

NOTE 5

Retour sur les Structures de données complexes

Séquences

► Il existe 2 familles

Non modifiables :

- Chaines de caractères 'bonjour'
- Tuples (2,3)

Modifiables :

- Listes [2,3]
- Dictionnaires {'clef1':2, 'clef2':3}
- Ensembles

Cas particulier des no-modifiables (chaînes ou tuples) : on ne peut pas modifier le contenu d'une chaine ou d'un tuple en accédant à un élément par son indice

Opérations sur les Séquences

Opération	Résultat	Exemple	résultat
$x \text{ in } s$	<i>True</i> si $x \in s$, <i>False</i> sinon	3 in [1,2,3]	True
$x \text{ not in } s$	<i>True</i> si $x \notin s$, <i>False</i> sinon	3 not in [1,2,3]	False
$s + t$	Concaténation de s et t	[1,2,3]+[2,3]	[1,2,3,2,3]
$s * n \text{ ou } n * s$	Concaténation de s , n fois	[1,2,3]*2	[1,2,3,1,2,3]
$s[i]$	élément de s <i>au</i> $i^{\text{ème}}$ indice (0 est le premier)	A='bonjour' A[0] A[2] A[-1]	<i>b</i> <i>n</i> <i>r</i>
$s[i:j]$	Tranche entre i et j dans s (j <i>n'est pas compris</i>)	A[2:5]	<i>njo</i>
$s[i:j:pas]$	Tranche entre i et j dans s <i>par pas</i>	A[2:6:2]	<i>no</i>
$len(s)$	Longueur de s	$len(A)$	<i>7</i>
$min(s)$	Plus petit élément de s	$min(A)$	<i>b</i>
$max(s)$	Plus grand élément de s	$max(A)$	<i>u</i>

Chaînes de caractères

- ▶ Délimitation par des apostrophes, guillemets simples ou triples :
 - *'Vive "python"'*
 - *"L'ESIEE-IT"*
 - *""" L'ESIEE-IT avec ce cours "python" vous permet de """*
- ▶ Triples guillemets permettent le retour à la ligne dans les chaînes et à la documentation de code
- ▶ Caractères spéciaux : \n, \t, \b

Opération	Résultat	Exemple	résultat
<i>c in s</i> <i>C not in s</i>	<i>True</i> si caractère <i>c</i> est dans <i>s</i>	'a' in 'chaîne' 'ai' not in 'chaîne'	True False
<, >, ==, !=	Comparaisons alphanumériques	'abc' < 'chaîne'	True (abc est bien avant chaîne dans l'ordre du dictionnaire)
<i>zip(a,b)</i>	Colle chaînes <i>a</i> et <i>b</i>	<i>zip('abc','def')</i>	<i>'abcdef'</i>

Méthodes sur les Chaînes de caractères

Les fonctions/méthodes agissant sur les chaînes de caractères sont classées en plusieurs catégories :

- Gestion des espaces
- Gestion des majuscules et minuscules
- Test de la nature d'une chaîne
- Recherche et remplacement
- Découpage et collage de chaînes
- Formatage

L'appel à une méthode ne modifie pas la chaîne en elle-même

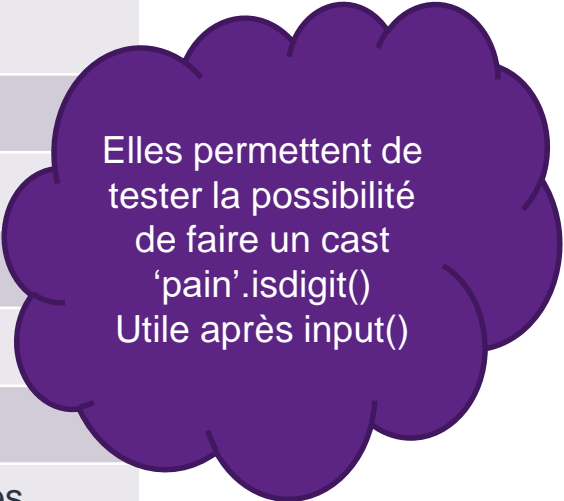
Une nouvelle chaîne est créée et renvoyée par les méthodes

Gestion des espaces et majuscules/minuscules

Méthode	Effet
<i>strip()</i>	Supprime les espaces en début et fin de chaîne
<i>rstrip()</i>	Supprime les espaces en fin de chaîne
<i>lstrip()</i>	Supprime les espaces en début de chaîne
<i>isspace()</i>	Renvoie <i>True</i> s'il n'y a que des espaces
<i>expandtabs(tabsize)</i>	Remplace les tabulations par des espaces
<i>upper()</i>	Remplace les minuscules par des majuscules
<i>lower()</i>	Remplace les majuscules par des minuscules
<i>capitalize()</i>	Passe la 1 ^{ère} lettre de la chaîne en majuscule et les autres en minuscules
<i>swapcase()</i>	Inverse les majuscules et minuscules

Nature de la chaîne

Méthode	Effet
<i>startswith(prefix)</i>	<i>True</i> si la chaîne commence par <i>prefix</i>
<i>endswith(prefix)</i>	<i>True</i> si la chaîne se termine par <i>prefix</i>
<i>isupper()</i>	<i>True</i> si la chaîne ne contient que des majuscules
<i>islower()</i>	<i>True</i> si la chaîne ne contient que des minuscules
<i>istitle()</i>	<i>True</i> si la chaîne est formatée comme un titre
<i>isdigit()</i>	<i>True</i> si la chaîne ne contient que des caractères numériques
<i>isalpha()</i>	<i>True</i> si la chaîne ne contient que des caractères alphabétiques
<i>isalnum()</i>	<i>True</i> si la chaîne ne contient que des caractères alphanumériques



Elles permettent de tester la possibilité de faire un cast
'pain'.isdigit()
Utile après input()

Ces fonctions sont des fonctions dites « objets », on doit mettre le nom de la chaîne à tester devant le nom de la fonction

Exemple :

a='Bonjour'

a.startswith('Bon')

Recherche/remplacement et découpage/collage

Méthode	Effet
<i>index(sub)</i>	Renvoie l'index de la 1 ^{ère} occurrence de la chaine <i>sub</i> , exception si pas trouvée
<i>rindex()</i>	Idem pour la dernière occurrence
<i>find(sub)</i>	Comme <i>index</i> mais renvoie -1 si la chaine n'est pas trouvée
<i>rfind()</i>	Idem pour la dernière occurrence
<i>count(sub)</i>	Retourne le nbr d'occurrence de la sous-chaine
<i>replace(old, new [,max])</i>	Remplace la sous-chaine <i>old</i> par <i>new</i> , au plus <i>max</i> fois (par défaut, remplace toutes les occurrences)
<i>split([sep[,max]])</i>	Crée une liste de chaine en découpant la chaîne suivant <i>sep</i> au plus <i>max</i> fois. Par défaut <i>split</i> utilise les espaces, tabulations les sauts de ligne comme séparateur
<i>join(list)</i>	Concatène tous les éléments de <i>list</i> en utilisant la chaine comme jointure. Tous les éléments de la liste doivent être des chaines de caractères.

Tuples et listes

- ▶ Les tuples sont des séquences non modifiables, entre parenthèses ou suite de valeurs séparées par des virgules :
 - (a, b, c)
 - $2,3$
 - $(1,)$ un tuple avec un seul élément
 - $(1, ['chaine', 2, 4], '5')$ le tuple contient un entier, une liste et une chaine
- ▶ Les listes sont des séquences modifiables, entre `[]` :
 - $[a, (1,2), 'bonjour']$
 - $[(3,), (5,),]$ il est possible de mettre une virgule après dernier élément

Opérations sur les tuples et listes

Opération	Résultat
$len(x)$	Taille de la séquence x
$s[i]=x$	Affecte la valeur x au $i^{\text{ème}}$ élément de s
$del\ s[i]$	Supprime le $i^{\text{ème}}$ élément de s
$s[i:j]=t$	Remplace la tranche $i:j$ par la séquence t
$del\ s[i:j]$	Supprime la tranche $i:j$ de s
$s += t$	Ajoute le contenu de t à la fin de s
$u = s + t$	Renvoie une nouvelle liste u dont le contenu est la concaténation de s et de t

Méthodes sur les listes

Méthode	Résultat
<i>append(x)</i>	Ajoute x en fin de liste
<i>insert(i,x)</i>	Ajoute x à l'index i
<i>pop(i)</i>	Retourne et supprime l'élément d'index i ou le dernier si i non précisé
<i>remove(x)</i>	Supprime le 1 ^{er} élément de valeur x
<i>count(x)</i>	Retourne le nombre d'occurrences de x
<i>index(x)</i>	Retourne l'index du 1 ^{er} élément de valeur x
<i>reverse()</i>	Renverse la liste
<i>sort([func])</i>	Trie la liste. Une fonction de comparaison peut être donnée

Si vous copiez une liste dans une autre avec l'opérateur '=', les deux listes pointent sur la même zone mémoire, ce sont donc la même liste. Toute modification sera visible sur les deux

Pour copier une liste : `liste2= liste1[:]`

Parcourir une liste

```
liste=[1,2,3]
for i in liste :
    print(i)
```



Résultat

1
2
3

Le for parcourt
élément par
élément

```
liste=[1,2,3]
for i,valeur in enumerate(liste) :
    print(i+ ' ' et '+valeur)
```



C'est le même résultat

Résultat

0 et 1
1 et 2
2 et 3

On récupère des
tuples avec position
et valeur

```
liste=[1,2,3]
for i in range(len(liste)) :
    print(i+ ' ' et '+liste[i])
```



On récupère les
positions et avec,
on retrouve la
valeur

Résultat

0 et 1
1 et 2
2 et 3

```
listedetuples=[('a',1), ('b',2), ('c',3)]
for lettre,chiffre in listedetuples :
    print(lettre+ ' ' et '+chiffre)
```



Résultat

Résultat

a et 1
b et 2
c et 3

Chaque élément de la liste est un tuple et
on utilise la multi-affectation pour le
« découper »

Les dictionnaires

Dans une liste, on a des positions (sous forme d'entiers) et des valeurs associées.



Dans un dictionnaire, on a des clefs (de type numérique, chaîne de caractères ou tuples) et des valeurs associées.

Liste=['a','b','c']

Positions 1, 2 et 3
Valeurs a,b,c

dico={1: 'a', 'deux': 'b', 3: 'c'}

clefs 1, 'deux' et 3
Valeurs a,b,c

Les valeurs sont de type quelconque (On peut faire une liste de dictionnaires)
L'ordre des éléments n'est pas conservé en manipulant les dictionnaires

Opération	Résultat
<i>len(a)</i>	Nombre de valeurs dans <i>a</i>
<i>a[k]</i>	Valeur de la clef <i>k</i>
<i>a[k] = v</i>	Assigne la valeur <i>v</i> à la clef <i>k</i>
<i>del a[k]</i>	Supprime la valeur <i>a[k]</i> de <i>a</i>
<i>k in a</i>	<i>True</i> si existe valeur associée à la clef <i>k</i>
<i>k not in a</i>	<i>True</i> si aucune valeur associée à la clef <i>k</i>

Méthodes sur les dictionnaires

Méthode	Effet
<code>keys()</code>	Renvoie la liste des clefs du dictionnaires
<code>values()</code>	Renvoie la liste des valeurs du dictionnaires
<code>items()</code>	Renvoie la liste des couples (<i>clefs:valeur</i>) du dictionnaires
<code>has_key()</code>	Vrai si existe valeur associée à la clef <i>k</i>
<code>get(k[,default])</code>	Retourne la valeur associée à la clef <i>k</i> ou default si pas de valeur
<code>clear()</code>	Vide le dictionnaire
<code>copy()</code>	Retourne une copie de surface du dictionnaire
<code>update(dict)</code>	Met à jour le dictionnaire à partir d'un autre dictionnaire

Si vous copiez un dictionnaire dans un autre avec l'opérateur '=', les deux dictionnaires pointent sur la même zone mémoire, ce sont donc le même dictionnaire. Toute modification sera visible sur les deux

Pour copier un dictionnaire : `d2= {}`
`d2.update(d1)`

Les ensembles

- ▶ Un *set* est une collection non ordonnée d'objets sans doublon
- ▶ Le module *sets* existe avec les deux classes suivantes :
 - *set* (modifiable)
 - et *immutableSet* (non modifiable)
- ▶ Un ensemble peut être créé à partir d'une liste, d'un dictionnaire, d'une chaîne ou un *tuple*.
- ▶ Si des doublons éventuels existent, ils seront alors absents de l'ensemble

```
>>liste=[1,2,1,5,3]
>>ensemble=set(liste)
>print(i)
```



Résultat
{1, 3 ,5}

La liste est convertie en ensemble,
les doublons disparaissent

```
>>chaine='bonjour'
>>ensemble=set(chaine)
>print(i)
```



Résultat
{'r', 'b', 'o', 'n', 'j', 'u'}

La chaîne est convertie en ensemble,
les doublons disparaissent

ESiEE[it]
L'école de l'expertise numérique



une école de la



www.esiee-it.fr