Première page internet avec Flask

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

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

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

Explication du code pas à pas

Ce code commence par importer le module Flask.

```
from flask import Flask
```

On donne ensuite un nom à l'application ici ce sera app

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

Ensuite vient la partie cruciale : définir une page (ou route) avec flask

@app.route permet de préciser à quelle adresse ce qui suit va s'appliquer.

Ici comme on est sur la page d'accueil, on met juste ("/")

@app.route(« / »)

Ensuite vient la fonction hello() qui va s'exécuter sur la page "/". On dit simplement que quand on arrive sur la page définie par la route juste au dessus, on va en même temps appeler la fonction hello().

Ici ça va donc afficher "Hello World" quand on arrive sur la page "/".

```
def hello():
        return "Hello World!"
```

Cette dernière partie permet juste de faire en sorte que l'application se lance quand on lance le code dans la console ou le terminal.

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

Ajouter un peu de style à la page

En gros, on va indiquer à Flask des templates de pages et des styles prédéfinis pour l'application. En ouvrant le dossier `hello_color` (le dossier de l'application) vous trouverez 3 éléments :

- `hello_world_green.py`
- un dossier `static` qui contient la feuille de style `main.css`
- un dossier `templates` qui contient une page `home.html`

La structure du dossier et le nom des dossiers `static` et `templates` sont importants : Flask sait que c'est dans ces dossiers là qu'il doit aller chercher les éléments qui l'intéressent.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

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

@app.route("/")
def hello():
    return render_template("home.html", message = "Hello World!")

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

Par rapport au précédent `hello_world.py`, on a modifié quelques éléments. On importe la méthode `render_template` qui va permettre de faire le lien entre le *.py* et la page html prédéfinie.
Ici, au lieu de renvoyer le texte directement, on demande d'injecter dans le template de la page home le message `Hello World`.

Le template home.html

Dans ces lignes, il y a deux éléments importants :

- l'appel au css pour avoir le style
- l'intégration de ce que python injecte dans la page

L'appel au css se fait dans la balise `<link>` où on indique le chemin pour accéder à la feuille de style. On intègre ce qui est envoyé grâce au

caractère `{{ }}` ici on aura donc le message `"Hello World"` qui sera affiché avec un type `h1`.

<!doctype html> <html lang= »en »>

<head>

<link rel = »stylesheet » type= »text/css »
href= »{{ url_for(“static”, filename=”main.css”) }} »

</head> <body>

<h1 > {{ message }} </h1>

</body>

</html>

Le main.css

Justement le `h1` dans le css, on lui dit à quoi il doit ressembler. On va lui donner une couleur verte et le centrer.

h1 {

font-size: 2em; color: green; text-align: center;

}