

## Séquence de caractères: str

La chaîne de caractère a été présentée comme un type de base, mais elle a certaines caractéristiques des séquences. C'est aussi une collection ordonnée, de caractères.

```
ch = 'moi'
```

instruction	resultat	commentaire
<code>len(ch)</code>		
<code>ch[0]</code>		
<code>for car in ch:     print(car)</code>		
<code>ch[0] = 't'</code>		erreur de type ...

On peut créer une liste à partir d'une chaîne de caractère, avec la méthode `split`. On met en paramètre le caractère séparateur, comme par exemple l'espace ' ', ou la virgule ','.

```
1. phrase = 'je vous entend'  
2. L = phrase.split(' ')  
3. print(L)  
4. # affiche [ ... , ... , ... ]
```

On peut créer une phrase à partir d'une liste, avec la méthode `join`. On met la liste en paramètre. On précise le séparateur en début d'instruction.

```
1. L = ['je', 'vous', 'entend']  
2. phrase = ' '.join(L)  
3. print(phrase)  
4. # affiche . . .
```

## Lire, écrire dans un fichier

### Ouvrir un fichier: `open`

La fonction `open` va créer un objet-fichier (objfichier ici) auquel on peut appliquer des méthodes.

Il est ouvert dans le mode spécifié:

- 'r' : ouverture en mode lecture (**r**ead)
- 'w' : ouverture en mode écriture (**w**rite)
- 'a' : ouverture du fichier en mode ajout (**a**ppend). On écrit à la fin du fichier.

Pour ouvrir le fichier 'scores.txt' en lecture: `f = open('scores.txt', 'r')`

### Lire un fichier: `read` ou `readlines`

Une fois le fichier ouvert, on peut appliquer la méthode `read` pour récupérer son contenu:

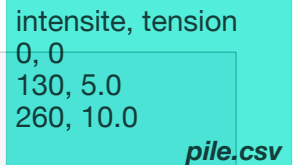
```
1. f = open("scores.txt", "r")  
2. t = f.read()  
3. print(t)  
4. # affiche l'intégralité du fichier:  
5. . . .  
6. f.close()
```

```
joueur, score  
Milena, 102  
Pavel, 99
```

score.txt

La méthode `readlines` permet d'identifier les valeurs et les associer à des variables ou des listes:

```
1. fichier = open('pile.csv','r') # ouverture du fichier de
   données
2. lignes = fichier.readlines()   # parcours du fichier par
   ligne
3.
4. lignes = lignes[1:]            # eliminer la premiere
   ligne qui contient les labels
5.
6. for ligne in lignes:
7.     intensite.append(float(ligne.split(',')[0]))
8.     tension.append(float(ligne.split(',')[1]))
9.
10. # intensite contient [ .., .., ..]
11. # tension contient [ .., .., ..]
12. fichier.close()
```



intensite	tension
0	0
130	5.0
260	10.0

pile.csv

## Transformer une liste en tableau


Pour des traitements mathématiques sur les valeurs de liste, (vecteurs, ...), il peut être utile de transformer la liste en tableau. On utilise alors la librairie `numpy`

```
1. import numpy as np
2. intensite = [0, 130, 260]
3. print(intensite * 2)
4. # affiche . . .
5. intensite=np.asarray(intensite) # changer le type en array
6. print(intensite)
7. # affiche array([ 0, 130, 260])
8. print(intensite * 2)
9. # affiche array([ 0, 260, 520])
```

## Ecrire dans un fichier: `write`

```
1. a = 10900
2. b = 19900
3. f = open('scores.txt', 'w')
4. f.write('score des joueurs : \n')
5. f.write('Milena : {} \n'.format(str(a)))
6. f.write('Pavel : {} \n'.format(str(b)))
7. f.close()
```

Le fichier contient alors :



```
score des joueurs : 
Milena : 10900 
Pavel : 19900 
```

scores.txt