LS4 - Travaux pratiques 3

16 février 2014

1 Prendre le langage en main

1.1 Structures de données Python

Donner la structure de donnée associée à `a` dans les instructions suivantes 7 :

```
a = {}
```

q01 :

- ☐ Chaîne de caractères (str)
- ☐ Tableau mutable (list)
- ☐ Tableau non mutable (tuple)
- ☐ Ensemble (set)
- ☐ Table d'association (dict)

```
a = ()
```

q02 :

- ☐ Chaîne de caractères (str)
- ☐ Tableau mutable (list)
- ☐ Tableau non mutable (tuple)
- ☐ Ensemble (set)
- ☐ Table d'association (dict)

```
a = []
```

q03 :

- ☐ Chaîne de caractères (str)
- ☐ Tableau mutable (list)
- ☐ Tableau non mutable (tuple)
- ☐ Ensemble (set)
- ☐ Table d'association (dict)

```
a = ''
```

q04 :

- ☐ Chaîne de caractères (str)
- ☐ Tableau mutable (list)
- ☐ Tableau non mutable (tuple)
- ☐ Ensemble (set)
- ☐ Table d'association (dict)

```
a = 5, 3, 'bla', [1,2,3]
```

q05 :

- ☐ Chaîne de caractères (str)
- ☐ Tableau mutable (list)
- ☐ Tableau non mutable (tuple)
- ☐ Ensemble (set)
- ☐ Table d'association (dict)

```
a = 'bla'
```

q06 :

- ☐ Chaîne de caractères (str)
- ☐ Tableau mutable (list)
- ☐ Tableau non mutable (tuple)
- ☐ Ensemble (set)
- ☐ Table d'association (dict)

```
a = 'bla',
```

q07 :

- ☐ Chaîne de caractères (str)
- ☐ Tableau mutable (list)
- ☐ Tableau non mutable (tuple)
- ☐ Ensemble (set)
- ☐ Table d'association (dict)

```
a = ['bla']
```

q08 :

- ☐ Chaîne de caractères (str)
- ☐ Tableau mutable (list)
- ☐ Tableau non mutable (tuple)
- ☐ Ensemble (set)
- ☐ Table d'association (dict)

```
a = {'bla'}
```

q09 :

- ☐ Chaîne de caractères (str)
- ☐ Tableau mutable (list)
- ☐ Tableau non mutable (tuple)
- ☐ Ensemble (set)
- ☐ Table d'association (dict)

```
t = [1, 3, 5, 8]
a = set(t)
```

q0a :

- ☐ Chaîne de caractères (str)
- ☐ Tableau mutable (list)
- ☐ Tableau non mutable (tuple)
- ☐ Ensemble (set)
- ☐ Table d'association (dict)

```
t = [1, 3, 5, 8]
a = {e:i for i, e in enumerate(t)}
```

q0b :

- ☐ Chaîne de caractères (str)
- ☐ Tableau mutable (list)
- ☐ Tableau non mutable (tuple)
- ☐ Ensemble (set)
- ☐ Table d'association (dict)

```
d = {'mot':3, 'mots':4, 'maux:4}
a = {f for f in d.values()}
```

q0c :

- ☐ Chaîne de caractères (str)
- ☐ Tableau mutable (list)
- ☐ Tableau non mutable (tuple)
- ☐ Ensemble (set)
- ☐ Table d'association (dict)

```
d = {'mot':3, 'mots':4, 'maux:4}
a = [f for f in d.keys()][0]
```

q0d :

- ☐ Chaîne de caractères (str)
- ☐ Tableau mutable (list)
- ☐ Tableau non mutable (tuple)
- ☐ Ensemble (set)
- ☐ Table d'association (dict)

```
d = {'mot':3, 'mots':4, 'maux:4}
a = [(m,f) for m, f in d.items()][-1]
```

q0e :

- ☐ Chaîne de caractères (str)
- ☐ Tableau mutable (list)
- ☐ Tableau non mutable (tuple)
- ☐ Ensemble (set)
- ☐ Table d'association (dict)

```
a = 'bla', [1, 3], {'a', 'b'}
```

q0f :

- ☐ Chaîne de caractères (str)
- ☐ Tableau mutable (list)
- ☐ Tableau non mutable (tuple)
- ☐ Ensemble (set)
- ☐ Table d'association (dict)

```
t = 'bla', [1, 3], {'a', 'b'}
a, b, c = t
```

q0g :

☐ Chaîne de caractères (str)
☐ Tableau mutable (list)
☐ Tableau non mutable (tuple)
☐ Ensemble (set)
☐ Table d'association (dict)

```
t = 'bla', [1, 3], {'a', 'b'}
a = t[2]
```

q0h :
☐ Chaîne de caractères (str)
☐ Tableau mutable (list)
☐ Tableau non mutable (tuple)
☐ Ensemble (set)
☐ Table d'association (dict)

1.2 Ensembles

Les exercices suivants portent sur les ensembles (structure de donnée `set`). Vous pouvez vous référer à la librairie standard de python stdtypes [1].

Question 1

Sans utiliser la fonction prédéfinie `set`, écrire une fonction `ens` qui prend en argument une chaîne de caractères et retourne l'ensemble de ses caractères.
Par exemple :
`ens("caracteres")` retourne `{'a', 'c', 'e', 's', 'r', 't'}`
`ens("")` retourne `set()`
q1a :

Question 2

Écrire une fonction `commun` qui calcule le nombre de lettres communes à deux chaînes de caractères.
Par exemple :
`commun("chaine", "caractere")` retourne `3`
`commun("ah", "ah ah ah")` retourne `2`
q1b :

Question 3

En utilisant le mécanisme de compréhension, écrire une fonction `consonnes` qui retourne l'ensemble des consonnes d'une chaîne de caractères.
Par exemple :
`consonnes("caractere")` retourne `{'c','r','t'}`
`consonnes("ah")` retourne `{'h'}`
`consonnes("ou")` retourne `set()`
q1c :

1.3 Tables d'associations

Les exercices suivants portent sur les tables d'associations (structure de donnée `dict`). Vous pouvez vous référer à la librairie standard de python stdtypes [2].

1. `http://docs.python.org/3.3/library/stdtypes.html#set-types-set-frozenset`
2. `http://docs.python.org/3.3/library/stdtypes.html#dict`

Question 1

1. Écrire une fonction `occ` qui prend en argument une chaîne de caractères et renvoie une table associant à ses caractères leur nombre d'occurence en utilisant la méthode count [3] des chaînes de caractères et le mécanisme de compréhension des tables d'association.
Par exemple :
`occ("caractere")` retourne `{'a': 2, 'c': 2, 'r': 2, 'e': 2, 't': 1}`
`occ("Une chaine, une cannette")` retourne :
`{'a': 2, ' ': 3, 'c': 2, 'e': 5, 'i': 1, 'h': 1, ',': 1, 'n': 5, 'u': 2, 't': 2}`
`occ("")` retourne `{}`

q2a :

2. Écrire une fonction `freq` qui prend en argument une chaîne de caractères et renvoie une table associant à ses caractères leur fréquence.
Par exemple :
`freq("caracteres")` retourne `{'a': 0.2, 'c': 0.2, 'e': 0.2, 's': 0.1, 'r': 0.2, 't': 0.1}`
`freq("Une chaine")` retourne :
`{'a': 0.1, ' ': 0.1, 'c': 0.1, 'e': 0.2, 'i': 0.1, 'h': 0.1, 'n': 0.2, 'u': 0.1}`
`freq("")` retourne `{}`

q2b :

Question 2

Dans le module `grandes_lettres.py` vous trouverez un dictionnaire et des fonctions permettant d'afficher des lettres sous la forme d'étoiles.

Voir grandes_lettres.py. [4] 1. Écrire une fonction `ligne` qui prend en arguments un numéro de ligne `n` et une lettre `c` minuscule et non accentuée, ou un espace et qui retourne la ligne verb[n] de la représentation de la lettre [verb[c]].
Par exemple :
`ligne(0,'t')` retourne la chaîne de caractères `"*****"`
`ligne(1,'t')` retourne la chaîne de caractères `"  *  "`
`ligne(2,'t')` retourne la chaîne de caractères `"  *  "`
`ligne(3,'t')` retourne la chaîne de caractères `"  *  "`
`ligne(4,'t')` retourne la chaîne de caractères `"  *  "`

q2c :

2. Écrire une fonction `grand_message` qui prend en argument une chaîne de caractères et réécrit le texte en grandes lettres. Pour cela, il faudra imprimer les étoiles ligne à ligne.
Par exemple :
`grand_message("toto")` imprime :

```
*****  **  *****  **
  *   *  *   *   *  *
  *   *  *   *   *  *
  *   *  *   *   *  *
  *    **    *    **
```

q2d :

3. `http://docs.python.org/3.3/library/stdtypes.html#str.count`
4. `Fichier fourni avec le sujet.`

Question 3

Voici un dictionnaire représentant une partie du réseau ferré français.

```
train = { "Paris": ["Lyon","Marseille","Bordeaux","Nantes","Toulouse","Lille","Nancy"],
          "Lyon": ["Marseille","Nancy","Paris"],
          "Marseille": ["Toulouse","Lyon","Paris"],
          "Bordeaux": ["Nantes","Toulouse"] ,
          "Nantes": ["Paris"],
          "Toulouse": ["Marseille", "Bordeaux"],
          "Lille": ["Paris","Nancy"],
          "Nancy": ["Lille","Lyon"]
          }
```

Écrire une fonction `voyage` qui calcule la liste des trajets entre deux villes ayant au plus un changement.
Par exemple,
`voyage("Paris","Toulouse")` renvoie la liste `[['Paris', 'Marseille', 'Toulouse'], ['Paris', 'Bordeaux', 'Toulo`
`voyage("Toulouse","Paris")` renvoie la liste `[['Toulouse', 'Marseille', 'Paris']]`
`voyage("Nantes","Toulouse")` renvoie la liste `[['Nantes', 'Paris', 'Toulouse']]`
`voyage("Lille","Lyon")` renvoie la liste `[['Lille', 'Paris', 'Lyon'], ['Lille', 'Nancy', 'Lyon']]`

q2e :

1.4 Fonctions, chaînes et tableaux mutables

Compréhension de tableaux mutables et tranches
split [5] join [6] strip [7] replace [8] lower [9]

Question 1

Un mot est une chaîne de caractères non vide et ne contenant pas d'espace.
Écrire une fonction `decoupe` qui prend en argument une chaîne de caractères et qui retourne le tableaux des mots de la chaîne.
Par exemple :
`decoupe(" Voici une phrase. ")` retourne `['Voici','une','phrase.']`
`decoupe("un-mot")` retourne `['un-mot']`
`decoupe(" ")` retourne `[]`

q3a :

Écrire une fonction `decoupe2` qui prend en argument une chaîne de caractères et qui retourne le tableaux des mots de la chaîne sans caractères de ponctuations (`.,;:?-()!`).
Par exemple :
`decoupe2(" Voici une phrase. ")` retourne `['Voici','une','phrase']`
`decoupe2("un-mot")` retourne `['un','mot']`
`decoupe2(" ")` retourne `[]`

q3b :

Écrire une fonction `decoupe3` qui prend en argument une chaîne de caractères et qui retourne le tableaux des mots de la chaîne sans caractères de ponctuations (`.,;:?-()!`) et en minuscule.
Par exemple :
`decoupe3(" Voici une phrase. ")` retourne `['voici','une','phrase']`

5. http://docs.python.org/3.3/library/stdtypes.html#str.split
6. http://docs.python.org/3.3/library/stdtypes.html#str.join
7. http://docs.python.org/3.3/library/stdtypes.html#str.strip
8. http://docs.python.org/3.3/library/stdtypes.html#str.replace
9. http://docs.python.org/3.3/library/stdtypes.html#str.lower

`decoupe3("un-mOt")` retourne `['un','mot']`
`decoupe3(" ")` retourne `[]`

q3c :

Question 2

On numérote les mots d'une chaîne de caractères en commençant par le numéro 1.
Écrire une fonction `mots_i_pair` qui prend en argument une chaîne de caractère et renvoie les mots de numéros pairs.
Par exemple :
`mots_i_pair(" Voici une phrase. ")` retourne `['une']`
`mots_i_pair("un-mot")` retourne `['mot']`
`mots_i_pair("Oh, le petit chat est mort.")` retourne `['le', 'chat', 'mort']`

q4a :

Écrire une fonction `mots_pairs` qui prend en argument une chaîne de caractère et renvoie les mots de longueurs pairs.
Par exemple :
`mots_pairs(" Voici une phrase. ")` retourne `['phrase']`
`mots_pairs("un-mot")` retourne `['un']`
`mots_pairs("Oh, le petit chat est mort.")` retourne `['oh', 'le', 'chat', 'mort']`

q4b :

Écrire une fonction `mots_occ` qui compte le nombre d'occurences d'un mot dans une chaine
Par exemple, si `s` désigne la chaîne de caractères suivante,
"""Il était une fois,
Dans la ville de Foix,
Une marchande de foie,
Qui vendait du foie...
Elle se dit : Ma foi,
C'est la première fois
Et la dernière fois,
Que je vends du foie,
Dans la ville de Foix"""
`mots_occ(s, 'foie')` retourne `2`
`mots_occ(s, 'foix')` retourne `2`
`mots_occ(s, 'fois')` retourne `4`
`mots_occ(s, 'foi')` retourne `1`

q4c :

Les exercices suivants portent sur les chaînes de caractères (structure de donnée `str`). Vous pouvez vous référer à la librairie standard de python stdtypes [10].

1.5 Interagir avec les fichiers

Les exercices suivants portent sur les fichiers (structure de donnée `set`). Vous pouvez vous référer à la librairie standard de python stdtypes [11]. Vous utiliserez notamment la fonction open [12] pour accéder aux fichiers.
Utiliser les mécanismes de tranchage [13] et de compréhension [14] de tableaux mutables pour répondre aux questions.

10. http://docs.python.org/3.3/library/stdtypes.html#text-sequence-type-str
11. http://docs.python.org/3.3/tutorial/inputoutput.html#reading-and-writing-files
12. http://docs.python.org/3.3/library/functions.html#open
13. http://docs.python.org/3.3/library/stdtypes.html#common-sequence-operations
14. http://docs.python.org/3.3/tutorial/datastructures.html#list-comprehensions

Question 1

Écrire une fonction `derniers` qui prend en argument un nom de fichier et renvoie la liste des derniers mots de chaque ligne.
Par exemple, si le fichier `fichier.txt` contient le texte suivant
Il était une fois,
Dans la ville de Foix,
Une marchande de foie,
Qui vendait du foie...
Elle se dit : Ma foi,
C'est la première fois
Et la dernière fois,
Que je vends du foie,
Dans la ville de Foix
`derniers("fichier.txt")` retourne `['fois', 'Foix', 'fois', 'foie', 'foi', 'fois', 'fois', 'foie', 'Foix']`

q5a :

Écrire une fonction `colonne` qui renvoie la chaine des iemes lettres de chaque ligne
Par exemple :
`colonne("fichier.txt", 1)` retourne `'IDUQECEQD'`
`colonne("fichier.txt", 2)` retourne `'lanul'tua'`
`colonne("fichier.txt", 0)` retourne `'indice trop petit ou trop grand'`
`colonne("fichier.txt", 20)` retourne `'indice trop petit ou trop grand'`

q5b :

Question 2 : La disparition

Écrire un script `disparition.py` qui, prenant en entrée un nom de fichier, en crée une copie dans laquelle toutes les occurrences du caractère `e` ont été supprimées. Attention à la casse et aux occurrences des variantes de `e` (`eéÉèÈêÊ`)
Par exemple :
La commande `./disparition.py fichier.txt` créer un fichier "fichier_sans_e.txt" qui contient le texte suivant :
Il tait un fois,
Dans la vill d Foix,
Un marchand d foi,
Qui vndait du foi...
ll s dit : Ma foi,
C'st la prmir fois
t la drnir fois,
Qu je vnds du foi,
Dans la vill d Foix
La commande `./disparition.py` imprime sur la sortie standard le texte suivant :
'préciser le nom du fichier'
Utiliser le module [15] sys [16] et la méthode

q6 :

15. `http://docs.python.org/3/tutorial/modules.html`
16. `http://docs.python.org/3/library/sys.html`

Question 3 : Cent mille milliards de poèmes de Raymond Queneau

Le fichier `vers-queneau.txt` contient une séquence de 14 groupements de 10 vers séparés par des lignes vides.

Voir vers-queneau.txt. [17] 1. Écrire une fonction `charger` qui charge le fichier`vers-queneau.txt` dans un dictionnaire associant aux entiers de 0 à 13, le tableau mutable des vers de chaque groupement.
2. Écrire une fonction `generer` telle que `generer(i,'vers-queneau.txt)` écrit dans un fichier `i-poeme.txt` un poème de 14 vers en prenant le ième vers de chaque groupement. Ne pas oublier de signer Raymond Queneau à la fin du poème.
3. Écrire une fonction `generer_alea` telle que `generer_alea('vers-queneau.txt)` écrit dans un fichier `alea-poeme.txt` un poème de 14 vers en prenant un vers aléatoirement dans chaque groupement. Utiliser pour cela le module random [18] et sa fonction choice [19] de la bibliothèque standard de python.

q7 :

2 Programmer

2.1 Traducteur automatique

On souhaite écrire un programme de traduction automatique simpliste. Le lexique de traduction est stocké dans un fichier dont chaque ligne est de la forme `mot:traduction`(on suppose que `traduction` est une chaîne de caractère sans espace).

Questions

1. Écrire une fonction `read_lexic` prenant comme argument le nom d'un fichier et renvoyant une table d'association dont chaque couple clé :valeur correspond à un mot du lexique et à sa traduction.
2. Écrire une fonction `translate_word` prenant un mot et un lexique et renvoyant sa traduction selon ce lexique.
3. Écrire une fonction `add_word` prenant deux mots et un lexique et ajoutant le mot et sa traduction au lexique.
4. Écrire une fonction `write_lexic` réciproque de la fonction verb[read_lexic]. Améliorer cette fonction en faisant en sorte que les entrées soient classées par ordre alphabétique.
5. Écrire une fonction principale qui demande à l'utilisateur un mot et lui fournit la traduction si elle existe dans le lexique. Si elle n'existe pas, il demande à l'utilisateur de donner la traduction et la sauvegarde dans le dictionnaire.
6. Écrire un fichier contenant un lexique de base et utilisez le pour tester votre script.

q8 :

2.2 Jeux de géographie

Le but de cet exercice est de créer un script `capitales` qui permet de tester ses connaissances sur les capitales du monde. On pourra l'utiliser dans un `terminal` de la façon suivante :

17. `Fichier fourni avec le sujet.`
18. `http://docs.python.org/3.3/library/random.html`
19. `http://docs.python.org/3.3/library/random.html#random.choice`

```
./capitales -v ville
    -> le script demande a l'utilisateur le nom du pays dont la capitale
    est ville, puis verifie la reponse
./capitales -n
    -> le script  donne un nom de  capitale et le joueur
    doit trouver le nom de pays associé. Le joueur peut
    decider de continuer ou d'arreter, a la fin il obtient une
    statistique sur le nombre de bonne reponse.
```

Questions

1. Écrire une fonction "file_to_dic" qui convertit un fichier contenant des pays et leur capitale en un dictionnaire. Tester cette fonction sur le fichier `capitales.txt` que vous trouverez sur Didel.

Voir capitales.txt. [20] 2. Écrire une fonction pour chacune des options `{-v, -n}` (on s'appuiera pour la dernière sur le module random [21]).

3. Écrire la partie principale du code qui selon les options appelle la bonne fonction.

q9 :

20. `Fichier fourni avec le sujet.`
21. `http://docs.python.org/3.3/library/random.html`