CHAPITRE 5 : LES SÉQUENCES (STR, LIST, TUPLE)

A. LES CHAÎNES DE CARACTÈRES

Une donnée de type chaîne de caractères est une suite de caractères quelconques. Une constante chaîne de caractères s'indique en écrivant les caractères en question soit entre **apostrophes**, soit entre **guillemets**.
L'encadrement par de **triples guillemets** ou de **triples apostrophes** permet d'indiquer des chaînes qui s'étendent sur plusieurs lignes.

I. Définir une chaîne :

a = "" #définir une chaîne vide print (a)	le script affiche :
a = str() #définir une chaîne vide print (a)	le script affiche :
a = "je suis une chaine" print (a)	le script affiche : **je suis une chaine**
a = 'je suis une chaine' print (a)	le script affiche : **je suis une chaine**
a = "j'ai bien compris" print (a)	le script affiche : **j'ai bien compris**
a = 'J\'ai toujours la même chose' print (a)	le script affiche : **J'ai toujours la même chose**
a = 'je suis\nune chaine\nsur plusieurs\nlignes ... ' print (a)	le script affiche : **je suis** **une chaine** **sur plusieurs** **lignes ...**
a = """je suis une chaine sur plusieurs lignes ... """ print (a)	le script affiche : **je suis** **une chaine** **sur plusieurs** **lignes ...**

II. Concaténation de deux chaînes : l'opérateur +

a = "MP" b = "PC" c = "SI" print (a+c) print (b+c)	le script affiche : **MPSI** **PCSI**

III. Répétition d'une chaîne : l'opérateur *

a = "cpge" b = 4 print (a*b)	le script affiche : **cpgecpgecpgecpge**

IV. Accès à un caractère individuel : chaîne [indice]

On accède aux caractères d'une chaîne en considérant celle-ci comme une séquence indexée par les entiers : le **premier** caractère a l'indice **0**, le second l'indice 1, le dernier l'indice **n-1**, (**n** étant le nombre de caractères dans la chaîne).

indice positif	0	1	2	3
	c	p	g	e
indice négatif	-4	-3	-2	-1

Exemple :

a = "cpge" print(**a[1]**) print(**a[-2]**)	le script affiche : **p** **g**

Pour obtenir la taille d'une chaîne (nombre de caractères) on utilise la fonction intégrée **len**

Exemples:

a = "moncpge" b = **len**(a) print (b)	le script affiche : **7**

a = "c'est mon cpge" print (**len**(a))	le script affiche : **14**

V. Parcourir les chaînes.

```
for var in chaîne :
        blocs d'instructions
```

```
for var in range(0,len(chaîne),1) :
        blocs d'instructions
```

Exemples :

a = "je suis une chaine" **for** i **in** a **:** print(i,end='')	le script affiche : **je suis une chaine**

a = "je suis une chaine" **for** i **in range**(**len**(a)) **:** print(a[i],end='')	le script affiche : **je suis une chaine**

VI. Tranche d'une chaîne : chaîne[debut : fin : pas]

indice positif	0	1	2	3	4	5	6
	B	o	n	s	o	i	r
indice négatif	-7	-6	-5	-4	-3	-2	-1

S="Bonsoir" print(**S[1:3]**) print(**S[3:]**) print(**S[:3]**) print(**S[3:3]**) print(**S[-3:-1]**)	le script affiche : **on** **soir** **Bon** **oi**

VII. Modification des chaînes.(N'est pas autorisée)

Il n'est pas possible de modifier une chaîne de caractères.

a = "Bonjour tout le monde." a[3] = "x"	**Traceback (most recent call last):** **File "<stdin>", line 1, in ?** **TypeError: object does not support item assignment**

VIII. Les autres méthodes des chaînes de caractères:dir(str)

• **find()** : Donne la position d'une sous-chaîne dans une chaîne.	a="Bonjour tout le monde." b=a.**find**('ou') print(b)	le script affiche : **4**
• **count()** : Compte le nombre de souschaînes dans la chaîne :	a="Bonjour tout le monde." b=a. **count** ('ou') print(b)	le script affiche : **2**
• **replace() :** Remplace une sous-chaîne par une autre.	a="Bonjour tout le monde." a=a.**replace**('tout le monde','cpge') print(a)	le script affiche : **Bonjour cpge.**
• **strip()** : Supprime les blancs inutiles en début et en fin de la chaîne	a=" cpge TANGER " a=a.**strip**() print(a)	le script affiche : **cpge TANGER**
• **lower()** : convertit une chaîne en minuscules.	a="CHAINE EN MAJUSCULES" a=a.**lower**() print(a)	le script affiche : **chaine en majuscules**
• **upper()** : convertit une chaîne en majuscules.	a="chaine en minuscules" a=a.**upper**() print(a)	le script affiche : **CHAINE EN MINUSCULES**
• **capitalize() :** Convertit la première lettre en majuscule.	a="chaine en minuscules" a=a.**capitalize**() print(a)	le script affiche : **Chaine en minuscules**
• **title()** : Convertit la première lettre de tous les mots en majuscule.	a="chaine en minuscules" a=a.**title**() print(a)	le script affiche : **Chaine En Minuscules**
• **swapcase() :** Intervertit les	a="MAJUSCULES - minuscules"	le script affiche :

majuscules et les minuscules	a=a.**swapcase**() print(a)	**majuscules - MINUSCULES**
• **isalnum()** : Renvoie la valeur vrai si la chaîne est constituée de lettres ou de chiffres et si len(s)>0.	a="ABC123abc" b=a.**isalnum**() print(b)	le script affiche : **True**
• **isalpha()** : Renvoie la valeur vrai si la chaîne est constituée de lettres et si len(s)>0.	a="ABCabc" b=a.**isalpha**() print(b)	le script affiche : **True**
• **isdigit()** : Renvoie la valeur vrai si la chaîne est constituée de chiffres et si len(s)>0	a="0123" b=a.**isdigit**() print(b)	le script affiche : **True**
• **islower()** : Renvoie la valeur vrai si la chaîne n'est constituée que de lettres minuscules.	a= "abcd" b=a.**islower**() print(b)	le script affiche : **True**
• **isupper()** : Renvoie la valeur vrai si la chaîne n'est constituée que de lettres majuscules.	a="ABCDE" b=a.**isupper**() print(b)	le script affiche : **True**
• **isspace()** : Renvoie la valeur vrai si la chaîne n'est constituée que d'espaces.	a= " " b=a.**isspace**() print(b)	le script affiche : **True**
• **istitle()** : Renvoie la valeur vrai si la chaîne est une séquence de titre, tous les mots commencent par une majuscule.	a="Bonjour A Tous." b=a.**istitle**() print(b)	le script affiche : **True**
• **center(n) :** Renvoie une copie de la chaîne centrée sur n caractères.	a= "Bonjour" a=a.**center**(15) print(a)	le script affiche : **Bonjour**
• **ljust(n)** : Renvoie une copie de la chaîne justifiée à gauche sur n caractères.	a="Bonjour" a=a.**ljust**(15) print(a)	le script affiche : **Bonjour**
• **rjust(n)** : Renvoie une copie de la chaîne justifiée à droite sur n caractères.	a="Bonjour" a=a.**rjust**(15) printa)	le script affiche : **Bonjour**

B. LES LISTES

Une liste est une suite de valeurs présentées entre crochets et séparées par des virgules. Elles peuvent contenir des types distincts de données.

Exemple :

```
a = [10,11,12,13,14,15,16]                  #a est une liste de données
b=['Nous', 'sommes', 'en', "l'an", 2014]    #b est une liste de données
```

I. Définir une liste :

a = [] #définir une liste vide print (a)	le script affiche : **[]**

a = list() #définir une liste vide	le script affiche :

print (a)	**[]**

a = **list(**range(2, 10, 2)**)** #définir la liste d'entiers **[2,4,6,8]**

a = [1, 12, 3, -4, 5] #définir une liste d'entiers
print(a)
	le script affiche :
	[1, 12, 3, -4, 5]

#définir une liste d'entiers et de chaînes
a = ['Nous', 'sommes', 'en', "l'an", 2015]
print (a)
	le script affiche :
	['Nous', 'sommes', 'en', "l'an", 2015]

#définir une liste qui contient comme élément une autre liste
a=[1, 4, 'cercle', 4, 'triangle', **['point', 6]**]
print (a)
	le script affiche :
	[1, 4, 'cercle', 4, 'triangle', ['point', 6]]

On peut définir une liste en utilisant une syntaxe proche d'une définition mathématique nommée : **List Comprehension** :

Exemples :

L=[i **for i in** range(10)] print(L)	le script affiche : **[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]**

L=[i**2 **for i in** range(10)] print(L)	le script affiche : **[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]**

L=[i **for i in** range(10) **if** i%2==0] print(L)	le script affiche : **[0, 2, 4, 6, 8]**

S= "voici une liste de mots"
L=S.**split()** #convertir la chaîne S en liste L
L1=**[** mot.upper() **for** mot **in** L**]**
print(L)
print(L1)
	le script affiche :
	['voici', 'une', 'liste', 'de', 'mots']
	['VOICI', 'UNE', 'LISTE', 'DE', 'MOTS']

Une liste est **une séquence** comme pour les chaînes de caractères :

II. Les opérateurs sur les listes :+ , * , in

II-1. Concaténation de deux listes : l'opérateur +

L1, L2 = [1, 2, 3], [4, 5] L1 += L2 print (L1)	le script affiche : **[1, 2, 3, 4, 5]**

II-2. Répétition d'une liste : l'opérateur *

a = [0,1] b = 4 print (a*b)	le script affiche : **[0, 1, 0, 1, 0, 1, 0, 1]**

II-3. l'existence d'une valeur dans une liste : l'opérateur in

On peut tester l'existence d'une valeur dans une liste en utilisant **in** :

L = ["MPSI",1,2,3, "PCSI",1] a= "MP" **in** L print(a)	le script affiche : **False**

III. Accès à un élément individuel : liste[indice]

On accède aux éléments d'une liste en considérant celle-ci comme une séquence indexée par les entiers : le **premier** élément a l'indice **0**, le second l'indice 1, le dernier l'indice **len(liste)-1**,

indice positif	0	1	2	3
	03	Février	2015	TANGER
indice négatif	-4	-3	-2	-1

Exemple :

L = [03," Février ",2015,"TANGER"] print(**L[1]**) print(**L[-2]**)	le script affiche : **Février** **2015**

IV. Tranche d'une liste : liste[debut : fin : pas]

indice positif	0	1	2	3	4	5	6
	10	11	12	13	14	15	16
indice négatif	-7	-6	-5	-4	-3	-2	-1

Exemple :

a = [10,11,12,13,14,15,16] print(**a[0:2]**) **a[0:2]** = [ ] print(a) **a[0:1]** = [100,200] print(a)	le script affiche : **[10, 11]** **[12, 13, 14, 15, 16]** **[100, 200, 13, 14, 15, 16]**

V. Parcourir les listes.

```
for j in liste :
        blocs d'instructions
```

```
for i in range(len(liste)) :
        blocs d'instructions
```

Exemples :

L = [12,"Janvier",2013,"TANGER"] **for** i **in** L **:** print(i,end='-')	le script affiche : **12-Janvier-2013-TANGER-**

L = [12,"Janvier",2013,"TANGER"] **for** i **in range(len**(L)) **:** print(L[i],end='-')	le script affiche : **12-Janvier-2013-TANGER-**

VI. Modification des listes.

VI-1. Modifier la valeur d'une case :

L = [12,"Janvier",2013,"TANGER"] **L[-1]=200** print(L)	le script affiche : **[12,"Janvier",2013,200]**

VI-2. Modifier une tranche :

L = [12,"Janvier",2013,"TANGER"] **L[0 :3]=** ["CPGE",200] print(L)	le script affiche : **["CPGE",200, "TANGER"]**

VII. ATTENTION ! Copie d'une liste : L1=L[:]

L = ['Dans', 'python', 'tout', 'est', 'objet'] T = L T[4] = 'bon' print(T) print(L)	le script affiche : **['Dans', 'python', 'tout', 'est', 'bon']** **['Dans', 'python', 'tout', 'est', 'bon']**

L = ['Dans', 'python', 'tout', 'est', 'objet']
T = L[:] #copie d'une liste
T[4] = 'bon'
print(T)
print(L)
	le script affiche :
	['Dans', 'python', 'tout', 'est', 'bon']
	['Dans', 'python', 'tout', 'est', 'objet']

VIII. Les méthodes des listes :

- **Ajouter un élément**

Pour ajouter un/des élément(s) à une liste, on utilise les méthodes suivantes : **append**, **extend**, **insert**

- **append(x)** : Insère la valeur x à la fin de la liste spécifiée : **liste.append(x).**
- **extend(L)** : Rallonge la liste en ajoutant à la fin de la liste spécifiée la liste passée en paramètre : **liste.extend(L).**

- **insert(i,x)** : Insère l'élément x à l'emplacement i de la liste spécifiée : **liste.insert(i,x)**.

Exemples :

L = ["MPSI",1,2,3, "PCSI",1] **L.append(2)** print(L)	le script affiche : **['MPSI', 1, 2, 3, 'PCSI', 1, 2]**

L = ["MPSI",1,2,3] L2 = ["PCSI",1,2] **L. extend (L2)** print(L)	le script affiche : **['MPSI', 1, 2, 3, 'PCSI', 1, 2]**

#Attention !! L = ["MPSI",1,2,3] L2 = ["PCSI",1,2] **L. append (L2)** print(L)	le script affiche : **['MPSI', 1, 2, 3, ['PCSI', 1, 2]]**

L = ["MPSI",1,2,3,1,2] **L. insert(4 , "PCSI")** print(L)	le script affiche : **['MPSI', 1, 2, 3, 'PCSI', 1, 2]**

- **Supprimer un élément :**

Pour supprimer un élément de la liste, on utilise les méthodes suivantes : **remove,pop**

- **remove(x)** : Retire de la liste spécifiée le premier élément dont la valeur est égale à x : **liste.remove(x)**.

- **pop()** : Supprime le dernier élément de la liste spécifiée et le renvoie comme valeur de retour : **a = liste.pop()**.
- **pop(i)** : Supprime l'élément de rang i dans la liste spécifiée et le renvoie comme valeur de retour : **a = liste.pop(i)**.

Exemples :

L = ["MPSI",1,2,3, "PCSI",1,2] **L.remove("MPSI")** print(L)	le script affiche : **[1, 2, 3, 'PCSI', 1, 2]**

L = ["MPSI",1,2,3, "PCSI",1,2] **a=L.pop()** print(a) print(L)	le script affiche : **2** **["MPSI",1,2,3, "PCSI",1]**

L = ["MPSI",1,2,3, "PCSI",1,2] **a=L.pop(4)** print(a) print(L)	le script affiche : **PCSI** **['MPSI', 1, 2, 3, 1, 2]**

L = ["MPSI",1,2,3, "PCSI",1,2] **L.remove("PSI")**	**Traceback (most recent call last):** **File " tst.py", line 2, in <module>**

print(L)	L.remove("PSI") ValueError: list.remove(x): x not in list

Pq: On peut utiliser la fonction intégrée **del** pour supprimer un élément d'une liste:

Exemples :

L = ["MPSI",1,2,3, "PCSI",1,2] **del(L[5])** print(L)	le script affiche : **['MPSI', 1, 2, 3, 'PCSI', 2]**

L = ["MPSI",1,2,3, "PCSI",1,2] **del(L[0 :4])** print(L)	le script affiche : **['PCSI', 1, 2]**

- **Recherche d'éléments :**

Pour rechercher un élément dans une liste, on utilise la méthode : **index**

- **index(x)** : Renvoie l'indice dans la liste du premier élément dont la valeur est égale à x : **i = liste.index(x)**. Si aucun élément n'est trouvé, une erreur est générée.

Exemples :

L = ["MPSI",1,2,3, "PCSI",1] **i=L.index(1)** print(i)	le script affiche : **1**

L = ["MPSI",1,2,3, "PCSI",1] **i=L.index("PCSI")** print(i)	le script affiche : **4**

L = ["MPSI",1,2,3, "PCSI",1] **i=L.index("MP")** print(i)	**Traceback (most recent call last):** **File " tst.py", line 2, in <module>** **i=L.index("MP")** **ValueError: 'MP' is not in list**

- **Les autres méthodes des listes:**

• **count(x)**	Renvoie le nombre d'occurrences de la valeur x dans la liste considérée : **n = liste.count(x)**.
• **sort()**	Trie les éléments de la liste considérée : **liste.sort()**.
• **reverse()**	renverse l'ordre des éléments de la liste considérée : **liste.reverse()**.

- **Convertir une chaîne en liste : la méthode split()**

Exemple :

phrase="voici une liste de mots" L=phrase.**split()** print(phrase) print(L)	le script affiche : **voici une liste de mots** **['voici', 'une', 'liste', 'de', 'mots']**

- **Convertir une liste de chaînes de caractères en une chaîne de caractères : la méthode join()**

Exemple :

seq = ["A", "T", "G", "A", "T"] L="-".**join**(seq) print(seq) print(L)	le script affiche : **['A', 'T', 'G', 'A', 'T']** **A-T-G-A-T**

Les éléments de la liste initiale sont concaténés les uns à la suite des autres et intercalés par un séparateur qui peut être n'importe quelle chaîne de caractères (ici, nous avons utilisé un tiret).

Attention, la fonction **join()** ne s'applique qu'à une liste de chaînes de caractères.

Exemple :

maliste = ["A", 5, "G"] L="-".join(maliste)	le script affiche :**des erreurs**

C. LES TUPLES

Les **tuples**, ou **n-uplets**, sont très similaires aux listes *(ensemble de valeurs hétérogènes séparées par des virgules)* à la différence qu'ils **sont non modifiables** et ils utilisent les parenthèses au lieu des crochets :

Exemple :

a =(11, 'MPSI', 33) #définir un tuple	

Les opérations disponibles sur les listes sont à peu près toutes disponibles sur les tuples, sauf celles qui les modifient.

I. Tuple vs listes :

les tuples sont très compacts (i.e. ils occupent peu de mémoire) et l'accès à leurs éléments est très rapide.

II. Définir un tuple :

a = () #définir un tuple vide print (a)	le script affiche : **()**
a = tuple() #définir un tuple vide print (a)	le script affiche : **()**
a = (2, 10, 21) print (a)	le script affiche : **(2, 10, 21)**
a = tuple(range(10)) #définir un tuple d'entiers print (a)	le script affiche : **(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)**
a = 12345, 54321, 'salut !' print (a)	le script affiche : **(12345, 54321, 'salut !')**
a = (2,) #définir un tuple avec un seul élément print (a)	le script affiche : **(2,)**

L'affectation et l'indiçage fonctionne comme avec les listes, mais si l'on essaie de modifier un des éléments du tuple, Python renvoie un message d'erreur. Si vous voulez ajouter un élément (ou le modifier), vous devez créer un autre tuple :

III. Exemples :

```
x = (1,2,3)
print( x)
print( x[2])                #indiçage positif
print( x[-1])               #indiçage négatif
print(x[0:2])               #tranche
```

le script affiche :
(1, 2, 3)
3
3
(1, 2)

```
t1, t2 = (1, 2, 3), (4, 5)
print(t1)
print(t2)
```

le script affiche :
(1, 2, 3)
(4, 5)

```
x = (1,2,3)
x[2] = 15                   #modifier un tuple
```

N'est pas autorisée

```
x = ('MPSI',1,2,3)
y=x+('PCSI',1,2)        #concaténation de deux tuples
print(x)
print(y)
```

le script affiche :
('MPSI', 1, 2, 3)
('MPSI', 1, 2, 3, 'PCSI', 1, 2)

```
x = ('Python',3)
y=x*2                   #Répétition d'un tuple
print(x)
print(y)
```

le script affiche :
('Python', 3)
('Python', 3, 'Python', 3)

```
x = ('MP', 'SI', 'PC')   #Parcourir un tuple
for i in x:
        print(i)
```

le script affiche :
MP
SI
PC

```
x = ('MP', 'SI', 'PC')   #Parcourir un tuple
for i in range(len(x)):
        print(x[i])
```

le script affiche :
MP
SI
PC

```
x = ('MP', 'SI', 'PC')   #Parcourir un tuple
for i in enumerate(x):
        print(i)
```

le script affiche :
(0, 'MP')
(1, 'SI')
(2, 'PC')

```
x = ('MP', 'SI', 'PC')   #Parcourir un tuple
for i in enumerate(x):
        print(i[0],i[1])
```

le script affiche :
0 MP
1 SI
2 PC

IV. Emballage et Déballage :

```
t = 3, 2, 'Python'          # emballage en tuple (tuple packing)
                            # les valeurs 3, 2, 'Python' sont emballées ensemble dans un tuple t.
x, y, z = t                 # l'opération inverse : déballage de tuple (tuple unpacking)
```