Probleme

October 15, 2021

1 Un problème à résoudre en python

En partant d'un programme simple permettant de lire les lignes d'un fichier, on s'applique à le transformer de sorte que l'on obtienne la liste des caractères présents dans le fichier chacun associé à son nombre d'occurences.

Tout d'abord le programme permettant de lire un fichier.

[20]:
```
fichier = open("contenu.txt","r")
for ligne in fichier:
    print("J'ai lu: "+ligne)
fichier.close()
```

```
J'ai lu: bonjour, $

J'ai lu: comment ca va ?

J'ai lu: Est-ce qu'il y a des haricots a la cantine ?

J'ai lu: Bien a toi, JB
```

On observe que le fichier contient des phrases constituées de mots. Ainsi dans la boucle principle qui permet de lire ligne par ligne, pour chaque ligne obtenue on s'applique à la découper en mots par l'usage de la **méthode** (fonction contextualisée) `split()` des chaînes de caractères.

Connaître les méthodes applicables à une chaîne de caractères nécessite le parcours de la documentation. Par exemple celle du type `string` de Python : https://docs.python.org/3/library/string.html.

Malheureusement dans ce document, la méthode `split()` n'est pas référencée car comme indiqué tout en haut les `string` Python sont aussi des `Text Sequence`et il faut donc chercher aussi dans la documentation de ce type : https://docs.python.org/3/library/stdtypes.html#textseq.

On y trouve alors la méthode `split` dont le premier fragment indique :

> str.split(sep=None, maxsplit=- 1)
>
> > Return a list of the words in the string, using sep as the delimiter string. If maxsplit is given, at most maxsplit splits are done (thus, the list will have at most maxsplit+1 elements). If maxsplit is not specified or -1, then there is no limit on the number of splits (all possible splits are made).

Ensuite pour chaque mot, une chaîne de caractères et donc une séquence, on boucle pour en extraire les caractères.

[21]:
```
fichier = open("contenu.txt","r")
for ligne in fichier:
    print("ligne: "+ligne)
    for mot in ligne.split():
        print("mot: "+mot)
        for lettre in mot:
            print("J'ai lu: "+lettre)
fichier.close()
```

```
ligne: bonjour, $

mot: bonjour,
J'ai lu: b
J'ai lu: o
J'ai lu: n
J'ai lu: j
J'ai lu: o
J'ai lu: u
J'ai lu: r
J'ai lu: ,
mot: $
J'ai lu: $
ligne: comment ca va ?

mot: comment
J'ai lu: c
J'ai lu: o
J'ai lu: m
J'ai lu: m
J'ai lu: e
J'ai lu: n
J'ai lu: t
mot: ca
J'ai lu: c
J'ai lu: a
mot: va
J'ai lu: v
J'ai lu: a
mot: ?
J'ai lu: ?
ligne: Est-ce qu'il y a des haricots a la cantine ?

mot: Est-ce
J'ai lu: E
J'ai lu: s
```

```
J'ai lu: t
J'ai lu: -
J'ai lu: c
J'ai lu: e
mot: qu'il
J'ai lu: q
J'ai lu: u
J'ai lu: '
J'ai lu: i
J'ai lu: l
mot: y
J'ai lu: y
mot: a
J'ai lu: a
mot: des
J'ai lu: d
J'ai lu: e
J'ai lu: s
mot: haricots
J'ai lu: h
J'ai lu: a
J'ai lu: r
J'ai lu: i
J'ai lu: c
J'ai lu: o
J'ai lu: t
J'ai lu: s
mot: a
J'ai lu: a
mot: la
J'ai lu: l
J'ai lu: a
mot: cantine
J'ai lu: c
J'ai lu: a
J'ai lu: n
J'ai lu: t
J'ai lu: i
J'ai lu: n
J'ai lu: e
mot: ?
J'ai lu: ?
ligne: Bien a toi, JB

mot: Bien
J'ai lu: B
J'ai lu: i
J'ai lu: e
```

```
J'ai lu: n
mot: a
J'ai lu: a
mot: toi,
J'ai lu: t
J'ai lu: o
J'ai lu: i
J'ai lu: ,
mot: JB
J'ai lu: J
J'ai lu: B
```

Comme on cherche à ne retenir que les caractères alphabétiques, on utilise alors la méthode `isalpha()` pour déterminer si celui-ci est alphabétique ou non :

> str.isalpha()
>
> Return True if all characters in the string are alphabetic and there is at least one character, False otherwise. Alphabetic characters are those characters defined in the Unicode character database as "Letter", i.e., those with general category property being one of "Lm", "Lt", "Lu", "Ll", or "Lo". Note that this is different from the "Alphabetic" property defined in the Unicode Standard.

La définition de ce qu'est un caractère alphabétoique est assez technique et dépend de l'encodage des caractères. Mais laissons cet point de côté.

Une fois que l'on a déterminé si un caractère est alphabétiue ou non, on va utiliser ce résultat pour effectuer des calculs différents selon le cas. Pour cela on utilise l'alternative permettant si une condition est vraie de réaliser un traitement et sinon un autre :

```
if condition:
    # calcul si c'est vrai
else:
    # calcul si c'est faux
```

La partie `else` peut être omise si on ne souhaite réaliser un calcul que pour la partie *positive*.

[22]:
```
fichier = open("contenu.txt","r")
for ligne in fichier:
    for mot in ligne.split():
        for lettre in mot:
            if lettre.isalpha():
                print("alphabétique: "+lettre)
            else:
                print("non alphabétique:"+lettre)
fichier.close()
```

```
alphabétique: b
alphabétique: o
alphabétique: n
alphabétique: j
alphabétique: o
```

```
alphabétique: u
alphabétique: r
non alphabétique:,
non alphabétique:$
alphabétique: c
alphabétique: o
alphabétique: m
alphabétique: m
alphabétique: e
alphabétique: n
alphabétique: t
alphabétique: c
alphabétique: a
alphabétique: v
alphabétique: a
non alphabétique:?
alphabétique: E
alphabétique: s
alphabétique: t
non alphabétique:-
alphabétique: c
alphabétique: e
alphabétique: q
alphabétique: u
non alphabétique:'
alphabétique: i
alphabétique: l
alphabétique: y
alphabétique: a
alphabétique: d
alphabétique: e
alphabétique: s
alphabétique: h
alphabétique: a
alphabétique: r
alphabétique: i
alphabétique: c
alphabétique: o
alphabétique: t
alphabétique: s
alphabétique: a
alphabétique: l
alphabétique: a
alphabétique: c
alphabétique: a
alphabétique: n
alphabétique: t
alphabétique: i
```

```
alphabétique: n
alphabétique: e
non alphabétique:?
alphabétique: B
alphabétique: i
alphabétique: e
alphabétique: n
alphabétique: a
alphabétique: t
alphabétique: o
alphabétique: i
non alphabétique:,
alphabétique: J
alphabétique: B
```

Ici on affiche que les caractètes alphabétiques.

[23]:
```
fichier = open("contenu.txt","r")
for ligne in fichier:
    for mot in ligne.split():
        for lettre in mot:
            if lettre.isalpha():
                print("alphabétique: "+lettre)
fichier.close()
```

```
alphabétique: b
alphabétique: o
alphabétique: n
alphabétique: j
alphabétique: o
alphabétique: u
alphabétique: r
alphabétique: c
alphabétique: o
alphabétique: m
alphabétique: m
alphabétique: e
alphabétique: n
alphabétique: t
alphabétique: c
alphabétique: a
alphabétique: v
alphabétique: a
alphabétique: E
alphabétique: s
alphabétique: t
alphabétique: c
alphabétique: e
alphabétique: q
```

```
alphabétique: u
alphabétique: i
alphabétique: l
alphabétique: y
alphabétique: a
alphabétique: d
alphabétique: e
alphabétique: s
alphabétique: h
alphabétique: a
alphabétique: r
alphabétique: i
alphabétique: c
alphabétique: o
alphabétique: t
alphabétique: s
alphabétique: a
alphabétique: l
alphabétique: a
alphabétique: c
alphabétique: a
alphabétique: n
alphabétique: t
alphabétique: i
alphabétique: n
alphabétique: e
alphabétique: B
alphabétique: i
alphabétique: e
alphabétique: n
alphabétique: a
alphabétique: t
alphabétique: o
alphabétique: i
alphabétique: J
alphabétique: B
```

Maintenant que nous avons obtenu la liste des caractères alphabétiques, il nous faut les compter. Mais auparavant nous devons essayer de trouver comment structurer les données que nous cherchons à obtenir. Nous souhaiterions obtenir quelque chose qui associe à une lettre donnée, son nombre d'occurences. La structure de données adéquate s'appelle un **dictionnaire**. C'est-à-dire une structure de donnée qui contient des couples (clé,valeur), les clés et valeurs pouvant être de n'importe quel type. Une dictionnaire peut être vu comme un tableau de deux colonnes, la première contenant les clés et la seconde les valeurs associées. De plus dans un dictionnaire on peut rechercher la valeur associée à une clé connue.

Par exemple :

[24]:
```
mon_dictionnaire = { "einstein":1879, "chomsky":1928 }
print(mon_dictionnaire)
```

```
{'einstein': 1879, 'chomsky': 1928}
```

[25]:
```
mon_dictionnaire["einstein"]
```

[25]:
```
1879
```

Ainsi, si l'on est capable de générer un dictionnaire comme celui qui suit on pourrait alors présenter les résultats comme on l'entend :

[26]:
```
r = { "a":12, "b":2}
print(f"a est présent {r['a']} fois.")
```

```
a est présent 12 fois.
```

Si l'on souhaite boucler (passer en revue) l'ensemble des couples (clé,valeur) d'un dictionnaire on peut l'effectuer en en récupérant la séquence par un appel à `items()`:

[27]:
```
for (k,v) in r.items():
    print(f"la lettre {k} est présente {v} fois.")
```

```
la lettre a est présente 12 fois.
la lettre b est présente 2 fois.
```

Il ne nous reste plus qu'à générer la structure en parcourant les caractères alphabétiques contenus dans le fichier.

Pour cela, on commence par un dictionnaire vide (ne contenant rien). ALors à chaque caractère rencontré, si celui-ci est rencontré pour la première fois (on le détermine en interrogeant le dictionnaire sur l'existence de la clé par une instruction comme `lettre in dictionnaire`) on ajoute au dictionnaire le couple (lettre,1). Si la lettre avait déjà été rencontré il faut modifier l'entrée du dictionnaire pour ajouter 1, par une instruction comme `dictionnaire[lettre] = dictionnaire[lettre]+1`.

On notera au passage que l'affichage produit les lettres dans l'ordre alphabétique, ce qui est obtenu par un appel à la fonction de tri `sorted` qui recevant une structure de donnée en trie les éléments :

[28]:
```
resultat = {}
fichier = open("contenu.txt","r")
for ligne in fichier:
    for mot in ligne.split():
        for lettre in mot:
            if lettre.isalpha():
                l = lettre.lower() # transforme la lettre en minuscule
                if l in resultat:
                    resultat[l] = resultat[l]+1 # une occurence de plus
                else:
                    resultat[l] = 1 # première rencontre
fichier.close()
for l in sorted(resultat):
```

```
    print(f"la lettre {l} est présente {resultat[l]} fois.")
```

```
la lettre a est présente 8 fois.
la lettre b est présente 3 fois.
la lettre c est présente 5 fois.
la lettre d est présente 1 fois.
la lettre e est présente 6 fois.
la lettre h est présente 1 fois.
la lettre i est présente 5 fois.
la lettre j est présente 2 fois.
la lettre l est présente 2 fois.
la lettre m est présente 2 fois.
la lettre n est présente 5 fois.
la lettre o est présente 5 fois.
la lettre q est présente 1 fois.
la lettre r est présente 2 fois.
la lettre s est présente 3 fois.
la lettre t est présente 5 fois.
la lettre u est présente 2 fois.
la lettre v est présente 1 fois.
la lettre y est présente 1 fois.
```