Flask

Fiche de synthèse

1 Le principe

Flask (`pip install flask`) est une bibliothèque Python pour écrire un **serveur web**. Le navigateur (client) envoie une requête HTTP ; Flask appelle la **vue**, la fonction associée à l'adresse ; ce qu'elle renvoie est la page.

```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def accueil():
    return "<h1>Bonjour !</h1>"

@app.route("/table/<int:n>")
def table(n):
    return f"<p>{n} x 2 = {n * 2}</p>"

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

Le site se visite à l'adresse `http://127.0.0.1:5000`. Adresse sans route : erreur 404.

2 Gabarits

Dans le dossier `templates` ; `render_template("page.html", x=valeur)` transmet les variables. Jinja : `{{ x }}` insère, `{% for %}` répète, `{% if %}` choisit. Une page hérite d'une base : `{% extends "base.html" %}`, puis `{% block contenu %}`. CSS et images dans `static`, appelés par `url_for('static', filename='style.css')`.

3 Formulaires

```
from flask import request

@app.route("/imc", methods=["GET", "POST"])
def imc():
    message = ""
    if request.method == "POST":
        m = float(request.form["masse"])
        t = float(request.form["taille"])
        message = f"IMC : {m / t ** 2:.1f}"
    return render_template("imc.html",
                           message=message)
```

GET : données dans l'adresse, `request.args.get("mot", "")`. POST : `request.form`, et `methods` sinon erreur 405. Les données arrivent en chaînes.

4 Session

HTTP est sans mémoire. `session["score"] = 3` garde une valeur propre à chaque visiteur, dans un cookie signé par `app.secret_key`. Après un POST, `redirect(url_for("vue"))`. Mots de passe : enregistrer une empreinte (`generate_password_hash`), jamais le mot de passe.

5 Serveur et navigateur

Python et Jinja s'exécutent sur le **serveur** (partie serveur, *back end*) ; HTML, CSS et JavaScript dans le **navigateur** (partie client, *front end*). Flask envoie les fichiers de `static` sans les exécuter ; un `print` d'une vue s'affiche dans le terminal du serveur. En ligne : un hébergeur, un fichier `requirements.txt` (`flask`, `gunicorn`) et la commande `gunicorn app:app` ; jamais `debug=True`.

À retenir.

- Une adresse, une vue ; `templates` et `static`.
- `request.args` (GET), `request.form` (POST).
- `session` et `secret_key` pour se souvenir d'un visiteur.
- Python sur le serveur, JavaScript dans le navigateur.