NSI | 05

TYPES CONSTRUITS DE DONNÉES

[Stéphane BEAUDET – Frédéric PEURIERE]

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 chaine de
caractère 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]="O"
>>> 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:

```
liste1 = [10, 20, 30]
liste2 = ["A","B","C","D"]
tuple =(300,30,300)
```

Pour :	Je tape dans la console :
ACCÈS AUX ÈLÈMENTS	
accéder à l´élément 30 de **liste1**	>>>
accéder à l´élément "A" de **liste2**	>>>
tester la présence de 30 dans **tuple**	>>>
connaître l´index de "D" dans **liste2**	>>>
compter le nombre de 300 dans **tuple**	>>>
PARCOURS D´UNE SEQUENCE	
parcourir les éléments de **liste2**	>>> ..... □□ print (i, end=" ")
ACTION SUR LES LISTES ET CREATION EN COMPREHENSION	
ajouter l´élément 40 à la fin de **liste1**	>>>
remplacer l´élément "A" par "B" dans **liste2**	>>>
enlever "D" de **liste2**	>>>
créer en compréhension une liste de nombres pairs de 0 à 100	>>>

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"	>>>
ajouter la clé "A" de valeur 12	>>>
modifier cette valeur à 12,0107	>>>
afficher la taille du dictionnaire	>>>
afficher toutes les clés et valeurs	>>> ..... □□ `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	>>>
afficher l´abscisse du troisième point	>>>
ajouter un quatrième point de coordonnées (0,1)	>>>
Afficher les coordonnées de tous les points	>>> ..... □□ `print (i)`