Séance 3 : Boucles for et séquences

L1 – Université Côte d'Azur

Exercice 1 — Calcul de la fréquence des mots d'un texte

Pour chaque fonction de l'exercice, écrire des tests.

1. Créez une chaîne `alphabet="abcdefghijklmnopqrstuvwxyz"` contenant toutes les lettres de l'alphabet, dans l'ordre et en minuscule. Vérifiez avec `assert` que sa taille est bien de 26.
2. Écrivez une fonction `indice(lettre)` qui donne l'indice d'un caractère dans la chaîne `alphabet`. On renverra 26 si la lettre n'est pas dans l'alphabet. Ainsi, `indice('a')` vaut 0 et `indice('z')` vaut 25
3. Écriver une fonction `statistiques(texte)` qui renvoie un tableau de taille 27 à partir d'un texte non vide. Ce tableau contient dans chaque case d'indice `i` le nombre de fois où la lettre d'indice `i` est apparue dans le texte.

```
>>> tableau = statistiques("salut à toi mon ami")
>>> tableau[0] # nombre de a (« à » ne compte pas)
2
>>> tableau[1] # nombre de b
0
>>> tableau[26] # nombre d'autres symboles (4 espaces et le « à »)
5
```

4. Écrire une fonction `affichage(tableau)` qui affiche un version lisible du tableau, pour chaque case non vide, on affichera la lettre ainsi que son nombre d'occurence.

```
>>> affichage(tableau)
a : 2
i : 2
l : 1
m : 2
n : 1
o : 2
s : 1
t : 2
u : 1
Autres : 5
```

5. Écrire une fonction `plus_fréquent(tableau)` qui renvoie la lettre la plus fréquente du texte. Si deux lettres ont la même fréquence, on choisira la première dans l'ordre alphabétique.

```
>>> plus_fréquent(tableau)
'a'
```

6. Comment modifier la fonction pour renvoyer, dans le cas où plusieurs lettres ont la même fréquence, la dernière dans l'ordre alphabétique ? On appelera cette nouvelle variante `plus_fréquent_bis(texte)`.

```
>>> plus_fréquent_bis(tableau)
't'
```

7. Enfin, écrivez une fonction, `tous_les_plus_fréquents(texte)` qui renvoie la chaîne contenant toutes les lettres de l'alphabet ayant la fréquence maximale.

```
>>> tous_les_plus_fréquents(tableau)
'aimot'
```

Exercice 2 — Concaténer

Écrivez et tester une fonction `concatenation(L1,L2)` qui à partir de deux listes `L1` et `L2` renvoie la liste `L1+L2`. On ne pourra pas utiliser l'opération + de Python. Il faudra créer un tableau de la bonne taille avant de le remplir.

```
# Première approche. On crée un variable k pour parcourir la liste L
def concatener(L1,L2):
    n = len(L1) + len(L2)
    L = [0] * n

    k=0 # indice dans L
    for i in range(len(L1)):
        L[k] = L1[i]
        k = k+1

    for i in range(len(L2)):
        L[k] = L2[i]
        k = k+1
    return L

# Deuxième approche. On ne crée pas de variable k mais on trouve une
# formule (i + len(L1) pour le décalage de l'indice dans L à partir de
# l'indice de L2.

def concatener_bis(L1,L2):
    n = len(L1) + len(L2)
    L = [0] * n

    for i in range(len(L1)):
        L[i] = L1[i]

    for i in range(len(L2)):
        L[i+ len(L1)] = L2[i]
    return L

assert concatener([1,2,3], [4,5,6,7]) == [1,2,3] + [4,5,6,7]
assert concatener_bis([1,2,3], [4,5,6,7]) == [1,2,3] + [4,5,6,7]
```

Exercice 3 — Tables de vérité

1. Écrivez une fonction `car` qui transforme chaque booléen en caractère. On aura, `car(True)=='T'` et `car(False)=='F'`.

```
def car(b):
    if b:
        return "T"
    else:
        return "F"
```

2. Écrivez `table_2_vérité` qui prend une fonction booléenne à deux arguments et trace sa table de vérité. On utilisera **obligatoirement** des boucles `for` sur la liste `L = [True, False]`.

```
def formule(a,b):
    return not (a and b)
```

```
>>> tables_2_vérité(formule)
a b P
T T F
T F T
F T T
F F T
```

```
def tables_2_vérité(formule):
    booléens = [True, False]
    print("a b P")
    for i in range(len(booléens)):
        for j in range(len(booléens)):
            a = booléens[i]
            b = booléens[j]
            print(car(a),car(b), car(formule(a,b)))
```

3. Même question avec `table_3_vérité` qui prend des fonctions booléenes à trois arguments. Testez sur la formule du TD 0 :`(a or not c) and (not a or b) and (not b or c)`.

```
def tables_3_vérité(formule):
    booléens = [True, False]
    print("a b c P")
    for i in range(len(booléens)):
        for j in range(len(booléens)):
            for k in range(len(booléens)):
                a = booléens[i]
                b = booléens[j]
                c = booléens[k]
                print(car(a),car(b), car(c), car(formule(a,b,c)))

def formule(a,b,c):
    return (a or not c) and (not a or b) and (not b or c)
```

```
>>> tables_3_vérité(formule)
a b c P
T T T T
T T F F
T F T F
T F F F
F T T F
F T F F
F F T F
F F F T
```

4. Démontrer à la main des égalités mathématiques, c'est dépassé. Écrivez une fonction `égalité_formules_2(f1,f2)` qui prend des fonctions booléennes à deux arguments et revoie `True` si elles sont égales (et `False` sinon).

```
def égalité(f1,f2):
    booléens = [True, False]
    for i in range(len(booléens)):
        for j in range(len(booléens)):
            a = booléens[i]
            b = booléens[j]
            if f1(a,b) != f2(a,b):
                return False
    return True
```

5. Utilisez votre fonction pour démontrer les lois de Morgan de manière automatique.
`not (a and b) = not a or not b` ainsi que `not (a or b) = not a and not b`

```python
def f1(a,b):
    return not (a and b)

def f2(a,b):
    return not a or not b

def g1(a,b):
    return not (a or b)

def g2(a,b):
    return not a and not b

assert égalité(f1,f2)
assert égalité(g1,g2)
```

Exercice 4 — Print Proust

Écrivez une fonction `justification_à_gauche(s,n)` qui prend une chaîne de caractères `s` contenant une phrase (sans symbole `'\n'`) et qui l'affiche en mettant au plus `n` caractères par ligne, en justifiant à gauche (si un mot est trop grand il sera seul sur sa ligne).

```
>>> justification_à_gauche("Mais au lieu de la simplicité, c'est le faste que…" , 12)
Mais au lieu
de la
simplicité,
c'est le
faste que…
```

Testez votre programme en affichant cette phrase [1] de Proust en entier [2] sur 80 caractères par ligne.

Mais au lieu de la simplicité, c'est le faste que je mettais au plus haut rang, si, après que j'avais forcé Françoise, qui n'en pouvait plus et disait que les jambes « lui rentraient », à faire les cent pas pendant une heure, je voyais enfin, débouchant de l'allée qui vient de la Porte Dauphine – image pour moi d'un prestige royal, d'une arrivée souveraine telle qu'aucune reine véritable n'a pu m'en donner l'impression dans la suite, parce que j'avais de leur pouvoir une notion moins vague et plus expérimentale – emportée par le vol de deux chevaux ardents, minces et contournés comme on en voit dans les dessins de Constantin Guys, portant établi sur son siège un énorme cocher fourré comme un cosaque, à côté d'un petit groom rappelant le « tigre » de « feu Baudenord », je voyais – ou plutôt je sentais imprimer sa forme dans mon coeur par une nette et épuisante blessure – une incomparable victoria, à dessein un peu haute et laissant passer à travers son luxe « dernier cri » des allusions aux formes anciennes, au fond de laquelle reposait avec abandon Mme Swann, ses cheveux maintenant blonds avec une seule mèche grise ceints d'un mince bandeau de fleurs, le plus souvent des violettes, d'où descendaient de longs voiles, à la main une ombrelle mauve, aux lèvres un sourire ambigu où je ne voyais que la bienveillance d'une Majesté et où il y avait surtout la provocation de la cocotte, et qu'elle inclinait avec douceur sur les personnes qui la saluaient.

1. oui, c'est une seule phrase !
2. Si nécessaire le texte est disponible à l'adresse : https://upinfo.univ-cotedazur.fr/~obaldellon/L1/py/tp3/marcel-proust.html

```python
def justification_à_gauche(s,n) :
    mot = ''
    ligne = ''
    taille_ligne = 0
    taille_mot = 0
    for i in range(len(s)) :
        if s[i] != ' ' : # Si on est à l'intérieur d'un mot
            mot = mot + s[i]
            taille_mot = taille_mot + 1
        else : # Si on a fini de lire un mot
            if taille_ligne+taille_mot+1 <= n and taille_ligne > 0 :
                # Si le mot rentre dans la ligne
                ligne = ligne + ' ' + mot
                taille_ligne = taille_ligne + taille_mot + 1
            else :
                # Sinon, on commence une nouvelle ligne
                if taille_ligne > 0 :
                    print(ligne)
                ligne = mot
                taille_ligne = taille_mot
            # On remet mot et taille_mot à zéro
            mot = ''
            taille_mot = 0
    # Il reste à s'occuper du dernier mot
    if taille_ligne + taille_mot != 0 :
        if taille_ligne + taille_mot > n :
            print(ligne)
            print(mot)
        else :
            print(ligne,mot)
```

Exercice 5 — Tapis : le retour !

À faire chez vous : refaire les tapis du TP 2 mais en utilisant uniquement des boucles `for` (donc sans `while`) et sans utiliser de fonctions auxilaires (donc vous êtes obligés de faire une boucle `for` dans une boucle `for`).

```python
def tapis_a(largeur,hauteur):
    for i in range(hauteur):
        for j in range(largeur):
            étoile()
        nouvelle_ligne()

def tapis_b(largeur,hauteur):
    for i in range(hauteur):
        for j in range(largeur):
            if j%2==0: # colonne pair
                étoile()
            else: # colonne impair
                dièse()
        nouvelle_ligne()


def tapis_c(largeur,hauteur):
    for i in range(hauteur):
        for j in range(largeur):
            if i%2==0: # ligne pair
                if j%2==0: # colonne pair
                    étoile()
                else:      # colonne impair
                    dièse()
            else: # ligne impair
                if j%2==0: # colonne pair
                    dièse()
                else:      # colonne impair
                    étoile()
        nouvelle_ligne()


def tapis_d(largeur,hauteur):
    for i in range(hauteur):
        for j in range(largeur):
            if i%3==0: # première ligne
                if j%2==0: # colonne pair
                    étoile()
                else:      # colonne impair
                    dièse()
            elif i%3==1: # seconde ligne
                if j%2==0: # colonne pair
                    dièse()
                else:      # colonne impair
                    étoile()
            else: # troisième ligne
                étoile()
        nouvelle_ligne()
```