TD 7 -
Listes et chaînes de caractères.

Informatique
MPSI/PCSI - Lycée Thiers

Exercice 1 - Enoncé

I - Algorithmes de tri : Un algorithme de Tri prend en paramètre une liste de nombre et l'ordonne dans le sens croissant (ou décroissant).

Exercice 1. *Le Tri à bulle.*

Un exemple d'algorithme de est le *Tri à bulle*. Il s'opère de la façon suivante :

- Parcourir les éléments du tableau de gauche à droite.
- Dès que l'on rencontre deux éléments consécutifs qui ne sont pas dans le bon ordre, on échange leur position. C'est à dire :
  `SI tableau[i] > tableau[i+1]`
  `ALORS : échanger tableau[i] et tableau[i+1]`
- Recommencer tant que l'on a changé quelque chose.

Ecrire une fonction `TriBulle` en `python` qui prend en paramètre une liste de nombres et retourne la liste triée par l'algorithme du tri à bulle.

Exercice 1 - Correction

```
# Tri a bulle
def triBulle(L):
    n = len(L)
    chgt = True
    while chgt:
        chgt = False
        for k in range(n-1):
            if L[k] > L[k+1]:
                L[k], L[k+1] = L[k+1], L[k]
                chgt = True
        n = n-1
    return L
```

Après le k-ième passage dans la boucle `for` les k derniers éléments sont à leur place définitive. C'est pourquoi après chaque boucle `for` on décrémente `n`.

Exercice 2 - Enoncé

Exercice 2. *Le Tri par sélection.* Un autre exemple d'algorithme de tri est le tri par sélection. il s'opère de la façon suivante :

- Chercher l'indice de l'élément maximal de `L`,
- échanger dans `L` l'élément maximal avec le dernier élément de `L`,
- Recommencer le même procédé pour les éléments de `L` allant du premier à l'avant-dernier,
- et ainsi de suite...

1. Ecrire une fonction `echange(L,i,j)` prenant en paramètre une liste `L` et deux entiers positifs `i`, `j`, et qui échange dans la liste `L` ses éléments d'indice `i` et `j`.
2. Ecrire une fonction `IndiceMax(L,k)` qui prend en paramètre une liste `L` de nombres et un entier positif `k` et qui retourne l'indice de l'élément maximal dans la sous-liste de `L` allant de l'indice `0` à l'indice `k`.
3. En appelant les fonctions précédentes, écrire une fonction `triSelection(L)` prenant en paramètre une liste de nombres et qui les ordonne dans le sens croissant en appliquant un tri par sélection.

Exercice 2 - Corrigé

1.

```
def echange(L,i,j):
    L[i], L[j] = L[j], L[i]
```

2.

```
def indiceMax(L,k):
    ind = 0
    for i in range(1,k+1):
        if L[i] > L[ind]:
        ind = i
    return ind
```

3.

```
def triSelection(L):
    n = len(L)
    for k in range(n-1,0,-1):
        i = indiceMax(L,k)
        echange(L,i,k)
    return L
```

Exercice 3 - Enoncé

Exercice 3. *Épreuve type ; Oral - Banque PT.*

Soit n un entier vérifiant $n \leqslant 26$. On souhaite écrire un programme qui code un mot en décalant chaque lettre de l'alphabet de n lettres. Par exemple pour $n = 3$, le décalage sera le suivant :

avant décalage	a	b	c	d	...	...	x	y	z
après décalage	d	e	f	g	...	...	a	b	c

Le mot `oralensam` devient ainsi `rudohqvdp`.

1. Définir une chaîne de caractères contenant toutes les lettres dans l'ordre alphabétique (caractères en minuscule).
2. Écrire une fonction `decalage`, d'argument un entier `n`, renvoyant une chaîne de caractères contenant toutes les lettres dans l'ordre alphabétique, décalées de `n`, comme indiqué ci-dessus.
3. Écrire une fonction indices, d'arguments un caractère `x` et une chaîne de caractères `phrase`, renvoyant une liste contenant les indices de `x` dans `phrase` si `x` est une lettre de phrase et une liste vide sinon.
4. Écrire une fonction codage d'arguments un entier `n` et une chaîne de caractères `phrase`, renvoyant `phrase` codé avec un décalage de n lettres.
5. Comment peut-on décoder un mot codé ?

Exercice 3 - Corrigé

1. variable `alphabet`.

```
alphabet = 'abcdefghijklmnopqrstuvwxyz'
```

2. Fonction `decalage`.

```
def decalage(n):
    return alphabet[n:] + alphabet[:n]
```

3. Fonction `indices`.

```
def indices(x,phrase):
    n = len(phrase)
    L = []
    for i in range(n):
        if x == phrase[i]:
            L.append(i)
    return L
```

4.

```
def codage(n,phrase):
    alphabetCode = decalage(n) # Obtention de l'alphabet codé
    # Constitution de la liste des caractères de phrase
    listePhrase = []
    for x in phrase:
        listePhrase.append(x)
    # Codage de la liste en utilisant la fonction indices(,)
    for i in range(26):
        lettre = alphabet[i]
        lettreCode = alphabetCode[i]
        ind = indices(lettre,phrase)
        for i in ind:
            listePhrase[i] = lettreCode
    # Obtention de la chaine codée :
    phraseCode = ''
    for lettre in listePhrase :
        phraseCode += lettre
    return phraseCode
```

5. Pour décoder un message crypté avec un décalage de `n`, il suffit de coder avec décalage `-n` (la fonction `decalage` marche aussi avec un argument négatif).

```
def decodage(n, phrasecode):
    return codage(-n, phrasecode)
```