RÉSUMÉ 16 - STOCKER DANS UN DICTIONNAIRE

Lien vers l'activité : Stocker dans un dictionnaire
www.infoforall.fr - Dernière modif. : 19 07 2020

16.1 DICTIONNAIRE

A - CRÉER UN DICTIONNAIRE

```
1  dicoFR_AN = {'encodage' : "encoding", 'cryptage' : "encryption"}
```

Pour plus de clarté, on pourrait taper le code ainsi :

```
1  dicoFR_AN = {
2      'encodage' : "encoding",
3      'cryptage' : "encryption"
4  }
```

B - NOMBRE DE COUPLES CLÉ-VALEUR

On utilise la fonction len comme avec les tableaux ou les strings.

```
>>> mon_dico = {'encodage' : "encoding", 'cryptage' : "encryption"}
>>> nbr = len(mon_dico)
>>> nbr
2
```

C - RAJOUTER OU METTRE À JOUR UN COUPLE CLÉ-VALEUR

```
>>> mon_dico['balise'] = "tag"
>>> mon_dico
{'encodage': 'encoding', 'cryptage': 'encryption', 'balise': 'tag'}
```

D - ACCÈS AUX VALEURS AVEC LA CLÉ

```
>>> mon_dico['boucle']
'loop'
```

E - MUTABLE

Les dictionnaires sont mutables : on peut **modifier le contenu** sans avoir à recréer un nouveau dictionnaire. On peut donc garder le même identifiant.
Transmettre un dictionnaire via un paramètre le rend donc modifiable **par effet de bord**.

F - TEST D'EXISTENCE D'UNE CLÉ (AVANT LECTURE, POUR ÉVITER UNE LEVÉE D'EXCEPTION)

```
>>> 'boucle' in mon_dico
True
>>> 'toto' in mon_dico
False
```

Attention, avec un dictionnaire, on teste la clé, pas la valeur. Pour être plus explicite, on a accès à cette syntaxe, plus claire mais plus longue : ::python >>> 'boucle' in mon_dico.keys() True >>> 'toto' in mon_dico.keys() False

G - EXEMPLE DE FONCTION-TEST

```python
def incrementer(leDict, laCle) :
    '''Fonction qui renvoie True si la clé existe déjà et l'incrémente.
    Sinon, elle renvoie False et crée une valeur associée de 0

    :: param leDict(dic) :: le dictionnaire sur lequel on veut agir
    :: param laCle(?) :: la clé qu'on veut tester
    :: return (bool) :: True si leDict[laCle] existe

    :: exemples ::
    >>> nbr_lettres = {'a':45, 'b':12, 'd':5}
    >>> incrementer(nbr_lettres, 'z')
    False
    >>> nbr_lettres
    {'a': 45, 'b': 12, 'd': 5, 'z': 1}

    '''
    if laCle in leDict.keys() :
        leDict[laCle] = leDict[laCle] + 1  # ou leDict[laCle] += 1
        return True
    leDict[laCle] = 1  # return ligne 18 : si on arrive ici c'est que c'est faux
    return False

comptesLettres = {'a':1, 'b':20, 'c':300}

if __name__ == '__main__' :
    import doctest
    doctest.testmod()
```

16.2 FICHIER TEXTE

- Il faut ouvrir le fichier avec la fonction **open**.
- l'argument 'r' (read) signale qu'on ne donne qu'un accès lecture : on ne pourra pas le modifer.
- l'argument 'w' (write) signalerait qu'on ne donne qu'un accès écriture : on détruit le contenu précédent et on écrit un nouveau fichier.
- l'argument 'a' (append) signalerait qu'on ne donne qu'un accès rajout : on écrit à la suite du contenu précédent. Si le fichier n'existe pas, il sera créé.
- l'argument nommé **encoding** permet de signaler quelle technique d'encodage/décodage des octets utilisés. Ici, il s'agit du standard actuel 'utf-8'.
- la méthode **close** permet ensuite de fermer le fichier. Comme d'habitude avec les méthodes, on a le NOM de l'objet, suivi d'un POINT, suivi du nom de la METHODE. Mal fermer un fichier peut poser problème. Voir la méthode **with ... as** sinon.

Voici un exemple qui extrait une par une les lignes d'un fichier. Pour chacun des lignes, on affiche ensuite les caractères un par un.

```python
monFichier = open('dunwich_horror-lovecraft.txt', 'r', encoding='utf-8')

for ligne in monFichier :
    for caractere in ligne :
        print(caractere)

monFichier.close()
```

16.3 COMPLÉMENTS SUR LES STRINGS

A - CHAÎNES CONSTANTES DÉJÀ PRÉSENTES DANS PYTHON

```
>>> import string
>>> string.ascii_letters
'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'
>>> string.punctuation
'!"#$%&\'()*+,-./:;<=>?@[\]^_`{|}~'
```

B - QUELQUES MÉTHODES DES STRINGS

Attention, puisque les strings sont non mutables, ces méthodes ne TRANSFORMENT pas le string. Elles renvoient un nouveau string qu'il faut éventuellement stocker dans une variable portant le même nom.

- Mettre en minuscules : `caractere.lower()`
- Mettre en majuscules : `caractere.upper()`
- Tester la présence d'un caractère ou séquence dans une chaine : `if sequence in chaine_totale :`
- Remplacer tous les caractères d'un string par un autre :
  - `!` en `a` : `a = a.replace('!', 'a')`
  - suppression des passages à la ligne (code 10 en ASCII) : `a = a.replace('\n', ' ')`
  - suppression des retour-chariots (code 13 en ASCII) : `a = a.replace('\r', ' ')`

C TRANSFORMATION D'UN STRING EN TABLEAU EN PRÉCISANT UN SEPARATEUR (ICI LES ESPACES) :

```
>>> a = "Hello, it's a small world !\n"
>>> tableau_mots = a.split(' ')
>>> tableau_mots
['Hello,', "it's", 'a', 'small', 'world', '!\n']
```