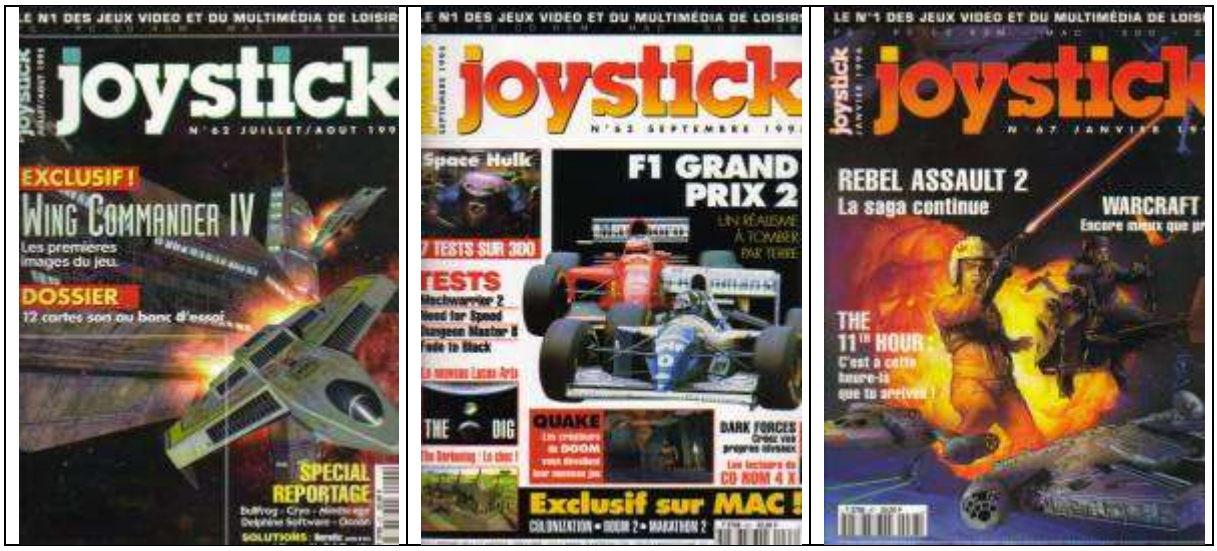
Moteur de recherche de retro-gaming : Boyer-Moore-Horspool



Joystick est un ancien magazine français de presse francophone spécialisé dans les jeux vidéo sur ordinateurs personnels. Paru initialement sous la forme d'un hebdomadaire en 1988 et 1989, il passe mensuel en 1990. Sa diffusion s'arrête en 2012. Il constitue aujourd'hui une formidable source d'informations historiques sur le jeu vidéo. Une grande partie de ses numéros sont disponibles sous la forme de scans JPG sur le site Abandonware-magazine : https://www.abandonware-magazines.org/affiche_mag.php?mag=30&page=presentation

Nous allons réaliser un moteur de recherche, basé sur le texte de chaque page obtenu par OCRisation, sous la forme d'un (lourd) fichier texte.

Nous aborderons différentes méthodes de recherche dans un texte, afin d'optimiser les temps d'exécution sur de grandes quantités de texte.

Les ressources numériques connexes du projet sont disponibles sur : https://picassciences.com/2021/04/02/moteur-de-recherche-retro-gaming-methode-de-boyer-moore-horspool/

I. Les fonctions intégrées de recherche dans Python :

A l'aide d'une recherche internet, comment peut-on chercher un motif « ABC » dans une chaine de caractères « BABACB**ABC**AAABBABAB » en utilisant des fonctions intégrées à Python, renvoyant la position du motif sur la chaine de caractères.

II. Approche naïve :

En informatique, une approche naïve, est l'approche la plus simple que l'on puisse implémenter pour résoudre un problème. Cela permet de poser une base que l'on peut ensuite améliorer.

Regarder la vidéo 1.

1. Appliquer le principe proposé, en écrivant une nouvelle ligne pour chaque étape, avec le motif suivant « RIE » dans la chaine suivante :

« UN GRAND GUERRIER ? PERSONNE PAR LA GUERRE NE DEVIENT GRAND. (M.Y) »

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
U	N		G	R	A	N	D		G	U	E	R	**R**	**I**	**E**	R			
R	I	E																	

2. Réaliser en Python un algorithme effectuant la tâche précédente. Créer une fonction prenant comme paramètres `Motif` et `Chaine`, qui retourne la position du `Motif` sur la `Chaine`. (S'il y a plusieurs fois le `Motif`, l'algorithme ne renvoie que la 1ère occurrence).

```
def RechercheNaiveSimple(Motif, Chaine):
    LongueurMotif = len(Motif)
    LongueurChaine = len(Chaine)

    # i parcours Chaine

    for i in range(0, …… ):
         # 2 lignes à ajouter
```

Indice : On peut utiliser de la découpe de chaine de caractère avec la notation Chaine[…. : ….]

3. Prendre le cas d'une chaine comportant plusieurs fois un motif. Améliorer votre algorithme pour retourne une **liste** de position. Tester la citation avec le motif « ERR » qui apparait deux fois.

III. Approche plus efficace : Algorithme de Boyer-Moore-Horspool:

Regarder la vidéo 2.

Appliquer le principe de Boyer-Moore-Horspool, en écrivant une nouvelle ligne pour chaque étape, avec le motif suivant « RIE » dans la chaine suivante : « UN GRAND GUERRIER ? PERSONNE PAR LA GUERRE NE DEVIENT GRAND. (M.Y) »

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
U	N		G	R	A	N	D		G	U	E	R	**R**	**I**	**E**	R			
R	I	E																	

III.1 : Pré-traitement du Motif

Ecrire en Python l'algorithme de pré-traitement du `Motif`, avec une fonction qui prend comme paramètre la chaine de caractère « `Motif` »

La fonction `PreTraitement` retourne un **dictionnaire** contenant les lettres du Motif et les indices de décalages. Voici des exemples de résultats de cette fonction

Motif	Résultat de `PreTraitement`
« mot »	{'m': 2, 'o': 1}
« yoda »	{'y': 3, 'o': 2, 'd': 1}
« chimie »	{'c': 5, 'h': 4, 'i': 1, 'm': 2}
« physique »	{'p': 7, 'h': 6, 'y': 5, 's': 4, 'i': 3, 'q': 2, 'u': 1}

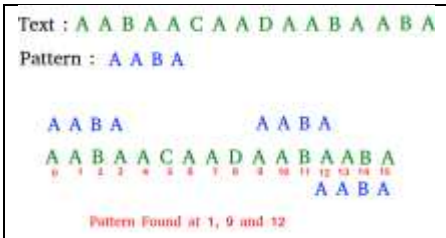
```
def PreTraitement(Motif):
   dictionnaire = {}
   for i in range(

   return dictionnaire
```

III.2 : Mise en œuvre de l'algorithme de Boyer-Moore-Horspool

Etape 1

Prouver la correspondance `Chaine` / `Motif` pour un indice de parcours de chaine.

	Dans cet exemple, on détaille le fonctionnement quand on arrive à 12. On regarde : si la lettre en 14 (12+2) correspond à la position 2 du motif si la lettre en 13 (12+1) correspond à la position 1 du motif si la lettre en 12 (12+0) correspond à la position 0 du motif SI on parvient à parcourir tout le Motif de la fin au début, il y a correspondance Chaine / Motif, on ajoute alors à une liste la position 12.

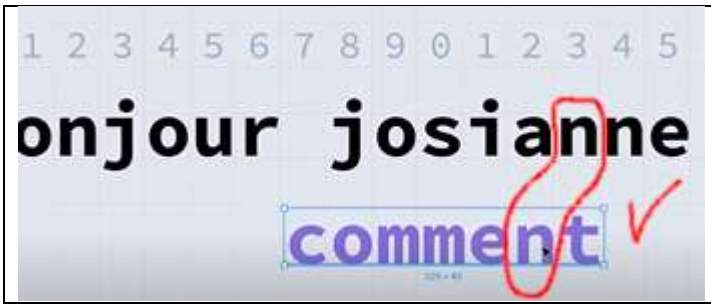
```
def Moore(Motif, Chaine):
   TablePreTraitement = PreTraitement(Motif)
   ListeRetournee=[]
   IndiceChaine = 0

   while IndiceChaine <= len(Chaine)-len(Motif):
      IndiceMotif = len(Motif)-1

      while Chaine[IndiceChaine+IndiceMotif] == ......... :
         if ..................... :
            ListeRetournee.append(IndiceChaine) # Motif trouvé
         IndiceMotif += -1
   IndiceChaine += 1
```

Etape 2

Gérer le décalage du Motif sur la Chaine :



On regarde si lettre au rang `IndiceChaine + len(Motif) - 1` soit 7 + 6 est comprise dans le dictionnaire de `PreTraitement`
Si non, on décale de la longueur du motif
Si oui, on utilise le dictionnaire de `PreTraitement` pour décaler.

```
def Moore(Motif, Chaine):
   TablePreTraitement = PreTraitement(Motif)
   ListeRetournee=[]
   IndiceChaine = 0

   while IndiceChaine <= len(Chaine)-len(Motif):
      IndiceMotif = len(Motif)-1

      while Chaine[IndiceChaine+IndiceMotif] == ..... :
         if ..... :
            ListeRetournee.append(IndiceChaine) # Motif trouvé
         IndiceMotif += -1

      if ..... :
         decalage = .....
      else :
         decalage = .....
      IndiceChaine += decalage

   return ListeRetournee
```

Vérifier votre fonction : Pour voir les décalages successifs de manière pédagogique, on peut ajouter avant le `IndiceChaine += decalage` la ligne suivante :

```
print(Chaine[IndiceChaine:], "\n", Motif, " -> ",decalage, sep = "")
```

Exemple :

Voir les vidéos 3a, 3b et 3c

```
print (Moore("GUER", "UN GRAND GUERRIER ? PERSONNE PAR LA GUERRE NE
DEVIENT GRAND."))
```

<u>On obtient</u>

```
UN GRAND GUERRIER ? PERSONNE PAR LA GUERRE NE DEVIENT GRAND.
GUER -> 3
GRAND GUERRIER ? PERSONNE PAR LA GUERRE NE DEVIENT GRAND.
GUER -> 4
D GUERRIER ? PERSONNE PAR LA GUERRE NE DEVIENT GRAND.
```

```
GUER -> 2
GUERRIER ? PERSONNE PAR LA GUERRE NE DEVIENT GRAND.
GUER -> 4
RIER ? PERSONNE PAR LA GUERRE NE DEVIENT GRAND.
GUER -> 4
_? PERSONNE PAR LA GUERRE NE DEVIENT GRAND.
GUER -> 4
ERSONNE PAR LA GUERRE NE DEVIENT GRAND.
GUER -> 4
NNE PAR LA GUERRE NE DEVIENT GRAND.
GUER -> 4
PAR LA GUERRE NE DEVIENT GRAND.
GUER -> 4
LA GUERRE NE DEVIENT GRAND.
GUER -> 3
GUERRE NE DEVIENT GRAND.
GUER -> 4
RE NE DEVIENT GRAND.
GUER -> 4
E DEVIENT GRAND.
GUER -> 1
DEVIENT GRAND.
GUER -> 4
IENT GRAND.
GUER -> 4
GRAND.
```

IV. Analyse des performances

Comparer les performances temporelles de la méthode Boyer-Moore-Horspool par rapport à la méthode naïve, sur l'extrait texte du magazine Joystick. Quelle est l'influence de la longueur du motif sur les performances ?

Ouvrir un fichier :

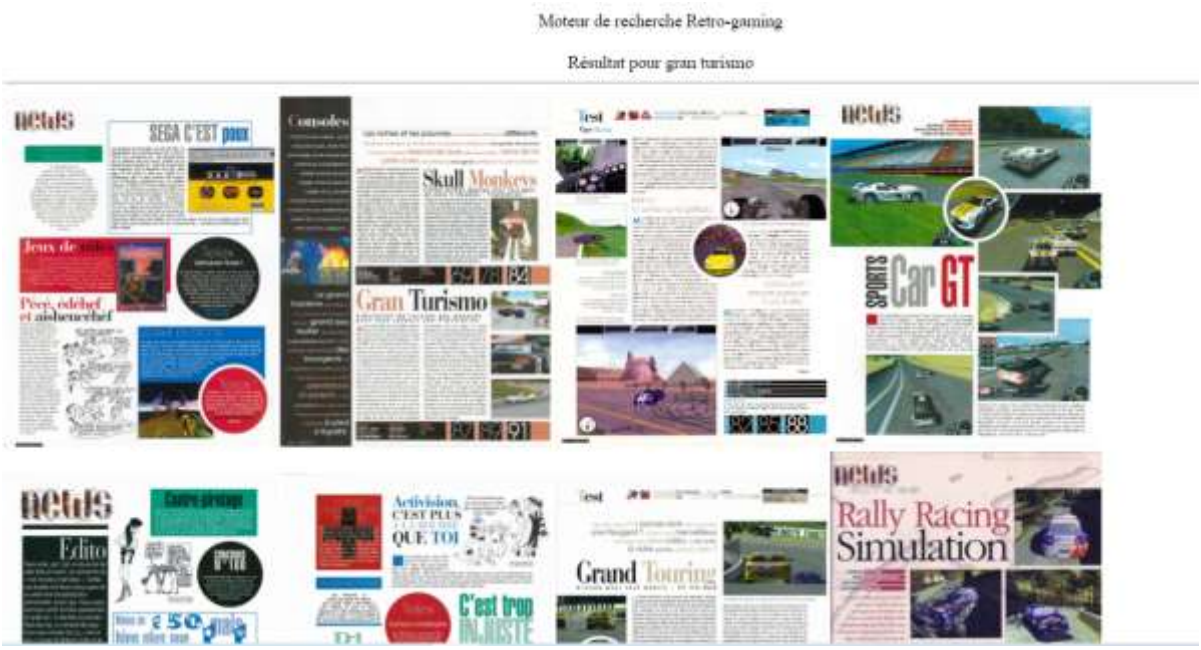
```
f = open("C:\\CHEMIN\\JoystickExtrait.txt", "r", encoding="utf8")
TexteComplet = f.read()
```

Mesurer une durée :

Utiliser `import time` et `time.time()` pour mesurer le temps.

V. Réalisation du moteur de recherche.

L'idée est de créer un script Python qui va générer un mur des pages correspondant à un mot clé, sous la forme d'une page HTML. Par exemple, le mot clé « gran turismo » sur le fichier `Joystick.txt` génère une page HTML avec le résultat suivant :



Etape 1 : Structure des données

Ouvrir dans Notepad++ le fichier `JoystickExtrait.txt`, afin de prendre connaissance de la structure des données. Vous trouverez les pages successives de chaque magazine, séparées par des blocs de séparations incluant le lien web vers l'image de la page.

```
CONTENU D'UNE PAGE N
<NOUVELLEPAGE>URL_PARTIELLE_DE_L_IMAGE_DE_LA_PAGE_N</NOUVELLEPAGE>
CONTENU D'UNE PAGE N+1
<NOUVELLEPAGE>URL_PARTIELLE_DE_L_IMAGE_DE_LA_PAGE_N+1</NOUVELLEPAGE>
CONTENU D'UNE PAGE N+2
<NOUVELLEPAGE>URL_DE_PARTIELLE_L_IMAGE_DE_LA_PAGE_N+2</NOUVELLEPAGE>
```

On utilise `JoystickExtrait.txt` pour commencer

Etape 2 : Recherche

Utiliser la méthode de Moore pour trouver les index des `Motifs` « star wars » *(avec un espace)* dans le fichier `JoystickExtrait.txt`

On doit obtenir la liste des positions suivantes `[1425948, 1709858, 1710293, 1743310, 1743613, 2448186, 2452622, 2936311, 3034042]`

Etape 3 : Repérer les pages du magazine.

On recherche, à partir de chaque position des `Motifs`, la position des balises `<NOUVELLEPAGE>` et `</NOUVELLEPAGE>` qui se trouvent forcement juste après chacun des `Motifs`.

Pour trouver la position des balises, on utilise toujours la méthode de Moore. On fera attention au fait que la fonction renvoie une liste alors que nous voulons ici uniquement la position de la première balise `<NOUVELLEPAGE>` et `</NOUVELLEPAGE>` après la position du `Motif` (inutile d'obtenir les listes des positions des balises jusqu'à la fin).

On doit obtenir dans l'ordre des colonnes suivantes :

i | Position des `Motifs` | Position de `<NOUVELLEPAGE>` | Position de `</NOUVELLEPAGE>`

```
0 1425948 1426194 1426265
1 1709858 1711255 1711326
2 1710293 1711255 1711326
3 1743310 1745408 1745479
4 1743613 1745408 1745479
5 2448186 2452091 2452161
6 2452622 2452994 2453064
7 2936311 2938882 2938952
8 3034042 3034413 3034483
```

Etape 4 : Affichage de la liste des pages du magazine

On extrait la liste des URL partielles comprises entre les balises `<NOUVELLEPAGE>` et `</NOUVELLEPAGE>`

On doit obtenir la liste des URL partielles suivantes :

ListeDesImages = ['Joystick/joystick_numero065/Joystick 065 - Page 092 (novembre 1995).jpg',
'Joystick/joystick_numero066/Joystick 066 - Page 009 (décembre 1995).jpg',
'Joystick/joystick_numero066/Joystick 066 - Page 009 (décembre 1995).jpg',
'Joystick/joystick_numero066/Joystick 066 - Page 022 (décembre 1995).jpg',
'Joystick/joystick_numero066/Joystick 066 - Page 022 (décembre 1995).jpg',
'Joystick/joystick_numero067/Joystick 067 - Page 059 (janvier 1996).jpg',
'Joystick/joystick_numero067/Joystick 067 - Page 061 (janvier 1996).jpg',
'Joystick/joystick_numero068/Joystick 068 - Page 058 (février 1996).jpg',
'Joystick/joystick_numero068/Joystick 068 - Page 090 (février 1996).jpg']

On ajoutera dans l'étape 5, un domaine devant ces URL partielles pour former des URL complètes.

Note : les pages qui apparaissent deux fois contiennent plusieurs occurrences du `Motif`. Vous pouvez si vous le désirez enlever les doublons avec `ListeDesImages = list(dict.fromkeys(ListeDesImages))`

Etape 5 : Création du mur d'images en HTML

Note : Vous remarquez qu'il manque le domaine devant chaque URL comme par exemple « `Joystick/joystick_numero068/Joystick 068 - Page 090 (février 1996).jpg` »)

Il faut donc coller au début de chaque URL le préfixe suivant :
http://download.abandonware.org/magazines/

Nous allons créer une page web HTML qui contient des balises "`<img src= http://download.abandonware.org/magazines/..... width=250>`" pour chacune des images. On sauvegarde l'ensemble dans un fichier HTML sur le disque.

Note : Afin de gagner en performance, on ne charge que les images à l'écran et pas l'ensemble des images de la page d'un seul coup. Ajoutez dans la balise img, le paramètre `loading="lazy"` *(ou* `loading=\"lazy\"`*)*

Conversion des URL

Vous remarquez que les URL contiennent des symboles spéciaux comme « N°2 », il faut donc convertir vos URL avec urllib.parse.quote(ListeDesImages[i]). (avec un `import urllib.parse` au préalable)

Il ne faut pas convertir le préfixe `http://download.abandonware.org/magazines/` mais uniquement la liste `ListeDesImages`.

Ecrire un fichier

```
f = open('C:\\CHEMIN\\index.html", "w")

f.write('Picassciences is awesome !')
f.close()
```

Aperçu du résultat final attendu

Vous disposez de vidéos d'aperçu du résultat final sur picassciences.