



TYPES CONSTRUITS DE DONNÉES

Savoir construire, parcourir et manipuler les trois types de données construits: tuples, listes et dictionnaires.

Connaître la spécificité de chacun.

1. TUPLES ET LISTES:

En Python, une séquence est une collection ordonnée d'objets qui permet d'accéder à ses éléments par leur numéro de position (index) dans la séquence. Les **tuples** et les **listes** sont des séquences.

✓ **ACTIVITE:** Dans la console, observez les résultats obtenus puis complétez le tableau « aide mémoire ».

Création de séquences et accès aux éléments:

```
>>> liste1 = [13, 7, 21, 6, 7]
>>> tuple = (3.1, 7, 1.2)
>>> tuple      # affichage
>>> liste2 = ["U", "S", "A"]
>>> liste3 = liste1 + liste2
>>> liste3     # affichage
```

Les éléments d'une séquence sont séparés par une virgule. Pour un tuple, on les écrit entre parenthèse et entre crochets pour une liste.

```
>>> liste1[2]
>>> liste1[0:3] #slice de 0 à 3
>>> len(liste1) #nombre d'éléments
>>> 21 in liste1 #appartenance
>>> tuple[0]
>>> liste2.index("U")
>>> liste1.count(7)
```

On accède aux éléments d'une séquence comme avec une chaîne de caractères en indiquant son index entre crochet.

Parcours d'une séquence avec une boucle for:

```
>>> for i in liste2:
    print(i)
```

i prend successivement la valeur de chaque élément de la liste

```
>>> for i in range(len(liste2)):
    print(liste2[i], end=" ")
```

(sans saut de ligne)
Même résultat, on fait varier ici l'indice i de 0 à 3

Contrairement aux listes, les tuples ne sont **pas modifiables** (ou mutables). Testons quelques techniques de modification et de créations de listes:

```
>>> liste1[2]=3
>>> liste1
>>> liste2[0]="0"
>>> liste2
>>> liste2.append("S")
>>> liste2
>>> liste2.pop(2)
>>> liste2
```

Les méthodes `append()` et `pop()` sont les plus utiles.

On peut construire une liste "en compréhension", sans avoir à énumérer les éléments:

```
>>> liste=[i for i in range(1, 11)]
>>> liste
>>> liste=[i**2 for i in range(1, 11)]
>>> liste
```

```
>>> [i for i in "UN TEXTE"]
>>> liste
```

✓ AIDE MEMOIRE, TUPLES ET LISTES:

<pre>liste1 = [10, 20, 30] liste2 = ["A", "B", "C", "D"] tuple = (300, 30, 300)</pre>		NSI 05
Pour :	Je tape dans la console :	
ACCÈS AUX ÉLÉMENTS		
accéder à l'élément <code>30</code> de <code>liste1</code>	<code>>>> liste1[2]</code>	
accéder à l'élément <code>"A"</code> de <code>liste2</code>	<code>>>> liste2[0]</code>	
tester la présence de <code>30</code> dans <code>tuple</code>	<code>>>> 30 in tuple</code>	
connaître l'index de <code>"D"</code> dans <code>liste2</code>	<code>>>> liste2.index("D")</code>	
compter le nombre de <code>300</code> dans <code>tuple</code>	<code>>>> tuple.count(300)</code>	
PARCOURS D'UNE SEQUENCE		
parcourir les éléments de <code>liste2</code>	<pre>>>> for i in liste2: print(i, end=" ")</pre>	
ACTION SUR LES LISTES ET CREATION EN COMPREHENSION		
ajouter l'élément <code>40</code> à la fin de <code>liste1</code>	<code>>>> liste1.append(40)</code>	
remplacer l'élément <code>"A"</code> par <code>"B"</code> dans <code>liste2</code>	<code>>>> liste2[0]="B"</code>	
enlever <code>"D"</code> de <code>liste2</code>	<code>>>> liste2.pop(3)</code>	
créer en compréhension une liste de nombres pairs de 0 à 100	<pre>>>> [i for i in range(0,101,2)] ou: >>> [2*i for i in range(1, 51)]</pre>	

2. LES DICTIONNAIRES:

Les dictionnaires Python (appelés aussi tableaux associatifs) permettent d'associer des valeurs à des *clés* et non des index. À partir d'une clé, on peut alors accéder directement à la valeur qui lui est associée:

Syntaxe: `dico = {"clé1": valeur1, "clé2": valeur2, etc..}`

✓ ACTIVITE:

Création d'un dictionnaire:

```
>>> eleve = {"prenom": "Ana", "age": 14}
création d'un dictionnaire toujours entre accolades
clés "prénom" et "age" associées au valeurs "Ana" et 14

>>> len(eleve)
nombre de couples clé/valeur

>>> eleve["prenom"]
>>> eleve["age"]
affichage des valeurs
```

Modification d'un dictionnaire:

```
>>> eleve["taille"] = 1.64
ajout d'un couple clé/valeur

>>> eleve["age"] = 15
modification d'une valeur

>>> eleve # affichage
```

Parcours d'un dictionnaire:

```
>>> for i in eleve.keys():
    print(i) # affichage des clés
>>> for i in eleve.values():
    print(i) # affichage des valeurs
>>> for i,j in eleve.items():
    print(i,"->",j) # affichage des couples clé/valeur
```

👉 LES STRUCTURES IMBRIQUEES:

Il est possible de gagner en complexité en combinant listes, tuples et dictionnaires. Voir un exemple dans l'aide mémoire et en exercices.

✓ AIDE MEMOIRE, DICTIONNAIRES ET STRUCTURES IMBRIQUEES

DICTIONNAIRES	
carbone = {"symbole": "C", "Z": 6}	
Pour :	Je tape dans la console :
afficher la valeur associée à la clé "Z"	>>> carbone["Z"]
ajouter la clé "A" de valeur 12	>>> carbone["A"]=12
modifier cette valeur à 12,0107	>>> carbone["A"]=12.0107
afficher la taille du dictionnaire	>>> len(carbone)
afficher toutes les clés et valeurs	>>> for i,j in carbone.items(): □□ print (i, "->",j)
STRUCTURES IMBRIQUEES	
créer une liste de 3 tuples de deux éléments qui représente par exemple les coordonnées (x,y) de trois points	>>> liste=[(4,5),(-1,0), (2.5,1)]
afficher les coordonnées du deuxième point	>>> liste[1]
afficher l'abscisse du troisième point	>>> liste[2][0]
ajouter un quatrième point de coordonnées (0,1)	>>> liste.append((0,1))
Afficher les coordonnées de tous les points	>>> for i in liste: □□ print (i)