

# TME Solitaire: Le secret et l'apparent

Durée : 45 minutes

le 10 Novembre, 2015

ATTENTION : Dans toute la suite les mots sont considérés sans accent.  
La soumission de votre copie se fait comme d'habitude sur la page  
<http://www-licence.ufr-info-p6.jussieu.fr/lmd/licence/2015/ue/1I001-2015oct>

**Question 1 : (4 points)** Écrivez une fonction `arriere_avant` qui étant donné une chaîne de caractères *chaîne* renvoie la chaîne de caractères issue de la concaténation de la seconde moitié de *chaîne* puis de la première moitié. Lorsque la longueur de *chaîne* est impaire, le caractère du milieu ne change pas de place. Par exemple :

```
>>> arriere_avant("")
""
>>> arriere_avant("renard")
"ardren"
>>> arriere_avant("louve")
"veulo"
>>> arriere_avant("aabcc")
"ccbbaa"
>>> arriere_avant("Belle Marquise, vos yeux... me font mourir beau d'amour")
"me font mourir beau d'amour Belle Marquise, vos yeux..."
```

**Question 2 : (1 point)** Que pouvez-vous dire des différents appels :

```
>>>arriere_avant(arriere_avant("renard"))
>>>arriere_avant(arriere_avant("louve"))
>>>arriere_avant(arriere_avant("Il se passe quelque chose d'etrange."))
```

**Question 3 : (2 points)** On définit dans la suite l'objet suivant :

```
\#clef : ???
clef=[('i', 'd'), ('o', 'n'), ('t', 'k'), ('w', 'h'), ('a', 'l'), ('f', 'y'),
      ('u', 's'), ('e', 'r'), ('v', 'c'), ('b', 'g'), ('j', 'm'), ('q', 'p'),
      ('x', 'z'), ('d', 'i'), ('n', 'o'), ('k', 't'), ('h', 'w'), ('l', 'a'),
      ('y', 'f'), ('s', 'u'), ('r', 'e'), ('c', 'v'), ('g', 'b'), ('m', 'j'),
      ('p', 'q'), ('z', 'x'), ('I', 'D'), ('O', 'N'), ('T', 'K'), ('W', 'H'),
      ('A', 'L'), ('F', 'Y'), ('U', 'S'), ('E', 'R'), ('V', 'C'), ('B', 'G'),
      ('J', 'M'), ('Q', 'P'), ('X', 'Z'), ('D', 'I'), ('N', 'O'), ('K', 'T'),
      ('H', 'W'), ('L', 'A'), ('Y', 'F'), ('S', 'U'), ('R', 'E'), ('C', 'V'),
      ('G', 'B'), ('M', 'J'), ('P', 'Q'), ('Z', 'X')]
```

Vous trouverez cet objet sur la page

<https://www-almasty.lip6.fr/~pierrot/papers/clef.py>

Copiez puis collez ces lignes dans votre fichier avant d'écrire la fonction suivante. Quel est le type de *clef* ? Complétez la déclaration de celle-ci.

**Question 4 : (9 points)** Écrivez une fonction **correspondance** qui étant donné une chaîne de caractères *chaîne* renvoie la chaîne de caractères pour laquelle chaque caractère de *chaîne* a été remplacé par le caractère correspondant, selon la liste donnée par *clef*. Par exemple :

```
>>> correspondance("")
""
>>> correspondance("i")
"d"
>>> correspondance("renard")
"erolei"
```

Lorsqu'un caractère n'apparaît pas dans *clef*, celui-ci reste inchangé. Par exemple :

```
>>> correspondance("Les espaces, comme la ponctuation, ne sont pas corrompus.")
"Aru ruqlvru, vnjjr al qnovkslkdn, or unok qlu vneenjqsu."
```

Pour remplacer un caractère *c* par son correspondant, il faut donc parcourir les paires de *clef*, et, s'il existe une paire dont le premier élément est *c*, remplacer *c* par le second caractère de cette paire.

**Question 5 : (1 point)** Que pouvez-vous dire des différents appels :

```
>>>correspondance(correspondance("renard"))
>>>correspondance(correspondance("louve"))
>>>correspondance(correspondance("Il se passe quelque chose d'etrange."))
```

**Question 6 : (3 points)** On a intercepté le `message_secret` que vous trouverez ici : <https://www-almasty.lip6.fr/~pierrot/papers/clef.py>

En vous aidant des questions précédentes, et sachant que pour chiffrer le message son auteur a d'abord appliqué la fonction `arriere_avant`, puis la fonction `correspondance`, proposez une ligne de code qui permette de déchiffrer le message.

Dans quel lieu se déroule la scène ?

---

**Pour aller plus loin : (Bonus)** Regardez plus attentivement l'objet `clef`. Vous constatez, en faisant abstraction des parenthèses et des guillemets, que l'on peut lire le début de la phrase "i don't know half of you half as well as i should like, and i like less than half of you half as well as you deserve." Ecrivez ainsi la fonction `creation_clef` qui, étant donné une chaîne de caractères `phrase` constituée uniquement de minuscules, de points, de virgules, d'apostrophes et d'espaces, reconstitue une clef (dont vous donnerez le type) selon les principes suivants :

- les espaces, les apostrophes et la ponctuation ne sont pas pris en compte
- les couples de majuscules sont identiques aux couples de minuscules
- l'ordre des couples importe peu
- si le couple ("a", "b") est présent dans la clef alors le couple ("b", "a") l'est aussi

D'un point de vue algorithmique, votre fonction devra parcourir l'ensemble des caractères de votre `phrase`, et pour chacun d'entre eux, l'ajouter à un couple si ce caractère n'a pas encore été rencontré. Par exemple :

```
>>> creation_clef("portez ce vieux whisky au juge blond qui fume.")
[('p', 'o'), ('r', 't'), ('e', 'z'), ('c', 'v'), ('i', 'u'), ('x', 'w'),
 ('h', 's'), ('k', 'y'), ('a', 'j'), ('g', 'b'), ('l', 'n'), ('d', 'q'),
 ('f', 'm'), ('o', 'p'), ('t', 'r'), ('z', 'e'), ('v', 'c'), ('u', 'i'),
 ('w', 'x'), ('s', 'h'), ('y', 'k'), ('j', 'a'), ('b', 'g'), ('n', 'l'),
 ('q', 'd'), ('m', 'f'), ('P', 'O'), ('R', 'T'), ('E', 'Z'), ('C', 'V'),
 ('I', 'U'), ('X', 'W'), ('H', 'S'), ('K', 'Y'), ('A', 'J'), ('G', 'B'),
 ('L', 'N'), ('D', 'Q'), ('F', 'M'), ('O', 'P'), ('T', 'R'), ('Z', 'E'),
 ('V', 'C'), ('U', 'I'), ('W', 'X'), ('S', 'H'), ('Y', 'K'), ('J', 'A'),
 ('B', 'G'), ('N', 'L'), ('Q', 'D'), ('M', 'F')]
```

Lorsque tous les caractères de la phrase ont été parcourus, on complète la clef par les lettres de l'alphabet non encore utilisées. Par exemple :

```
>>> creation_clef("i don't know half of you as well as i should like,
and i like less than half of you half as well as you deserve.")
[('i', 'd'), ('o', 'n'), ('t', 'k'), ('w', 'h'), ('a', 'l'), ('f', 'y'),
```

```
('u', 's'), ('e', 'r'), ('v', 'c'), ('b', 'g'), ('j', 'm'), ('q', 'p'),  
( 'x', 'z'), ('d', 'i'), ('n', 'o'), ('k', 't'), ('h', 'w'), ('l', 'a'),  
( 'y', 'f'), ('s', 'u'), ('r', 'e'), ('c', 'v'), ('g', 'b'), ('m', 'j'),  
( 'p', 'q'), ('z', 'x'), ('I', 'D'), ('O', 'N'), ('T', 'K'), ('W', 'H'),  
( 'A', 'L'), ('F', 'Y'), ('U', 'S'), ('E', 'R'), ('V', 'C'), ('B', 'G'),  
( 'J', 'M'), ('Q', 'P'), ('X', 'Z'), ('D', 'I'), ('N', 'O'), ('K', 'T'),  
( 'H', 'W'), ('L', 'A'), ('Y', 'F'), ('S', 'U'), ('R', 'E'), ('C', 'V'),  
( 'G', 'B'), ('M', 'J'), ('P', 'Q'), ('Z', 'X')]
```

Indications: Créez simplement les couples de minuscules, dans un premier temps. Vous pouvez copier/coller l'ensemble *alphabet* donné en ligne si cela vous aide. Pour créer ensuite les couples de majuscules, on rappelle que la fonction **ord** de signature

```
str -> int
```

renvoie le numéro Unicode du caractère pris en argument tandis que la fonction **chr** de signature

```
int -> str
```

renvoie le caractère de numéro Unicode l'entier pris en argument. Par exemple :

```
>>>ord("a")  
97  
>>>chr(98)  
'b'
```

A votre tour, créez la clef de votre choix et chiffrez-nous un message !