Taper help(str.find) dans une console Python

L’objectif des questions suivantes est de programmer des fonctions de recherche textuelle sans bien sûr utiliser la méthode str.find déjà existante.

# Coder une fonction *identique*

Dans un fichier nommé *algoRechercheNaive.py*, écrire la fonction *identique(texte: str, motif: str, i: int) :*

La fonction teste si le motif de longueur k est présent dans le texte à la position i, c’est-à-dire si *texte[i:i+k] == motif*, en comparant les caractères de *texte[i:i+k]* et de *motif* un par un de gauche à droite jusqu’à trouver un caractère différent.

Cette fonction identique renvoie un booléen : True si le motif est présent à cette position du texte, et False sinon.

Exemple : *identique('Il fait beau', 'fa', 3) --> True*

*identique('Il fait beau', 'fa', 0) --> False*

*identique('Il fait beau', 'fa', 6) --> False*

# Coder l’algorithme naïf de recherche textuelle

Écrire la fonction *recherche\_naive(texte: str, motif: str):*

Cette fonction recherche la chaine appelée motif dans la chaine appelée texte de façon naïve et **renvoie un entier**, égal à la position du premier caractère du motif dans la chaine texte si la recherche aboutit, et à -1 dans le cas contraire.

La comparaison s’effectue à l’aide de la fonction *identique*

Exemple : *recherche\_naive('Il fait be****au*** *et ch****au****d', 'au') --> 10*

*recherche\_naive('Il fait be****au*** *et ch****au****d', 'fa') --> 3*

*recherche\_naive('Il fait be****au*** *et ch****au****d', 'zo') --> -1*

# Algorithme naïf de recherche textuelle version 2

Écrire une fonction recherche\_naive\_v2(texte: str, motif: str):

Cette fonction est similaire à la fonction recherche\_naive mais ne fait pas appel à la fonction identique (la comparaison du motif étant inclus dans la fonction recherche\_naive\_v2).

Exemple : *recherche\_naive\_v2('Il fait be****au*** *et ch****au****d', 'au') --> 10*

*recherche\_naive\_v2('Il fait be****au*** *et ch****au****d', 'fa') --> 3*

*recherche\_naive\_v2('Il fait be****au*** *et ch****au****d', 'zo') --> -1*

Remarque : Vous pouvez utiliser un break pour sortir prématurément d’une boucle for, pour les plus courageux vous pouvez vous en passer.

# Compteur de comparaisons et nombre d’occurrences du motif dans le texte

Écrire une fonction recherche\_naive\_v3(texte: str, motif: str):

Cette fonction devra renvoyer le nombre de comparaisons effectuées, ainsi que le nombre d’occurrences du motif dans le texte avec leurs positions sous forme d’une liste.

Exemple 1:

texte = "Il y a deux histoires : l'histoire officielle, menteuse, qu’on enseigne, puis l'histoire secrète, où sont les véritables causes des événements."

motif = "histoire"

print(recherche\_naive\_v3(texte,motif))

-----> 157, 3, [12, 26, 80]

Ce qui veut dire 157comparaisons, 3 occurrences du motif trouvées, positions des motifs trouvés dans le texte : 12, 26, 80

Exemple 2:

texte = "Rien de grand ne s'est accompli dans le monde sans passion"

motif = "savant"

print(recherche\_naive\_v3(texte,motif))

-----> 61, 0, []

Ce qui veut dire 61 comparaisons, 0 occurrence du motif trouvée, pas de positions de motifs trouvés dans le texte