La langue Zorglangue

CPP

31 mars 2015

1 But du TP et algorithme

Dans ce TP, il vous est demandé de programmer un système qui permet de traduire un texte de la langue française en un autre écrit dans la langue zorglangue. La zorglangue est une langue écrite et parlée inventée par **Zorglub**, personnage de la série de bande dessinée Spirou et Fantasio. Le principe de ce langage est d'inverser les lettres de chaque mot d'un segment de texte en conservant l'ordre des mots dans le segment. On traitera tous les segments de texte de la langue française : un mot, une phrase ou une portion de texte. Ainsi, pour traduire la phrase "Zorglub, a-t-il mis au point Zorg-Langue ?" on doit tenir compte des lettres majuscules/minuscules, des lettres accentuées et des caractères de ponctuation, comme le montre l'exemple suivant. La même opération de traduction appliquée à un fragment de texte de la zorglangue doit donner le segment de texte correspondant dans la langue française.

```
   "Zorglub, a-t-il mis au point Zorg-Langue?"
  + traduire
-----------------------------------------------
   "Bulgroz, a-t-li sim ua tniop Groz-Eugnal?"
```

```
   "Bulgroz, a-t-li sim ua tniop Groz-Eugnal?"
 + traduire
-----------------------------------------------
   "Zorglub, a-t-il mis au point Zorg-Langue?"
```

Afin de faciliter la traduction, chaque segment de texte est fragmenter en lexème (token). Das le cas présent, un lexème est une unité de signification appartenant au lexique suivant : un mot composé de lettres majuscules et/ou minuscules et éventuellement des lettres accentuées, caractère de ponctuation et le caractère d'espacement. À noter, que lors de l'inversion des lettres d'un mot la place occupée par une lettre majuscule sera reprise par une lettre majuscule. Par exemple, `inverser("PyThon")`→ "NoHtyp" ; et `inverser("NoHtyp")`→ "PyThon".

```
   "C'est bien Zorg-Langue!"
  + fragmenter
-------------------------------------------------------------------
   ['C', "'", 'est', ' ', 'bien', ' ', 'Zorg', '-', 'Langue', '!']
```

2 Travail demandé

Votre rendu devra contenir les fichiers suivants :

`inverse.py` : fichier contenant une fonction `inverser` qui inverse les lettres d'un mot (type `str`) du lexique défini ci-dessus et retourne le mot inversé (type `str`).

`test_inverse.py` : les tests de la fonction `inverser`.

`fragment.py` : fichier contenant la fonction `fragmenter` paramétrée par une chîne de caractères (type `str`), et qui retourne une liste (type `list`) de lexèmes (voir exemple).

`test_fragment.py` : les tests de la fonction `fragmenter`

`traduire.py` : fichier contenant la fonction `traduire` paramétrée par une chîne de caractères (type `str`) et retourne la traduction correspondante sous forme de chaîne de caractères.

`test_traduire.py` : les tests de la fonction `traduire`.

`utils.py` : ce fichier peut contenir des fonctions utiles à la fois à `inverser`, `fragmenter` et `traduire`.

`test_utils.py` : si vous avez placé des fonctions dans `utils.py`, alors ce fichier doit contenir les tests pour ces fonctions.

`top.py` : le programme principal, qui fait toutes les interactions avec l'utilisateur (`print` et `input`).

`rapport.pdf` : votre compte-rendu. Le compte-rendu doit rappeler brièvement le principe du TP, et documenter les spécificités de votre code (a-t-il des limitations par rapport au travail demandé ? Y a-t-il des choses en plus ?). Il devrait faire environ 1 page.

Pour vous faciliter la tâche, un squelette de code est fourni, il contient tous ces fichiers. À vous de les compléter.

Une fois terminé, une exécution du programme principal (ouvrir `top.py` dans Spyder3, puis lancer « run module ») doit ressembler à (les portions en italiques sont les entrées clavier faites par l'utilisateur) :

```
Entrez un texte à traduire: Zorglub, a-t-il mis au point Zorg-Langue?
La traduction est -->  Bulgroz, a-t-li sim ua tniop Groz-Eugnal?
Voulez-vous traduire le résultat Y/n? Y
La traduction du résultat est -->  Zorglub, a-t-il mis au point Zorg-Langue?
Entrez "Q" pour quitter ou Entrée pour continuer:

Entrez un texte à traduire:
La traduction est -->
Voulez-vous traduire le résultat Y/n? Y
La traduction du résultat est -->
Entrez "Q" pour quiter ou Entrée pour continuer:  Q
```

3 Consignes et conseils

Il n'est pas suffisant de fournir une implémentation (un programme) correct. L'évaluation prendra également en compte :

- La clarté du code (les noms de variables et de fonctions sont-ils choisis judicieusement ? les portions de code communes à plusieurs fonctions sont-elle placées dans des fonctions au lieu d'être copié-collées ?). [1]
- Les commentaires dans le code [2] pour documenter ce que fait chaque fonction, ses préconditions, ...
- Les tests qui permettent de vérifier que chaque fonction fait bien ce qu'elle est censée faire. [3]

A Manipulations de chaînes en Python

En Python, les chaînes de caractères forment un *type composite* ; une liste de caractères non modifiable. La représentaion en machine dune chaîne de caractères est une suite de nombres (appelées code points) représentant chaque caractère [4]. Les caractères sont des chaînes de caractère de taille 1. Il n'y a pas de différence entre « Le caractère `'î'` » et « la chaîne composée de l'unique caractère `'î'` » (tous deux du type `'str'`).

Parmi les fonctions définies sur les chaînes de caractères utiles pour ce TP, on trouve : `isupper()` qui indique si une lettre est majuscule, `upper()` qui transforme une lettre minuscule en lettre majuscule et `lower()` qui transforme une lettre majuscule en lettre minuscule.

Étant donné une chaîne de caractères `chaine`, on peut accéder au nième caractère avec `chaine[n]` (attention, la numérotation commence à 0, donc le premier caractère est en fait `chaine[0]`). Il est possible de parcourir la chaîne de caractère `chaîne` de la droite vers la gauche. Dans ce cas, la numérotation commence à -1, ansi le dernier caractère est `chaîne[-1]`, l'avant dernier est `chaîne[-2]`...

On peut concaténer des chaînes avec l'opérateur `+`. Par exemple, `'messag'+ 'e'` vaut `'message'`.

Pour parcourir les caractères d'une chaîne, on peut utiliser une boucle `for` comme ceci :

```
for c in chaine:
    print('Un caractère :', c)
```

Pour plus d'informations, la page du cours donne quelques liens sur des documentations supplémentaires. Voir aussi la section 3.3.2 page 77 du livre « Informatique pour tous ».

Note : le squelette de code fourni utilise une variante de `import` *`module`* : `from` *`module`* `import *`. Cette construction permet par exemple d'écrire `traduire(...)`, après avoir écrit `from traduire import *`. C'est un peu plus court que le `traduire.traduire(...)` qui aurait été nécessaire avec un simple `import traduire`.

1. Les plus rigoureux d'entre-vous pourront s'assurer que leur code est conforme aux recommandations PEP8 (`http://legacy.python.org/dev/peps/pep-0008/`), mais il n'est pas obligatoire d'aller jusque là pour ce TP.
2. et/ou éventuellement des docstrings Python.
3. En général, les tests représentent plus de la moitié du temps de développement, et donc du coût d'un logiciel.
4. Les caractères moins usuels sont aussi représentables avec un nombre, mais plus grand : c'est le standard Unicode.