un nom d'utilisateur (une chaîne de caractères).
2. Affichez un message du type "Profil de l'utilisateur avec ID: `<id>` et nom: `<nom>`".
3. Utilisez le convertisseur `int` pour forcer l'ID à être un entier.

Code de base

```python
from flask import Flask

app = Flask(__name__)

@app.route('/')
def home():
    return "Bienvenue sur la page d'accueil"

@app.route('/profil/<int:id>/<nom>')
def user_profile(id, nom):
    return f"Profil de l'utilisateur avec ID: {id} et nom: {nom}"

if __name__ == '__main__':
    app.run(debug=True)
```

Test

- Visitez `/profil/1/John` pour tester la route.
- Essayez `/profil/2/Marie` pour voir le comportement avec un autre nom et ID.

Objectifs d'apprentissage

- Comprendre l'utilisation des types de paramètres dans les routes.
- Gérer des types de données différents dans une même URL (entiers et chaînes de caractères).

Exercice 4 : Routes avec méthodes HTTP (GET et POST)

Objectif

Dans cet exercice, vous allez créer une route qui gère les méthodes **GET** et **POST**. Vous ajouterez un formulaire HTML pour soumettre des données et les traiter avec Flask.

Instructions

1. Créez une route `/login` qui accepte les méthodes `GET` et `POST`.
2. Lorsque la méthode est `GET`, affichez un formulaire de connexion simple avec un champ pour le nom d'utilisateur.
3. Lorsque la méthode est `POST`, récupérez le nom d'utilisateur soumis et affichez "Connexion réussie pour `<nom d'utilisateur>`".

Code de base

```
from flask import Flask, request, render_template_string

app = Flask(__name__)

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        return f"Connexion réussie pour {username}"
return '''
        <form method="POST">
            Nom d'utilisateur: <input type="text" name="username">
            <input type="submit" value="Se connecter">
        </form>
```

```
    '''

if __name__ == '__main__':
    app.run(debug=True)
```

Test

- Accédez à la page `/login` pour voir le formulaire.
- Soumettez un nom d'utilisateur et vérifiez que la connexion est réussie.
- Testez avec plusieurs noms d'utilisateur.

Objectifs d'apprentissage

- Gérer plusieurs méthodes HTTP (`GET` et `POST`) avec une même route.
- Récupérer des données soumises via un formulaire HTML.
- Utiliser la méthode `POST` pour envoyer des données depuis le navigateur vers le serveur.

Ces exercices couvrent progressivement les principaux concepts liés aux routes dans Flask : la création de routes basiques, l'utilisation de paramètres dynamiques, la gestion de types de données dans les URL, et la gestion des différentes méthodes HTTP. Ils sont conçus pour guider les débutants dans l'apprentissage des fonctionnalités essentielles de Flask.