ISM MPSI/PCSI, Informatique
Année 2024/2025

Corrigé du TP Informatique 10

Exercice 1

2.(c) La valeur limite de `k` pour le test `texte[k:k+m]==mot` est `n-m`. En effet, au delà de cette valeur, la chaîne extraite est de taille strictement inférieure à `mot` et le résultat du test est donc nécessairement `False`.

Exercice 2

1. On saisit :

```
def rech1(mot,texte):
    m=len(mot)
    n=len(texte)
    for k in range(n-m+1):
        if mot==texte[k:k+m]:
            return True
    return False
```

2. On saisit :

```
def rech2(mot,texte):
    m=len(mot)
    n=len(texte)
    k=0
    while k<=n-m and mot!=texte[k:k+m]:
        k+=1
    return k<=n-m
```

Exercice 3

1. On saisit :

```
def detect3(elt,T,pos):
    """detect3(elt:str,T:str,pos:int)->bool
    Renvoie le test elt==T[pos:pos+m] avec m=len(elt)"""
    for k in range(len(elt)):
        if elt[k]!=T[pos+k]:
            return False
    return True
```

2. On saisit :

```
def rech3(mot,texte):
    """rech3(mot:str,texte:str)->bool
    Renvoie le test de présence de mot dans texte"""
    m=len(mot)
    n=len(texte)
    for k in range(n-m+1):
        if detect(mot,texte,k):
            return True
    return False
```

3. L'intérêt de procéder sans slicing est d'éviter la création de nouvelles chaînes lors de chaque extraction. On peut l'observer clairement en rendant la fonction `detect3` bavarde :

```
def detect3(elt,T,pos):
    print(id(T))
    for k in range(len(elt)):
        if elt[k]!=T[pos+k]:
            return False
    return True
```

L'expérimentation donne :

```
>>> rech3("jour","bonjour")
92621432
...
92621432
True
```

On constate que c'est le même identifiant pour la chaîne transmise en argument, autrement dit il n'y a pas de création de nouvelles chaînes.

Exercice 4

```
def str_egal(x,y):
    for k in range(len(x)):
        if x[k]!=y[k]:
            return False
    return True
```

```
def rech4(mot,texte):
    m=len(mot)
    n=len(texte)
    for k in range(n-m+1):
        if str_egal(mot,texte[k:k+m]):
            return True
    return False
```

Exercice 5

1. On saisit :

```
def detect5(elt,T,pos):
    """detect5(elt:str,T:str,pos:int)->bool
    Renvoie le test elt==T[pos:pos+m] avec m=len(elt)"""
    k=0
    m=len(elt)
    while k<m and elt[k]==T[k+pos]:
        k+=1
    return k==m
```

2. On saisit :

```
def rech5(mot,texte):
    """rech5(mot:str,texte:str)->bool
    Renvoie le test de présence de mot dans texte"""
    m=len(mot)
    n=len(texte)
    k=0
    while k<=n-m and not detect5(mot,texte,k):
        k+=1
    return k<=n-m
```

Exercice 6

On saisit :

```
def remp(mot1,mot2,texte):
    """remp(mot1:str,mot2:str,texte:str)->str
    Remplace dans texte toutes les occurrences de mot1 par mot2"""
    res=""
    m=len(mot1)
    n=len(texte)
    k=0
    while k<=n-1:
        if k<=n-m and detect3(mot1,texte,k):
            res+=mot2
            k+=m
        else:
            res+=texte[k]
            k+=1
    return res
```

Exercice 7

On saisit :

```
def pos(mot1,mot2):
    """pos(mot1:str,mot2:str)->int
    Renvoie :
    * 0 si mot1==mot2
    * 1 si mot1 après mot2 dans le dictionnaire
    * -1 si mot1 sinon"""
    n1=len(mot1)
    n2=len(mot2)
    k=0
    diff=False
    while k<n1 and k<n2 and not diff:
        diff=mot1[k]!=mot2[k]
        k+=1
    if not diff:
        if n1>n2:
            return 1
        elif n1==n2:
            return 0
        else:
            return -1
    else:
        k-=1
        if mot1[k]>mot2[k]:
            return 1
        else:
            return -1
```