1 Quelques généralités sur la notion de recherche textuelle

Dans ce chapitre, on s'intéresse au problème de la recherche des **occurrences** d'une chaîne de caractères, que l'on appellera *motif*, dans une autre chaine de caractère, que l'on appellera *texte*. Par exemple, il y a deux occurrences du motif « bra » dans le texte « abracadabra ». Plus précisément, on va chercher à quelle positions dans le texte le motif apparaît. En numérotant les positions à partir de 0, on a une occurrence de « bra » à la position 1, et une autre à la position 8 (c'est à dire le neuvième caractère du texte). Le résultat de cette recherche est donc la liste $[1,8]$.

Notre objectif dans ce chapitre est d'écrire une fonction `recherche(motif, texte)` qui renvoie une liste des positions auxquelles sont les occurrences d'un motif dans un texte.

Si la liste renvoyée est vide, alors cela signifie qu'il n'y a pas d'occurrence du motif dans le texte. Concrètement nous allons mener cette recherche de deux manières : l'une « naïve », dans laquelle on fera une recherche systématique, et une autre « optimisée » en repérant des recherches d'occurrence qui ne sont pas nécessaires. Cette recherche nécessite de bien connaître les chaînes de caractères en Python.

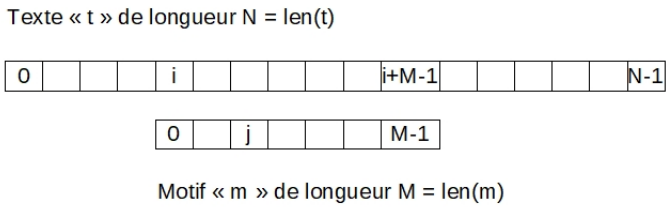
2 Rappels sur les chaînes de caractères

- Une chaîne de caractère peut être écrite au choix entre **apostrophes** ou entre **guillemets**. Ainsi on peut écrire indifféremment `'abracadabra'` ou `"abracadabra"`. Nous utiliserons la seconde forme, plus répandue dans les autres langages de programmation.
- La **longueur** d'une chaîne de caractère **s** est obtenue avec `len(s)`. Les caractères sont numérotés à partir de 0. Le $(i+1)$-ième caractère est obtenu avec `s[i]` pour i vérifiant $0 \leq i < len(s)$. Le premier caractère de **s** est donc `s[0]` et le dernier `s[len(s)-1]`.
- La **sous-chaîne** de **s** contenant les caractères de l'indice i inclus à l'indice j exclus est obtenue avec `s[i :j]`. Il s'agit d'une *nouvelle* chaîne de caractères, la chaîne **s** n'étant pas modifiée. D'une manière générale, les chaînes de caractères sont *immuables*.
- Les caractères et les chaînes peuvent être comparées avec l'opérateur « == ».

Dans tout ce chapitre, on note **m** le motif que l'on cherche et **t** le texte dans lequel on le recherche. On notera aussi à l'occasion **M** la longueur du motif **m** et **N** la longueur du texte **t**. On notera i une position dans le texte **t**, et j une position dans le motif **m**.

3 Recherche simple

Pour se poser la question d'une occurrence de **m** à la position i dans le texte **t**, il est pratique de visualiser le schéma ci-dessous :



On rappelle que dans cette partie, on va se limiter à une méthode de recherche simple, et donc systématique du motif à toutes les positions du texte. Nous allons décomposer la recherche en deux fonctions : l'une appelée **occurrence** qui dira si oui ou non il y a une occurrence du motif dans le texte à une position i, et une autre appelée **rechercheTexte**, qui à partir du texte et du motif renverra la liste des occurrences, en utilisant plusieurs fois la fonction « occurrence ».

3.1 la fonction « occurrence »

Proposer une implémentation de la fonction « occurrence »

```
texte = "abracadabri abracadabra"
mot = "abri"

def occurrence(m,t,i):
    '''
    une fonction qui renvoie renvoie True
    s'il y a une occurrence du mot m
    à la position i
    dans le texte t
    '''

assert(occurrence(mot, texte,0) == False)
assert(occurrence(mot, texte,2) == False)
assert(occurrence(mot, texte,7) == True)
assert(occurrence(mot, texte,15) == False)
```

Une fois écrite, votre fonction doit passer les **assert** indiqués...

Remarque : la recherche d'une occurrence à un indice i peut se faire en partant de i ou alors à rebours, en partant de $i+M-1$. Même si cela peut paraître équivalent, on verra en fait que la seconde option est statistiquement plus efficace.

3.2 la fonction « recherche »

Proposer une implémentation de la fonction « rechercheTexte »

```
def rechercheTexte(m,t):
    '''
    une fonction qui renvoie
    la liste des occurrences
    du mot t dans le texte t
    en utilisant la fonction occurrence
    '''


resultat = rechercheTexte(mot, texte)
print(resultat)
print(len(resultat))
```

Après écriture, vérifier que le résultat de la recherche donne simplement comme liste [7], car la seule occurrence de "abri" dans "abracadabri abracadabra" se produit pour $i = 7$, soit à la huitième position.

3.3 Pour anticiper ce qui va suivre...

Nous avons qualifié cette méthode de recherche de « simple » ou « naïve ». En effet, une fois que l'on a testé une occurrence à un indice i dans le texte, on va systématiquement ensuite aller tester l'occurrence à l'indice $i+1$, sans tenir compte de ce que l'on observe à l'indice i... c'est dommage!

Sauriez-vous détecter des situations où il serait inutile de vérifier une ou plusieurs occurrences suivantes de i ?