TP 7 - Fonctions, chaînes de caractères

La documentation en ligne de Python 3 se trouve ici : `http://docs.python.org/3/`. Consultez-la régulièrement si vous avez des doutes sur une fonction ou une fonctionnalité.

Attention : dans tout le TP, chaque fonction que vous écrivez doit être précédée d'un commentaire qui indique ce qu'elle fait en fonction des paramètres donnés !

Exercice 1. Chaînes de caractères

1. Déclarer et faire afficher la chaîne de caractères suivante (une seule chaîne) :

   ```
   Bonjour
   le "Monde"
   et l'Univers
   ```

2. Déclarer et afficher une deuxième chaîne de caractères identique, sans utiliser d'antislash.
3. Écrire une fonction `afficheQuestion` prenant en paramètre une chaîne *numero* et affichant la ligne suivante précédée d'une ligne blanche (avec le bon numéro) :

   ```
   ########## Question n ##########
   ```

   Par exemple `afficheQuestion('2.a')` affichera `Question 2.a`
4. On souhaite écrire une fonction prenant en paramètre une chaîne de caractères et réaffichant celle-ci avec le texte "voyez-vous" après chaque virgule et ", n'est-ce pas" avant chaque point. Par exemple, à partir du texte suivant :

   ```
   Hier, j'ai mangé une pomme.
   Après, j'ai eu des hallucinations, plein d'hallucinations.
   ```

   on obtient :

   ```
   Hier, voyez-vous, j'ai mangé une pomme, n'est-ce pas.
   Après, voyez-vous, j'ai eu des hallucinations, voyez-vous, plein d'halluci-
   nations, n'est-ce pas.
   ```

   Remarque : pour utiliser des caractères accentués comme dans cet exemple, vous devez ajouter le sha-bang suivant au début de votre fichier Python :

   ```
   #!/usr/bin/env python
   # -*- coding: utf-8 -*-
   ```

   (a) Écrire cette fonction nommée `langageV1` en utilisant `for` et `range`.

   (b) Ré-écrire la même fonction nommée `langageV2` en utilisant `for`, **mais pas** `range`.

   (c) Écrire la fonction `langageV3` qui **n'affiche pas** la chaîne obtenue mais la **renvoie**.

   (d) Modifier la fonction `langageV3` pour qu'elle prenne en paramètre les chaînes remplaçant les points et les virgules. On pourra, par exemple, tester :

   ```
   print(langageV3(texte, ", t'as vu,", ", quoi."))
   ```

   (e) **(facultatif)** Écrire la fonction `langageV4` qui fait exactement la même chose, mais sans utiliser de boucle (ni `while`, ni `for`).
   Indice : chercher dans la documentation en ligne relative aux chaînes de caractères : `http://docs.python.org/3/library/stdtypes.html#text-sequence-type-str`.

   (f) **(facultatif)** Refaire la fonction précédente avec une seule instruction.

Exercice 2. Intervalles
Répondre aux questions suivantes en une seule ligne en utilisant `range`. Si vous ne savez pas comment faire, allez voir la doc.

1. Tester si un entier entré par l'utilisateur est entre 0 et 10 (non inclus). Pouvez-vous faire la même chose avec un flottant ?
2. Tester si un entier entré par l'utilisateur est entre 10 et 20 (non inclus).
3. Afficher la liste des entiers entre 100 et 150. Pourquoi ne pouvez-vous pas faire la même chose avec des flottants ?
4. Afficher la liste des entiers pairs entre 100 et 150.
5. Afficher la liste des entiers impairs entre 150 et 200.
6. Afficher la liste décroissante des entiers 150 et 200.
7. Afficher la liste décroissante des entiers multiples de 3 entre -150 et -200.

Exercice 3. Code de César : cryptographie et cryptanalyse
Le but de cet exercice est de réaliser un système permettant de crypter des messages, en utilisant le code de César : on part d'un message en clair et on change toutes les lettres en les décalant d'un même nombre de positions (ce nombre est appelé la *clé*). Pour décoder le message il suffit alors de décaler dans l'autre sens, à condition de connaître la clé. Pour cet exercice, nous vous encourageons à chercher dans la doc quelles sont les fonctions qui peuvent vous être utiles. **Attention, ce qui suit ne fonctionne que pour des caractères sans accent !**

1. On va commencer par écrire la fonction d'encodage du texte : `encrypte(texteEnClair,cle)`. Toutes les lettres sont cryptées, mais pas les blancs et la ponctuation. Par exemple, ce code affiche les deux lignes notées en commentaire :

```
1 cache=encrypte('Toto est une girafe, avec des petites jambes.',12)
2 print(cache)                # Fafa qef gzq sudmrq, mhqo pqe bqfufqe vmynqe.
3 print(decrypte(cache,12)) # Toto est une girafe, avec des petites jambes.
```

   Pour cela, nous allons découper le travail :
   (a) Écrire une fonction `encrypteLettre` qui prend une seule lettre en paramètre et renvoie la lettre encodée (attention à la ponctuation et aux majuscules). Pour décaler les lettres, vous aurez besoin de 2 fonctions de bases de Python : `ord` et `chr`. La doc est là : `http://docs.python.org/3/library/functions.html`. La fonction `ord` prend une chaine contenant un seul caractère et renvoie son *code ascii* (`ord('a')` donne 97, `ord('b')` 98, etc...). Et la fonction `chr` fait l'inverse : elle renvoie le caractère en fonction du code ascii. Par exemple, le programme suivant permet de créer une liste contenant les lettres de l'alphabet :

```
1 alphabet=[]
2 for i in range(26):
3   alphabet.append(chr(ord('a')+i))
```

   Vous devez calculer l'indice de la lettre dans l'alphabet, puis le décaler avec la clé (en prenant soin de rester entre 0 et 25 : par exemple, 'z' décalé de 2 donne 'b') et retrouver le caractère correspondant. Pour vérifier : lorsque l'ont fait `for c in alphabet: print(encrypteLettre(c,10), end=' ')` :, on doit obtenir :
   `k l m n o p q r s t u v w x y z a b c d e f g h i j`.
   Si vous êtes coincé, appelez votre chargé de TP.
   (b) On constate que l'on fait 2 fois le même travail, pour une majuscule ou une minuscule. Rajouter une fonction auxiliaire qui prend 'a' ou 'A' en paramètre et permet de ne faire le calcul qu'une seule fois.

(c) Écrire une fonction `encrypteMot` qui prend un mot et renvoie le mot encodé.

(d) Enfin écrire `encrypte` qui renvoie le texte codé.

2. On passe au décodage. Écrire la fonction `decrypte(texteCache,cle)` qui permet de décoder (et renvoyer) un texte qui a été crypté avec la clé. Soyez astucieux, ne copiez-collez pas votre code, réutilisez plutôt vos fonctions !

3. Passons à la cryptanalyse, c'est-à-dire l'art de déchiffrer les messages sans avoir la clé ! Pour cela, nous allons essayer d'utiliser nos connaissances sur les fréquences des lettres dans un texte. En effet, vous n'êtes pas sans savoir que la lettre la plus commune en français est le 'e'... Donc, pour savoir quelle clé a permis de coder le texte, il suffit de tester toutes les clés possibles et de regarder celle qui donne le texte contenant le plus de 'e'. Écrire la fonction `devineCle(texteCache)` qui renvoie la clé devinée en utilisant ce principe et testez-la sur l'exemple. Attention, cette technique fonctionne d'autant mieux que le texte est long. Essayez de retrouver le texte original dans les deux cas suivants :
   Texte 1 :
   `Tuqd, v'mu ymzsq gzq bayyq. Mbdqe, v'mu qg pqe tmxxgouzmfuaze, bxquz p'tmxxgouzmfuaze.`
   `Vq hakmue pqe qxqbtmzfe, pqe xuoadzqe qf pqe bmzpme. Qf uxe qfmuqzf daeqe, fage daeqe.`

   Texte 2 :
   `Jli c'rsrkkrek ul mrjzjkrj, le rezdrc rl kyfiro zeuzxf, r c'rzxlzccfe jrwire, ez le trwriu, ez le tyriretfe, drzj gclkfk le rikzjfe, j'rmretrzk, kirzerek le size u'rcwr. Zc j'rggiftyr, mflcrek c'rgcrkzi u'le tflg mzw, drzj c'rezdrc gizk jfe mfc, uzjgrirzjjrek urej cr elzk rmrek hl'zc rzk gl c'rjjrzcczi.`

   Si vous voulez savoir pourquoi ça ne marche pas sur le second texte : `http://fr.wikipedia.org/wiki/La_Disparition_%28roman%29`

4. **(facultatif)** Cette façon de faire n'est pas très efficace, elle force à décoder 26 fois... Il vaudrait mieux chercher la lettre la plus fréquente dans le texte et chercher la clé permettant de la faire correspondre avec un 'e'. Écrire la fonction `devineCle2` qui utilise ce principe.
   Indice : (une fois que vous connaissez la position `pos` dans l'alphabet (entre 0 et 25) de la lettre la plus fréquente, il suffit de calculer : `( pos+ord('a')-ord('e') )%26`

5. On vient de voir qu'il n'est pas très difficile de décoder le message, même sans la clé. Cette méthode de cryptographie n'est donc pas très efficace. Pour empêcher de deviner en utilisant les fréquences, on va crypter en décalant de 1 la clé de codage à chaque mot. Avec l'exemple de "Toto...", ça donne : `Fafa rfg ibs vxgput, qlus uvj hwlalwk ctfuxl`.
   Écrire les fonctions `encrypteMieux(texte,cle)` et `decrypteMieux(texte,cle)` qui utilisent ce principe.

6. On se rend compte que les fonctions `encrypte` et `encrypteMieux` sont quasiment les mêmes. Transformez-les en une unique fonction `encrypteEncoreMieux` qui prend également en paramètre le décalage utilisé pour chaque mot : il suffit de mettre 0 pour retrouver `encrypte` et 1 pour `encrypteMieux` :

```
txt='Voici le texte'
print( encrypte(txt,12) )      # affiche: Hauou xq fqjfq
print( encrypteEncoreMieux(txt,12,0) )  # Hauou xq fqjfq
print( encrypteMieux(txt,12) )          # Hauou yr hslhs
print( encrypteEncoreMieux(txt,12,1) )  # Hauou yr hslhs
```

   Pensez à utiliser cette fonction pour décrypter aussi.

7. **(facultatif)** Essayez de modifier votre code pour gérer les accents...