Partie II : Algorithmique et Programmation

SOMMAIRE

Chapitre 7 : Les dictionnaires, tuples et ensembles

Dictionnaires, Tupes & Ensembles

I. Dictionnaires

1. ***Définition***

Un dictionnaire est une autre structure de données qui permet de **stocker** plusieurs valeurs y compris des tuples, des listes ou même encore d'autres dictionnaires.

L'utilité des dictionnaires, c'est qu'ils nous permettent de stocker des valeurs en utilisant nos propres indices, ainsi il suffit de fournir l'indice pour accéder à l'information stockée avec cet indice.

2. ***Création d'un dictionnaire***

Un dictionnaire, appelé aussi un « tableau associatif », est une séquence de paires « clé : valeur » séparées par virgules et mises entre accolades.

```
dict1 = {'C451236' : 'Anabih issa', 'E985477' : 'Amal Abbassi' }
dict2 = {'a': 1, 'b':2, 'c':3, 'd':4}
dict3 = {'name': 'Ouassit Youssef'}
dict4 = { } # dictionnaire vide
```

Les deux points « **:** » doit être utiliser comme séparateur entre la clé la valeur correspondante.

3. ***Accès aux valeurs d'un dictionnaire***

Comme toutes les autres séquences pour accéder aux valeurs d'un dictionnaire, vous devez utiliser les crochets :

```
D = {"name": "zara", "age": 7, "class": "first"}
```

D ["name"]	D ["age"]	D ["class"]	D ["state"]
'zara'	7	'fisrt'	KeyError

Une Erreur « **KeyError** » est générée si vous avez fourni une clé qui n'existe pas, comme dans « state ».

4. ***Dictionnaires sont modifiables « mutables »***

Vous pouvez mettre à jour un dictionnaire par ajout de nouvelle valeur, modifier des valeurs existantes ou supprimer des valeurs.

AJOUT DE VALEUR
D = {"name": "zara", "age": 7, "class": "first"}
D["school"] = "High School"
La clé “school” n’existe pas dans le dictionnaire « D » donc elle va être ajoutée avec la valeur « High School »

MODIFICATION DE VALEUR
D = {"name": "zara", "age": 7, "class": "first"}
D["age"] = 8
La valeur de la clé « age » va etre modifiée en «8»

SUPPRESSION DE VALEUR
D = {"name": "zara", "age": 7, "class": "first"}
del D['age'] # supprimer la paire de cle ‘age’
D.clear() # vider le dictionnaire
del D # supprimer le dictionnaire tout entier

5. ***Propriétés de clés d’un dictionnaire***

Les valeurs d’un dictionnaire n’ont aucune restriction et donc elles peuvent être de n’importe quel type. Ce n’est pas le cas pour les clés.

Il y’a deux importants points à se rappeler pour les clés d’un dictionnaire :

- Une clé ne peut pas référencer plus qu’une valeur.
- La clé doit être « immutable » c.-à-d. ça peut être de n’importe quel type sauf (liste, dictionnaire).

6. ***Operations de bases sur les dictionnaires***

D = {'C451236' : 'Anabih issa', 'E985477' : 'Amal Abbassi' }		
Expression	**Résultat**	**Description**
len(D)	2	Nombre de valeurs dans un disctionnaire
'C451236' in D	True	Teste d’appartenance d’une clé
D.has_key('C451236')	True	Teste d’appartenance d’une clé
for cle in D : print(D[cle])	‘Anabih issa’ ‘Amal Abbassi’	Iteration par boucle for

7. ***Manipulation des dictionnaires (méthodes)***

Python inclut des méthodes de manipulation des dictionnaires :

D = {"name": "zara", "age": 7, "class": "first"}	
D.clear()	Vide le dictionnaire ‘D’
D.copy()	Retourne une copie du dictionnaire ‘D’

D.fromkeys(keys, value)	Crée et retourne un dictionnaire à partir de la liste keys et la valeur 'value'
D.get(key, default)	Return la valeur de la clé **key** si ça existe dans le dictionnaire si non la fonction retourne la valeur **default**
D.items()	Retourne la liste de paires (clé, valeur)
D.keys()	Retourne la liste des clés du dictionnaire 'D'
D.update(D2)	Met à jour le dictionnaire 'D' par le dictionnaire 'D2'
D.values()	Retourne la liste de valeurs stockées dans le dictionnaire 'D'

II. Tuples

1. *Définition : Tuples (immutable list)*

Un tuple est une séquence non modifiable de valeurs. C'est une séquence comme les listes, la seule différence, c'est qu'il est non modifiable (immutable).

2. *Création d'un tuple*

On peut créer un tuple facilement par insertion de valeurs en deux parenthèses séparées par virgules.

Tupl1=("physics", "chemistry", 2015)
Tupl2 = (1,2,3,4,6,50)
Tupl3 = (12,) # n'oubliez pas la virgule après la valeur.
Tupl4 = () # tuple vide à partir de la version 3 de Python

3. *Accès aux valeurs d'un tuple*

Comme toutes les autres séquences pour accéder aux valeurs d'un tuple, vous devez utiliser les crochets :

T = ("physics", "chemistry", 1999, 2015)			
T [0]	T [1]	T [2]	T [3]
'physics'	'chemistry'	1999	2015

4. *Accès aux sous-tuples d'un tuple (tranches)*

Comme toutes les autres séquences pour accéder aux sous-tuples d'un tuple, vous devez utiliser deux indices séparés par « : »

T = ("physics", "chemistry", 1999, 2015)			
T [0 :1]	T [2 :]	T [-1:]	T [-1:-1]
('physics',)	(1999,2015)	(2015,)	()

5. *Tuples sont non-modifiables « immutables »*

Comme les chaines de caractères « str », les tuples sont non-modifiables donc, vous êtes interdits de modifier les valeurs et sous-tuples d'un tuples, mais vous pouvez utiliser ces valeurs et sous-tuples pour créer des variations de votre tuple initiale.

(12, 'Hello', 13.1452)	+	(14 , -1j+15 , 'Python')
(12, 'Hello' , 13.1452 , 14 , -1j+15 , 'Python')		

6. *Suppression des éléments d'un tuple*

Les tuples sont non-modifiables, donc la suppression des valeurs et sous-tuples d'un tuple n'est encore pas possible. Ce qu'on peut faire c'est supprimer tout le tuple.

tup = ("physics", "chemistry", 1999, 2015)
del tup
« **del** » supprimera tout le tuple, après cette operation la variable « tup » n'est plus definie.

---

7. *Operations de bases sur les tuples*

Les tuples utilisent les operateurs « + » et « * » juste comme les chaines de caractères « str ».

Expression	Résultat	Description
len((1,2,3))	3	Nombre de valeurs dans un tuple
(1,2,3)+(4,5,6)	(1,2,3,4,5,6)	Concaténation
('Hi',)*4	('Hi', 'Hi', 'Hi', 'Hi')	Répétition
3 in (1,2,3)	True	Teste d'appartenance
for x in (1,2,3) : print(x)	1 2 3	Iteration par boucle for

8. *Délimiteurs ne sont pas nécessaires*

Toute séquence de valeurs séparées par virgule qui n'est pas délimiter par des crochets, des parenthèses ou bien des accolades est par défaut un tuple.

T = 'abc', 'hello', 12.35, -1, False, 18+0.5j

9. *Manipulation des tuples (fonctions)*

Python inclut des fonctions de manipulation des tuples :

len(tuple)	Retourne le Nombre des valeurs dans un tuple
max(tuple)	Retourne la plus grande valeur dans un tuple
min(tuple)	Retourne la plus petite valeur dans un tuple

III. Ensembles

1. *Définition*

Un ensemble en Python est une collection d'objets sans répétition et sans ordre (donc sans numérotage). Ce n'est PAS une séquence ! On les note comme en maths avec des accolades {...}.

Les éléments sont de types quelconques.

Exemples : {5,3,-8,2} {'o','e','y','u','a','i'} {5,'foo',(3,-2),7.4}

L'ensemble vide se note set() et non {} qui crée un dictionnaire !

Un ensemble est défini à l'ordre près :

```
>>> {3,1,2} == {2,3,1}
True
>>> {3,2,3,2} == {2,3}
True
```

Un ensemble paraît trié en interne mais c'est uniquement pour accélérer la recherche d'un élément :

```
>>> e = {3,'b',2,'a',5}
>>> e
{'b', 2, 3, 5, 'a'}
```

2. *Création d'un ensemble*

- En extension :

```
>>> E = {5,2,3,1}
>>> E {1, 2, 3, 5}
```

- En compréhension :

```
>>> E = {x*x for x in range(20) if x % 3 == 0}
>>> E {0, 225, 36, 9, 144, 81, 324}
```

- Avec la fonction set(...) de la classe set :

```
>>> E = set('aeiouy')
>>> E {'o', 'i', 'e', 'a', 'y', 'u'}
>>> E = set([5,2,5,6,2])
>>> E {2, 5, 6}
```

- En ajoutant un élément à un ensemble E avec la méthode E.add(x)

```
>>> E = {5,3,2,1}
>>> E.add(8)
>>> E {8,1,2,3,5} mutation !
```

3. *Accès aux valeurs d'un ensemble*

- Les éléments d'un ensemble ne sont pas numérotés.

On ne peut pas utiliser une notation comme e[i] puisque parler de l'élément numéro i n'a pas de sens!

- L'opérateur in permet de savoir si un objet appartient à un ensemble.
- L'opération E < F permet de tester si l'ensemble E est strictement inclus dans l'ensemble F.

```
>>> {2,4} < {1,2,3,4}
True
```

- Axiome du choix : il est possible d'obtenir un élément d'un ensemble E et de le supprimer en même temps avec la méthode pop().

```
>>> E = {5,2,3,1}
>>> x = E.pop()
>>> (x, E)
(1, {2,3,5})
```

donc un ensemble est mutable !

4. Opérations sur les ensembles

- L'**ensemble vide** se note set(). Définissons-le : `empty = set()`
- La **réunion** E ∪ F = {x : x ∈ E ou x ∈ F} se note E | F en Python.

```
>>> {3,2,5,4} | {1,7,2,5}
{1, 2, 3, 4, 5, 7}
```

- L'**intersection** E ∩ F = {x : x ∈ E et x ∈ F} se note E & F en Python.

```
>>> {3,2,5,4} & {1,7,2,5}
{2, 5}
```

- La **différence** E - F = {x : x ∈ E et x ∉ F} se note E - F en Python.

```
>>> {3,2,5,4} - {1,7,2,5}
{3, 4}
```

- Ne pas confondre `E - {x}` qui construit un nouvel ensemble, avec la méthode `E.remove(x)` qui supprime x de l'ensemble E.

5. Manipulation des ensembles (fonctions)

Python inclut des fonctions de manipulation des ensembles :

len(ensemble)	Retourne le Nombre des valeurs dans un ensemble
max(ensemble)	Retourne la plus grande valeur dans un ensemble
min(ensemble)	Retourne la plus petite valeur dans un ensemble