LMI 1 - Semestre 2

Université d'Artois
Faculté des Sciences Jean Perrin

ALGO2 – Algorithmique et Programmation en Python 2

Fiche de TP numéro 1

Préalables

1. Le nom du répertoire centralisé qui contient toutes vos données, est `/home/ens/libreservice/prenom_nom`
   Vous pouvez y accéder depuis n'importe quel poste des salles machines de la faculté, que ce soit sous Linux ou sous Windows (nous travaillerons sous Linux en TP). Vous devriez trouver dans ce répertoire toutes vos données du premier semestre.
2. Dans ce répertoire, créez un sous-répertoire `algo2` dans lequel vous placerez tous les programmes écrits en TP (et tout votre travail relatif à Algo 2 d'une manière générale). Dans ce répertoire, créez deux autres répertoires : `tps` et `controles`.

Aujourd'hui vous travaillerez dans le répertoire `tps`. Vous pouvez organiser ce répertoire comme vous le souhaitez, mais vous aurez probablement intérêt à y créer au moins un sous-répertoire `tp1` pour aujourd'hui.

Présentation de l'environnement

Pour les TPs Python, vous travaillerez avec l'éditeur de texte Visual Code disponible pour Linux, Windows et Mac OS X.

Pour commencer la séance, placez vous dans le répertoire `tps/tp1`.

Conventions pour l'écriture des fonctions

Chaque définition de fonction commencera par un commentaire, qui contiendra toujours les mêmes informations.

1. La spécification de la fonction, c'est à dire le type de ses paramètres et son type de retour.
2. Une description textuelle de la fonction

Pour ré-importer une fonction déjà importée

Si vous êtes amenés à corriger votre fonction et donc à réimporter votre module (i.e. votre fichier contenant votre fonction), vous devez utiliser la fonction `reload` du module `imp` :

```
>>> import imp
>>> imp.reload(intro)
```

Exercices

Bricolage

Exercice 1 : On s'intéresse à la gestion d'un grand magasin. On dispose de deux structures : l'une donne le prix unitaire de chaque article, l'autre le nombre d'articles en rayon.

```
# TPrix : dictionnaire dont la clé est un article, la valeur son prix unitaire
# TPrix : dico(str : float)
lesPrix = {'perçeuse':60,'ampoule':3.5,'clous':1.99,'vis':2.20}
# TStock : dictionnaire dont la clé est un article, la valeur sa quantité
# TStock : dico(str : int)
leStock = {'perçeuse':15,'ampoule':75,'vis':300,'clous':400}
```

Q 1 . Spécifier puis écrire une fonction `informations_stocks` qui retourne, à partir des prix des articles et du nombre d'articles en rayon, les trois informations suivantes (dans l'ordre indiqué) :

- le montant total de marchandises en rayon ;
- le nbre total d'articles en rayon ;
- le prix moyen d'un article.

On considèrera qu'un article en rayon dont on ne connaît pas le prix coûte 1 euro ; s'il n'y a pas d'articles en rayon, le prix moyen d'un article est zéro.

Exemple :

```
>>> informations_stocks(leStock,lesPrix)
(2618.5, 790, 3.3145569620253164)
```

Note : On peut retourner un tuple facilement avec l'opérateur `,` : si `a`, `b` et `c` sont trois variables dans une fonction, l'instruction `return a,b,c` retourne un tuple composé de leurs trois valeurs.

Dans le grand magasin, les articles sont rangés dans différents départements.

```
# TDepts : dictionnaire dont la clé est un département,
# la valeur les articles qu'il a en rayon (et leur quantité)
# TDepts : dico(str : TStocks)
lesDepts = {'quincaillerie':leStock,'outillage':{'perçeuse':5,'visseuse':8}}
```

Q 2 . Spécifier puis écrire une fonction `info_depts` qui retourne les trois informations précédentes pour chaque département.

Exemple :

```
>>> infos = info_depts(lesDepts,lesPrix)
>>> infos
{'quincaillerie': (2618.5, 790, 3.3145569620253164),
 'outillage': (308, 13, 23.692307692307693)}
```

Q 3 . Spécifier puis écrire une fonction `index_articles` qui retourne, pour chaque article, les départements dans lesquels ont peut le trouver.

Exemple :

```
>>> index_du_magasin = index_articles(lesDepts)
>>> index_du_magasin
{'ampoule': ['quincaillerie'], 'vis': ['quincaillerie'],
 'perçeuse': ['quincaillerie', 'outillage'],
 'visseuse': ['outillage'], 'clous': ['quincaillerie']}
```

Q 4 . Générer la documentation en html de votre module.

Message secret

On désire crypter un texte par une méthode très simple : on établit une table de permutation associant à chaque lettre majuscule une autre majuscule.

Vous devez écrire et générer la documentation html de votre module.

Exercice 2 : Spécifier puis écrire une fonction `permuteListe(uneListe)` qui échange successivement chaque élément de la liste (l'élément au rang 0, puis au rang 1, puis ...) avec un autre élément de la liste choisi au hasard. La liste doit pouvoir contenir n'importe quel type d'élément. La liste donnée en paramètre est modifiée par cette fonction

Exemple :

```
> l = [ 'A', 'B', 'C', 'D' ]
> permuteListe(l)
['B', 'C', 'A', 'D']
> l
['B', 'C', 'A', 'D']
```

Exercice 3 : À l'aide de la fonction précédente, spécifier puis écrire une fonction `dictPerm()` qui fabrique et retourne un dictionnaire de permutation (tiré au hasard) pour les lettres majuscules.

Exemple d'utilisation :

```
> perm = dictPerm()
> perm
{'Z': 'A', 'X': 'N', 'Y': 'E', 'V': 'J', 'W': 'H', 'T': 'L', 'U': 'C',
 'R': 'O', 'S': 'Q', 'P': 'V', 'Q': 'Y', 'N': 'M', 'O': 'U', 'L': 'F',
 'M': 'K', 'J': 'I', 'K': 'R', 'H': 'T', 'I': 'B', 'F': 'Z', 'G': 'W',
 'D': 'P', 'E': 'G', 'B': 'X', 'C': 'D', 'A': 'S'}
```

Pour cette question, vous utiliserez la variable `ascii_uppercase` du module `string` qui contient la chaîne de caractères des lettres de l'alphabet en majuscules :

```
> import string
> string.ascii_uppercase
'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
```

Exercice 4 : Spécifier puis écrire la fonction `crypte(txt, perm)` qui crypte le texte `txt` à l'aide du dictionnaire de permutation `perm`.

Exercice 5 : Pour décoder le texte crypté, il faut appliquer les mêmes permutations à l'envers. Écrire une fonction `invertDict(perm)` qui renverse un dictionnaire. La fonction retourne un autre dictionnaire, comme dans cet exemple :

```
> D = { 'A' : 'C', 'B' : 'A', 'C' : 'B' }
> invertDict(D)
{'B': 'C', 'C': 'A', 'A': 'B'}
```

Exercice 6 : Spécifier puis écrire la fonction `decrypte(txt, perm)` qui décrypte le texte `txt` qui a été crypté à l'aide du dictionnaire de permutation `perm`.

Exercice 7 : Vous devez maintenant écrire un programme principal qui permet de crypter et décrypter des textes. Voilà un exemple d'interaction :

```
moi@maMachine:~/algo2/semaine2$ python3 codage.py
1 : Liste des codes disponibles.
2 : Créer un nouveau code.
3 : Crypter un texte.
4 : Décrypter un texte.
0 : Quitter l'application.
Votre choix : 2
Donnez le nom de votre nouveau code : prems
Le nouveau code est : {'U': 'N', 'D': 'V', 'E': 'G', 'F': 'J', 'M': 'U'...}
1 : Liste des codes disponibles.
2 : Créer un nouveau code.
3 : Crypter un texte.
4 : Décrypter un texte.
0 : Quitter l'application.
Votre choix : 2
Donnez le nom de votre nouveau code : deuz
Le nouveau code est : {'U': 'H', 'D': 'O', 'E': 'E', 'F': 'J', 'M': 'W'...}
1 : Liste des codes disponibles.
2 : Créer un nouveau code.
3 : Crypter un texte.
4 : Décrypter un texte.
0 : Quitter l'application.
Votre choix : 3
Liste des codes disponibles :
prems : {'U': 'N', 'D': 'V', 'E': 'G', 'F': 'J', 'M': 'U', 'J': 'H',...}
deuz : {'U': 'H', 'D': 'O', 'E': 'E', 'F': 'J', 'M': 'W', 'J': ' ',...}
--
Nom du code choisi : prems
Texte à crypter : bonjour a tous
Le texte crypté est :
QLSHLNMDPDFLNI
1 : Liste des codes disponibles.
2 : Créer un nouveau code.
3 : Crypter un texte.
4 : Décrypter un texte.
0 : Quitter l'application.
Votre choix : 4
Liste des codes disponibles :
... <ici les codes disponibles>
--
Nom du code à choisir : prems
Texte à décrypter : QLSHLNMDPDFLNI
Le texte décrypté est :
BONJOUR A TOUS
1 : Liste des codes disponibles.
2 : Créer un nouveau code.
```

```
3 : Crypter un texte.
4 : Décrypter un texte.
0 : Quitter l'application.
Votre choix : 7
Votre choix : 0
Bye
```

Vous devez écrire une fonction pour chacune des actions du menu :

Q 1 . Spécifier et écrire la fonction `affiche_menu` qui affiche toutes les options du menu principal et retourne le choix de l'utilisateur.

Q 2 . Spécifier puis écrire la fonction `afficher_les_codes` qui, à partir d'un dictionnaire qui, à un nom, associe un code, affiche tous les codes disponibles.

Q 3 . Spécifier puis écrire la fonction `ajouter_un_code` qui prend en paramètre le dictionnaire de tous les codes et en créé un nouveau.

Q 4 . Spécifier puis écrire les fonctions `crypter_un_texte` et `decrypter_un_texte`.

Q 5 . Écrire le script principal. Comment feriez-vous pour prendre en compte les espaces dans un texte ? les minuscules et les majuscules ?