Informatique PCSI
Corrigé suite TP 1

1 Corrigé des exercices

1.1 Exercice 1

```
def indice_maximum(liste):
    if liste == []:
        return None
    else:
        indice = 0
        maxi = liste[0]
        for i in range(len(liste)):
            if liste[i] > maxi:
                indice = i
                maxi = liste[i]
    return maxi, indice
```

Remarque : si on remplace la ligne `if liste[i] > maxi` par la ligne `if liste[i] >= maxi`, on obtient l'indice le plus grand.

1.2 Exercice 2

On commence par chercher le maximum puis on construit la liste des indices des éléments égaux au maximum.

```
def indices_maximum(liste):
    maxi = liste[0]
    for x in liste:
        if x > maxi:
            maxi = x
    indices = []
    for i in range(len(liste)):
        if liste[i] == maxi:
            indices.append(i)
    return maxi, indices
```

1.3 Exercice 3

La fonction `mystere` fusionne les deux listes passées en paramètres en respectant l'ordre, tant que tous les éléments d'une des deux listes n'ont pas été choisis. On choisit à chaque passage dans la boucle le plus petit élément parmi les deux éléments comparés. En cas d'égalité, on choisit l'élément de `liste2`.

Donc l'appel `mystere([2, 5, 6, 8], [1, 4, 7, 8, 9])` renvoie
la liste `[1, 2, 4, 5, 6, 7, 8, 8]`.

1.4 Exercice 4

```
def nombre_de_e(chaine):
    cpt = 0
    for car in chaine:
        if car == 'e':
            cpt = cpt + 1
    return cpt
```

Nous testons cette fonction dans l'interpréteur :

```
>>> nombre_de_e("Le soleil brille")
3
>>> nombre_de_e("Combien de e dans cette chaîne de caractères ?")
8
```

Attention, cette fonction ne compte que les 'e' pas les 'é', les 'è', les 'ê'.

1.5 Exercice 5

```
def compte(mot):
    lettres = {}
    for c in mot:
        if c in lettres:
            lettres[c] = lettres[c] + 1
        else:
            lettres[c] = 1
    return lettres
```

Une instruction comme `lettres[c] = lettres[c] + 1` provoquerait une erreur si la clé `c` n'existe pas, ce qui serait le cas dès la première lettre. On teste donc pour chaque lettre l'appartenance aux clés du dictionnaire.

1.6 Exercice 6

```
def stat(texte):
    lettres = {}
    for c in texte:
        if c not in " ,;:!?.":
            if c in lettres:
                lettres[c] = lettres[c] + 1
            else:
                lettres[c] = 1
    return lettres
```

Test sur un exemple :

```
>>> stat("MON PROGRAMME FONCTIONNE, JE SUIS CONTENT !")
{'M': 3, 'O': 5, 'N': 6, 'P': 1, 'R': 2, 'G': 1, 'A': 1, 'E': 4,
'F': 1, 'C': 2, 'T': 3, 'I': 2, 'J': 1, 'S': 2, 'U': 1}
```

On peut trier le résultat avec différentes instructions.
Test sur un exemple :

```
>>> d = stat("MON PROGRAMME FONCTIONNE, JE SUIS CONTENT !")
>>> sorted(d)
['A', 'C', 'E', 'F', 'G', 'I', 'J', 'M', 'N', 'O', 'P', 'R', 'S',
'T', 'U']
>>> sorted(d.items())
[('A', 1), ('C', 2), ('E', 4), ('F', 1), ('G', 1), ('I', 2), ('J', 1),
('M', 3), ('N', 6), ('O', 5), ('P', 1), ('R', 2), ('S', 2), ('T', 3),
('U', 1)]
```

1.7 Exercice 7

```
def distances(mot):
    n = len(mot)
    d = {}
    for i in range(n):
        d[mot[i]] = n - 1 - i
    return d
```