

## Lab 2: Création d'une application Web

Mohamed Dhaoui

Dans ce TP propose d'écrire une application Web pour jouer à un jeu simple : en partant d'un article Wikipédia, atteindre l'article « Philosophie » en suivant les liens d'un article à l'autre, et en aussi peu d'étapes que possible. L'application du TP permet de jouer à ce jeu en récupérant les pages de Wikipédia et en proposant les liens à l'utilisateur

Pour ce faire , on va utiliser le langage de programmation Python et le framework Web Flask. On fera appel au langages de l'environnement web comme html , css , javascript et des bibliothèques comme Bootstrap .

### 1- Interrogation de l'API Wikipédia :

---

Notre application web utilise l'API de Wikipedia pour renvoyer les 10 premiers liens de la page sélectionnée ou requêtée. Pour cela on a fait appel principalement aux fonctions ci-dessous, se trouvant dans le fichier `getpage.py` :

- `getRawPage` : une fonction qui prend comme argument un titre et renvoie un couple formé du nom de la page après résolution de redirections et le son contenu HTML .
- `getPage` : une fonction qui prend comme argument le titre d'une page et renvoie les 10 premiers liens qui s'y trouvent .

La fonction `getpage` tient compte de plusieurs contraintes pour filtrer les liens trouvés dans la réponse html de l'API :

- Ne prendre que les éléments dans des balises `<p>` hérités directement d'une balise `<div>` : pour ce faire , on a utilisé la fonction `.div.find_all("p", recursive=False)`

- Eliminer les liens externes et les liens rouges : Ceci a été fait en examinant la structure du lien et voir s'il contient des adresses en dehors de /wiki/ et /wikipedia et /w/ ou des redlinks ayant des titres après redirection « cette page n'existe pas »
- Eliminer les préfixes /wiki/ en utilisant la fonction `.replace()`
- Décoder les caractères non ASCII des titres de page et des liens
- Retirer les fragments des liens de page en cherchant les occurrences du caractère « # » dans le lien
- Remplacer les sous-tirets par des espaces en utilisant la fonction `replace()`
- Ne pas prendre en compte les liens en dehors de wiki , en se basant sur l'occurrence de ' : ' dans les liens
- Supprimer les doublons en gardant le même ordre d'apparition

## 2- Application web :

---

### 2.1 Flask

Le développement de l'application web est fait en utilisant Flask comme Backend et Jinja et HTML comme Front.

La partie Flask a été développée dans le fichier `philosophie.py`

on importe la classe Flask depuis le module flask, et on s'en sert pour instancier l'objet app. Cet objet est **fondamental** : il s'agit de notre application, ou de notre site web . En termes techniques, il s'agit d'une application WSGI...

Flask fonctionne avec les décorateurs en utilisant '@app.route' qui prend en paramètre une route. Cette route est celle par laquelle notre fonction sera accessible.

La route '/' est spéciale puisqu'elle représente la racine du site web. Il n'est donc pas besoin de la préciser dans l'adresse du navigateur. Dans le code ci-dessous , on appelle une fonction `index()` dès qu'on est sur la racine du site , cette fonction redirige vers une page « index.html » avec un message comme paramètre

```
@app.route('/', methods=['GET'])
def index():
    return render_template('index.html', message="Bonjour, monde !")
```

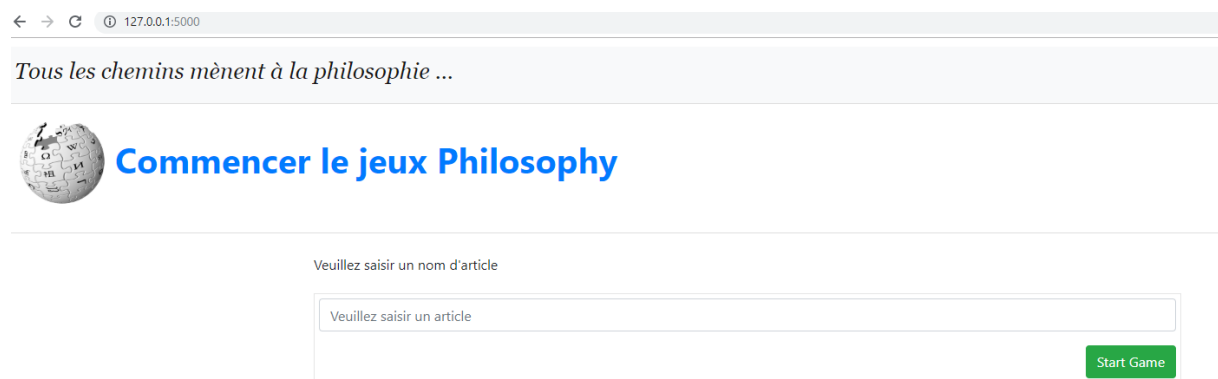
Pour lire le message depuis html , on utilise le langage Jinja :

```
<span class="hover1">{{ message }}</span>
```

Pour les routes , on a créé 4 :

- Route d'index qui permet de rediriger vers la page index.html

```
@app.route('/', methods=['GET'])
def index():
```



- Route « /new-game » qui lit l'input de l'utilisateur session['article'] et lui permet de commencer le jeux

```
@app.route('/new-game', methods=['POST'])
def NewGame():
```

- Route « /game » qui reçoit à chaque fois un nom de page , appelle la fonction getpage pour générer le titre et les liens de la page demandé et redirige vers game.html avec ces résultats .

```
@app.route('/game', methods=['GET'])
def Game():
```

← → 127.0.0.1:5000/game

Tous les chemins mènent à la philosophie ...

 **Titre de la page courante : sexe**

**Veuillez selectionner un article**

- ☒ Système reproducteur
- ☐ Comportement sexuel humain
- ☐ Sexualité
- ☐ Comparaison biologique entre la femme et l'homme
- ☐ Homme
- ☐ Femme
- ☐ Sexualité (reproduction)
- ☐ Ciliophora
- ☐ Organe sexuel
- ☐ Morphologie (biologie)

Votre parcours est : >>> Sexe

Votre score est 1

- La route « /move » s'exécute quand l'utilisateur appuie sur le bouton « move » , elle lit l'article sélectionné par l'utilisateur , met à jour le paramètre session['article'] et redirige vers la route /game pour générer les liens

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

## 2.2 Amélioration de fonctionnement de l'application web :

Ci-dessous les différents points qu'on a essayé d'améliorer au niveau de l'application :

- Plusieurs onglets : on a constaté que lorsque l'utilisateur commence à jouer sur un onglet et ouvre un autre onglet , le score affiché est la somme des scores des onglets ouverts . Pour remédier à cela , on a commencer par créer un champ `< input type='hidden' name='sessionidentifiant' >` dans lequel on stocke un id de session généré aléatoirement lorsque on appelle la fonction `new_game` , on a créé ensuite un dictionnaire de score ( variable public ) permettant de sauvegarde le score de chaque 'sessionidentifiant' . Du coup, pour incrémenter à chaque 'move' le score de 'sessionidentifiant' et on peut donc jouer sur plusieurs sessions simultanément. Une autre solution consiste à stocker directement le score dans le champ hidden .

- Si getpage n'extrait aucun lien , l'utilisateur aura un message indiquant qu'il a perdu :

← → ↻ 127.0.0.1:5000

Tous les chemins mènent à la philosophie ...



## Commencer le jeux Philosophy

---


Veuillez saisir un nom d'article

Start Game

**1ere page ne contient pas de lien ou n'existe pas ou elle qui fait reference directement à la philosophie , votre partie est finie**

-

- Si la page de départ n'existe pas ou c'est la page 'philosophie ' ou redirige vers celle-ci , l'utilisateur reçoit le même message que précédemment



## Commencer le jeux Philosophy

---

Veuillez saisir un nom d'article

Start Game

**1ere page ne contient pas de lien ou n'existe pas ou elle qui fait reference directement à la philosophie , votre partie est finie**

- Le premier bouton du radio doit être présélectionné : pour ce faire , on a ajouté l'attribut 'checked' au premier item seulement dans le fichier game.html



## Titre de la page courante : sexe

Veuillez selectionner un article

- ☒ Système reproducteur
- ☐ Comportement sexuel humain
- ☐ Sexualité
- ☐ Comparaison biologique entre la femme et l'homme
- ☐ Homme
- ☐ Femme
- ☐ Sexualité (reproduction)
- ☐ Ciliophora
- ☐ Organe sexuel
- ☐ Morphologie (biologie)

Move

Stop

Votre parcours est : >>> Sexe

Votre score est 1

- On a constaté que l'utilisateur peut tricher en utilisant le menu inspecter-élément du navigateur et en changeant l'attribut 'value' du radiobutton , pour remédier à cela , on vérifie dans la fonction move que l'article sélectionné existe bien dans la liste des liens proposée , si l'article n'existe pas dans , on lui affiche le message suivant :

← → ↻ 127.0.0.1:5000

Tous les chemins mènent à la philosophie ...



## Commencer le jeux Philosophie

Veuillez saisir un nom d'article

Veuillez saisir un article

Start Game

**Vous avez triché et votre partie est finie**

Elements Console Sources

```
<!doctype html>
<html lang="fr">
  <head>...</head>
  <body> ... $0
    <nav class="navbar navbar-expand-lg nav">
      <div class="card" style="width: 280rem;>
        <div class="container">...</div>
      </div>
    </body>
  </html>
```

## 2.3 Mise en forme CSS

Afin de soigner le design de l'application web, on a utilisé CSS pour les modifications demandées et Bootstrap pour le design des formulaires, les barres de titre et les boutons. Le code CSS se trouve dans le fichier `/static/style.css`. Ci-dessous quelques détails à propos des améliorations de design :

- Les pages `index.html` et `game.html` contiennent des titres (`<h1>`) permettant de rediriger vers la page de départ, le score de la partie et le titre de la page courante :

*Tous les chemins mènent à la philosophie ...*



**Titre de la page courante : Constitution**

Veuillez selectionner un article

- ☒ État
- ☐ Loi
- ☐ Pouvoir constituant
- ☐ Organisation internationale du travail
- ☐ Traité établissant une Constitution pour l'Europe
- ☐ Traité (droit international public)
- ☐ Constitution de la Californie
- ☐ Hiérarchie des normes
- ☐ Principe de constitutionnalité
- ☐ Constitution du Royaume-Uni

Move

Stop

Votre parcours est : >>> Santé > Constitution

Votre score est 2

- On a ajouté également le chemin parcouru des articles et un bouton 'stop' qui permet de quitter le jeux et revenir vers l'index
- Les formulaires sont centrés en ajoutant `'text-align: center;'` à la balise `body{}` et `'display: inline-block;'` à la balise `form` du fichier `css`. La largeur du formulaire est fixée avec l'attribut `'width'`

```
form {  
display: inline-block;  
width:1000px;  
border: 1px solid rgb(230, 230, 230);  
padding: 5px 5px;  
margin: 5px 0;}
```

- Les boutons du formulaire sont à droite du formulaire. Pour ce faire, on a utilisé `style="float: right;"`
- Les puces des boutons radio devraient être enlevés : Sur ce point, on a enlevé les puces en utilisant `'list-style: none;'` , puis on a modifié le fichier css avec les balises suivantes :

```
- .visibility {
-   visibility: hidden;
-
- }
-
- input[type=checkbox]+label {
-     font-weight: normal;
- }
- input[type=checkbox]:checked+label {
-     font-weight: bold;
- }
- input[type=checkbox]:focus+label {
-     border: 1px dotted #000;
- }
```

Cette modification n'était pas agréable à utiliser et manipuler, de ce fait, on a gardé la configuration initiale des radiobuttons , en ajoutant un espace en les puces et le label , et un 'hover' qui permet de changer la couleur des labels lorsqu'ils sont survolés par la souris . La largeur du 'hover' est la même que celui du formulaire.

```
label.hover1: hover {
color: rgb(48, 94, 47);
background-color: rgb(236, 235, 235);
height: 20px;
width: 1000px ;
line-height: 20px;
display: block;
}
```

- Les messages qu'on affiche à l'utilisateur sont de couleur différentes, le choix des couleurs dépend du contenu du message et cela a été fait avec jinja dans les fichiers html .



Pour le score et le parcours des pages , on affiche un message en bleu :

*Tous les chemins mènent à la philosophie ...*



## Titre de la page courante : Ciliophora

Veuillez selectionner un article

- ☒ Protozoaire
- ☐ Unicellulaire
- ☐ Cil vibratile
- ☐ Embranchement (biologie)
- ☐ Alveolata
- ☐ Organite
- ☐ Nutrition
- ☐ Motricité
- ☐ Excrétion
- ☐ Micromètre

Move

Stop

Votre parcours est : >>> Sexe > Ciliophora

Votre score est 2

Le message d'erreur est en rouge :

Veuillez saisir un nom d'article

Veuillez saisir un article

Start Game

**1ere page ne contient pas de lien ou n'existe pas ou elle qui fait  
reference directement à la philosophie , votre partie est finie**

Le message d'une partie gagnante en vert

Tous les chemins mènent à la philosophie ...



## Commencer le jeux Philosophy

Veillez saisir un nom d'article

Start Game

Partie gagnée !!

Votre score est 4

Votre parcours est : >>> Sexe > Morphologie (biologie) > Jean-Jacques Rousseau > Philosophie

L'exemple simulé dans l'image précédente est : **Sexe > Morphologie (biologie) > Jean-Jacques Rousseau > Philosophie**

### 3- Conclusion

---

Ce TP était très intéressant pour les étudiants de data science car il nous a permis de découvrir plusieurs concepts du développement web et aussi manipuler le Framework Flask pour créer des micro-services web avec python et les utiliser surtout pour déployer et mettre en production les modèles de machine learning conçus en python