TP 13 : Chaînes de caractères

1 Applications directes

Exercice 1. On considère deux chaînes de caractères *txt* et *mot* et on veut déterminer à quelles positions on retrouve *mot* dans *txt* en convenant qu'un point dans *mot* peut désigner n'importe quel caractère dans *txt*. Par exemple si :

$$txt = \texttt{"abcabbab"}$$
$$mot = \texttt{"ab."}$$

alors *mot* apparait dans *txt* aux positions 0 et 3. Adapter la fonction **`positions(txt,mot)`** vue en cours pour qu'elle réponde à ce problème. Vérifier son fonctionnement sur quelques exemples pertinents.

2 Mise en œuvre

Exercice 2 *Mastermind*. On veut programmer un jeu de *Mastermind* qui se déroulera de la façon suivante :

- L'ordinateur choisit de manière aléatoire une chaine de 4 lettres distinctes parmi les lettres a, b, c, d, e, f, g, h et la garde secrète ;
- L'ordinateur demande à l'utilisateur d'entrer une chaine de 4 lettres et donne le nombre de lettres apparaissant dans la séquence secrète et bien placées et le nombre de lettres apparaissant dans la séquence secrète et mal placées ;
- On recommence cette étape tant que l'utilisateur n'a pas trouvé quelle est la séquence secrète.

On rappelle que suivant les règles usuelles du *Mastermind*, les séquences utilisées ne peuvent pas comporter deux fois la même lettre. On commencera le programme par le code suivant qui sert entre autres à définir aléatoirement la combinaison secrète :

```
from random import randint

autorises = "abcdefgh"

def combinaison_aleatoire():
    s = ""
    while len(s)<4:
        c = autorises[randint(0,7)]
        if c not in s:
            s = s+c
    return s
```

On rappelle que si **c** est un caractère et **s** une chaine, l'instruction **c in s** renvoie **True** si **c** apparait dans **s** et **False** sinon.

(a) Écrire une fonction **saisie_valide(s)** qui renvoie **True** si **s** est une chaine de 4 caractères parmi a,b,...,h et tous distincts.

(b) Écrire une fonction **saisie()** qui demande à l'utilisateur d'entrer une combinaison et repose la question jusqu'à ce que la combinaison entrée soit valide (au sens précédent).

(c) Écrire une fonction **bien_places(s1,s2)** qui compte le nombre de caractères de la chaine **s1** qui se retrouvent à la même place dans la chaine **s2** (on pourra supposer que **s1** et **s2** sont des chaines de caractères de longueur 4).

(d) Écrire une fonction **mal_places(s1,s2)** qui compte le nombre de caractères de la chaine **s1** qui apparaissent dans **s2** mais pas à la même place (là encore, on pourra supposer que **s1** et **s2** sont des chaines de caractères de longueur 4).

(e) Écrire une fonction **jeu()** qui définit une chaine de caractères **secrete** en utilisant la fonction **combinaison_aleatoire**, appelle la fonction **saisie**, affiche le nombre de caractères bien placé ainsi que le nombre de caractères mal placés et répète tous ceci tant que la saisie de l'utilisateur ne correspond pas à la combinaison secrète.

(f) Jouer en lançant le programme avec *Ctrl+F5* et en tapant **jeu()** dans la console.

(g) Modifier le programme pour que l'on puisse arrêter le jeu en tapant **abandon** lorsque l'ordinateur demande d'entrer une combinaison. L'ordinateur doit alors afficher la bonne combinaison.

Corrections

Ex 1.

```
def positions(txt,mot):
    """
    La liste des positions où mot apparait dans txt
    mot est une chaine de caractères dans laquelle '.' désigne
    un caractère quelconque
    """
    Pos=[]
    for i in range(len(txt)-len(mot)+1):
        nb_egaux=0
        for j in range(len(mot)):
            if mot[j]=="." or mot[j]==txt[i+j]:
                nb_egaux=nb_egaux+1
        if nb_egaux==len(mot):
            Pos.append(i)
    return Pos
print(positions("abcabbac","ab."))
```

```
[0, 3]
```

```
from random import randint

autorises = "abcdefgh"

def combinaison_aleatoire():
    s = ""
    while len(s)<4:
        c = autorises[randint(0,7)]
        if c not in s:
            s = s+c
    return s
```

```
def saisie_valide(s):
    if s=="abandon":
        return True
    if len(s)!=4:
        return False
    for i in range(0,len(s)):
        if s[i] not in autorises:
            return False
    for i in range(0,len(s)-1):
        for j in range(i+1,len(s)):
            if s[i]==s[j]:
                return False
    return True
```

```
def saisie():
    s = input("Entrez une combinaison : ")
    while not saisie_valide(s):
        print("Uniquement 4 lettres distinctes parmi",autorises)
        s = input("Entrez une combinaison : ")
    return s
```

```
def bien_places(s1,s2):
    cpt = 0
    for i in range(len(s1)):
        if s1[i]==s2[i]:
            cpt = cpt+1
    return cpt
```

```
def mal_places(s1,s2):
    cpt = 0
    for i in range(len(s1)):
        if s1[i]!=s2[i] and s1[i] in s2:
            # Si le caractère s1[i] apparait dans s2 mais pas en s2[i]
            # c'est nécessairement qu'il est mal placé
            # car tous les caractères sont distincts
            cpt = cpt+1
    return cpt
```

```
def jeu():
    combinaison = combinaison_aleatoire()
    s = saisie()
    while s!=combinaison and s!="abandon":
        print("Bien placés :",bien_places(s,combinaison))
        print("Mal placés :",mal_places(s,combinaison))
        s = saisie()
    if s=="abandon":
        print("La bonne combinaison était",combinaison)
    else:
        print("Gagné !")
```