CHAÎNES DE CARACTÈRES

Définition

Une chaîne de caractères est une suite finie de caractères consécutifs, qu'on note entre apostrophes ou guillemets ;

La chaîne vide se note : `''` ou `""`

Accès à un caractère:

On peut stocker une chaîne dans une variable :

```
>>> s = 'Bonjour'
```

Pour accéder à chacun des caractères on utilise son indice : **`s[i]`**

```
>>> s[2]
'n'

>>> s[-1]
'r'

>>> s[-2]
'u'

>>> s[-7]
'B'
```

Concaténation :

On concatène deux chaînes à l'aide de l'opérateur **`+`**

```
>>> s = 'Bonjour '+ 'lecteur !'
>>> s
'Bonjour lecteur ! '
```

Longueur :

On utilise la fonction **`len( )`** pour obtenir la longueur d'une chaîne :

```
>>> len('Bonjour')
7
```

Sous-chaîne :

Un ensemble de caractères consécutifs à l'intérieur d'une chaîne s'appelle une sous-chaîne. Ainsi, **`'lecteur'`** ou **`'jour lec'`** sont des sous-chaînes de **`'Bonjour lecteur !'`**.

Pour extraire une sous-chaîne de s, on écrit **`s[i:j]`** où **`i`** est l'indice du premier caractère de la sous-chaîne et **`j`** est l'indice du dernier caractère plus un.

```
>>> s = 'Bonjour lecteur !'
>>> s[0:7]
'Bonjour'

>>> s[8:15]
 'lecteur'
```

Si **`j < -len(ch)+i`** , il n'y a pas de sous-chaîne correspondante. Python renvoie alors la chaîne vide : `''`

Si **`j`** dépasse la longueur de la chaîne, la sous-chaîne s'arrête au dernier caractère de la chaîne.

Test d'appartenance :

Il est à noter qu'il est également possible de tester la présence d'une sous-chaîne dans une chaîne avec la même construction :

```
>>> 'lecteur' in 'Bonjour lecteur !'
True
>>> 'Bjr' in 'Bonjour lecteur !'
False
```

L'opérateur in sert à tester l'appartenance d'un caractère à une chaîne :

```
>>> 'o' in 'Bonjour`
True
```

Conversion:

On peut convertir une valeur d'un type simple vers une chaîne de caractères à l'aide de la fonction str(e) :

```
>>> str(1.2)
'1.2'
```

Il est possible de reconvertir une telle chaîne vers une valeur d'un type simple :

```
>>> int('123')
123
>>> float('1.2')
1.2
>>> bool('True')
True
>>> bool('True')
True
```

Les chaînes ne sont modifiables

```
>>> s = 'Bonjour lecteur !'
>>> s[0]= 'b'                    # erreur
TypeError: 'str' object does not support item assignment
```

Function ord () et chr()

- Pour obtenir le code **ascii** d'un caractère (l'ordre d'u caractère dans une liste normalisée appelée ASCII), il suffit d'appliquer la fonction ***ord(c)*** où ***c*** est le caractère en question. ***ord*** renvoie un nombre entre 0 et 65535. S'il s'agit d'un caractère **ascii**, le code sera compris entre 0 et 255.

```
>>> ord('A'),ord('B'),ord('C')
(65, 66, 67)
>>> ord('a'),ord('b'),ord('c')
(97, 98, 99)
>>> ord(' ')
32
>>> ord('à'),ord('è'),ord('é')
(224, 232, 233)
```

- Pour obtenir le caractère correspondant à un code ascii on utilise la fonction ***chr(i)*** où i est un nombre compris entre 0 et 255 qui représente le code ascii du caractère.

```
>>> chr(65),chr(66),chr(67)
('A', 'B', 'C')
>>> chr(97),chr(98),chr(99)
('a', 'b', 'c')
>>> chr(32)
' '
>>> chr(224),chr(232),chr(233)
('à', 'è', 'é')
```

Exemple d'utilisation de ord () et chr()

Ecrire la fonction qui converti une chaine en majuscule (sans utiliser la méthode **upper())**

```
def majuscule(ch):
      CH=''
      for c in ch:                          #pour chaque caractère de ch
            if c>='a' and c<='z':
                  code=ord(c)               #obtenir le code ascii du caractère
                  code=ordre-32             #32 =ord('a') - ord('A')
                  car=chr(code)             #obtenir le caractère correspondant
            else:
                  car=c
            CH+=car
      return CH
```

```
>>> majuscule ('abcde')
'ABCDE'

>>> chaine=majuscule ('abédà')
>>> print(chaine)
'ABéDà'
```

Les chaines sont des objets

Sous Python, les listes sont des objets pour les quels on peut appliquer un certain nombre de ***méthodes*** (fonctions) particulièrement efficaces. En voici quelques-unes :

- **convertir une chaine en majuscule ou en minuscule**

```
>>> s= 'cpge – agadir'
>>> s2=s.upper()
>>> s2
'CPGE - AGADIR'

>>> s2.lower()     #retourne la chaine convertie en minuscule
'cpge – agadir'
```

- **compter le nombre d'occurrences d'un caractère dans la chaine**

```
>>> s.count('a')
2
```

- **Chercher une chaine dans une autre**

```
>>> s.find('a')  #retourne l'index du caractère dans la chaine s'il existe.
7
>>> s.find('aga')
7
>>> s.find('agx') #retourne -1 si le caractère n'existe pas dans la chaine.
-1
```

- **Transformer une chaine en une liste**

```
>>> Ch='mot1 mot2 mot3 mot4'
>>> Ch.split()
['mot1', 'mot2', 'mot3', 'mot4']

>>> Ch='mot1,mot2,mot3,mot4'
>>> Ch.split(',')
['mot1', 'mot2', 'mot3', 'mot4']
```

- **Transformer une liste de chaines en une seule chaine**

```
>>> t = [ "Ceci" , "est" , "un" , "tableau" , "de mots"]
>>>' '.join(t)
'Ceci est un tableau de mots'
```

- **Supprimer un caractère à gauche et à droite d'une chaine**

```
>>> ch='AABBAACCAA'
>>> ch.strip('A')
'BBAACC'
```

- **Supprimer un caractère à gauche d'une chaine**

```
>>> ch='AABBAACCAA'
>>> ch.lstrip('A')
'BBAACCAA'
```

- **Supprimer un caractère à droite d'une chaine**

```
>>> ch='AABBCCAA'
>>> ch.rstrip('A')
'AABBAACC'
>>> S='Ali\n'
>>> S
'Ali\n'
>>> s.rstrip('\n')
'Ali'
```

- **Déterminer la nature d'une chaine S :**

- **S.isalnum()** : renvoit **True** si **S** est composé uniquement des caractères {abcd...z} {ABCD...Z} {012..9}
- **S.isalpha()** : renvoit **True** si **S** est composé uniquement des caractères {abcd...z} {ABCD...Z}
- **S.isdigit()** : renvoit **True** si **S** est composé uniquement des caractères {012..9}
- **S.islower()** : renvoit **True** si **S** est composé uniquement de caractères minuscules
- **S.isspace()** : renvoit **True** si **S** est composé uniquement d'espaces (blanc, tabulations, retour chariot..)
- **S.istitle()** : renvoit **True** si **S** est un titre c'est à dire que la première lettre de chaque mot est une majuscule.
- **S.isupper()** : renvoit **True** si **S** est composé uniquement de caractères majuscules