TD

October 14, 2015

0.1 Chaines de caractères

Que fait ce code:

```
>>> print( '3,2,1,0,-1,-2,-3'.split('-') )
>>> print( '3,2,1,0,-1,-2,-3'.split(',') )
>>> print( '
>>> s = 'bonjour toi'
>>> print( 'm'.join( s.split('t') ) )
```

0.2 politesse

Ecrivez une fonction `politesse(s)` qui prenne en paramètre une chaine de caractère. La fonction devra insérer dans cette chaine le mot 'Monsieur' à la position du premier espace, et renvoyer le résultat. Exemple d'utilisation:

```
>>> print( politesse('bonjour Pierre') )
bonjour Monsieur Pierre
```

0.3 Pluriel

Vous disposez d'une liste de couples de mots mal orthographiés associés à leur orthographe correcte. Par exemple

```
ORTHO = [('chous','choux'),('pin','pain'),('dinner','diner'),('ve','veux')]
```

Ecrivez une fonction `corrige` qui prenne en paramètre cette liste et une phrase composée de mots séparés par des espaces, et qui renvoie la phrase avec l'orthographe corrigé. Par exemple:

```
>>> corrige('je ve des chous et du pin pour le dinner',ORTHO)
je veux des choux et du pain pour le diner
```

0.4 Split avec plusieurs séparateurs possibles

On ne peut pas utiliser la fonction `split` sur une chaine dont les mots ne sont pas séparés par le même caractère.

Par exemple, impossible d'appliquer `split` sur la chaine `'A,B C'` pour obtenir la liste `['A','B','C']`. En effet, voici ce que `split` donne sur cette chaine avec différents séparateurs:

```
>>> 'A,B C'.split(',')
['A', 'B C']
>>> 'A,B C'.split(' ')
['A,B', 'C,']
>>> 'A,B C'.split(', ')
['A,B C']
```

Ecrivez une fonction `split(s,liste_sep)` qui pallie ce problème. Votre fonction permettra de faire cela:

```
>>> split('hello,jo.tu vas bien',[',','.',' '])
['hello','jo','tu','vas','bien']
```

1 Ensembles

Que fait ce code:

```
>>> s = set([1,2,3,3])
>>> print s
>>> s2= {'a','b','B','b',3,2}
>>> print(s & s2)
>>> print(s | s2)
>>> print( list(s - s2) )
```

1.1 ensembles et chaines

Qu'affiche ce code ?

```
>>> c = 0
>>> s = set('bas haut-haut-bas-gauche-haut-droite-bas'.split('-') )
>>> for i in s:
>>>     print(i)
>>>     c+=1
>>> print (c)
```

1.2 sous-sembles

Ecrire une fonction `AllCouples` qui prend un ensemble en paramètre en renvoie l'ensemble de tous les couples.

Par exemple

```
>>> AllCouples({'a','b','c'})
{('a','a'),('a','b'),('a','c'),('b',a'),('b','b'),('b','c')}
```

Ecrivez une fonction `AllDiffCouples`, qui renvoie tous les couples `(i,j)` tels que $i \neq j$ et `(j,i)` n'est pas renvoyé

```
>>> AllDiffCouples({'a','b','c'})
{('a','b'),('a','c'),('b','c')}
```

2 Tableaux Numpy

2.1 Triangle inclus

On suppose qu'un triangle est représenté par une liste de trois listes de coordonnées.

par exemple: `T = [[10,10],[40,20],[20,0]]`

Ecrire une fonction `TriangleInterieur(T)` qui utilise la somme de tableaux Numpy pour renvoyer le nouveau triangle dont les sommets sont exactement au milieu des cotés de `T`

Par exemple:

```
>>> TriangleInterieur( [[10,10],[40,20],[20,0]] )
[[25,15],[30,10],[15,5]]
```