Informatique

13

Dictionnaires

Cours

Dictionnaires

1.I. Définition

Un dictionnaire est un objet Python qui permet de stocker des données. Contrairement aux listes ou aux tuples, où les éléments sont indexés par des entiers, les éléments (ou valeurs) d'un dictionnaire sont indexés par des **clés** qui peuvent être de n'importe quel type. Par exemple, on peut utiliser des chaînes de caractères pour les clés.

Un dictionnaire est alors un ensemble de couples clé : valeur

1.II. Sous Python

1.II.1 Création d'un dictionnaire

Le dictionnaire se crée sous Python à l'aide d'accolades :

```
dico = {}
```

On peut initialiser le dictionnaire en le créant :

```
tel = {'Valentin':'0658951423','Cathy':'0751210783','Theo':'0169842154'}
```

Il est possible d'ajouter autant d'éléments qu'on le veut, que le dictionnaire soit vide ou non, pour cela on écrit par exemple :

```
tel['Romain']='0836656565'
```

Voici ce que vaut notre dictionnaire après réalisation des commandes précédentes :

```
>>> tel
{'Valentin': '0658951423', 'Cathy': '0751210783', 'Theo': '0169842154', 'Romain': '0836656565'}
```

Les **clés** du dictionnaire que l'on vient de créer sont des chaines de caractères (prénoms) et les valeurs, des chaines de caractères (pour garder le premier 0) correspondant aux numéros de téléphones.

On peut stocker n'importe quel type de donnée dans un dictionnaire, voici un second exemple :

```
Eleve = {}
Eleve['nom'] = 'Pierre'
Eleve['classe'] = 'MPSI'
Eleve['notes info'] = [17,12,10]
Eleve['age'] = 19
```

```
>>> Eleve
{'nom': 'Pierre', 'classe': 'MPSI', '
notes info': [17, 12, 10], 'age': 19}
```

Remarque : il est possible d'utiliser des réels et des tuples comme clés :

```
>>> Dico = {1.2:1,"a":2,(1,1):3}

>>> Dico
{1.2: 1, 'a': 2, (1, 1): 3}
```

```
>>> Dico = {(1,1):1,(1,2):2,(2,1):3,(2,2):4}

>>> Dico[(1,2)]
2
```

1.II.2 Opérations sur les dictionnaires

On suppose dans ce paragraphe que le dictionnaire « dico » suivant existe :

```
>>> dico
{'nom': 'Pierre', 'classe': 'MPSI', 'notes info': [17, 12, 10], 'age': 19}
```

1.II.2.a Nombre d'éléments

Pour obtenir le nombre d'éléments dans un dictionnaire on utilise la commande « `len(…)` ».

```
>>> len(dico)
4
```

1.II.2.b Récupération de valeurs

Pour récupérer la valeur associée à une clé, on utilise les crochets comme pour une liste ou un tuple dans lesquels on précise la clé recherchée :

```
>>> dico['age']
19
```

1.II.2.c Parcours

On peut itérer sur les clés d'un dictionnaire dico à l'aide de l'instruction for cle in dico

Affichage des clés	Affichage des valeurs de chaque clé
`for cle in dico:` `  print(cle)`	`for cle in dico:` `  print(dico[cle])`
`for cle in dico.keys():` `  print(cle)`	`for val in dico.values():` `  print(val)`
nom classe notes info age	Pierre MPSI [17, 12, 10] 19
`for cle, val in dico.items():` `  print(cle,val)`	
nom Pierre classe MPSI notes info [17, 12, 10] age 19	

1.II.2.d Existence d'une clé

Pour vérifier si une clé (pas une valeur) existe dans un dictionnaire, on peut utiliser le test d'appartenance avec l'instruction « `in` » qui renvoie un booléen.

```
>>> 'nom' in dico
True

>>> 'Pierre' in dico
False

>>> 'prenom' in dico
False
```

1.II.2.e Suppression d'un couple

Pour supprimer un élément d'un dictionnaire on utilise la fonction « `del dico[cle]` » :

```
>>> dico
{'nom': 'Pierre', 'classe': 'MPSI', 'notes info': [17, 12, 10], 'age': 19}

>>> del dico['nom']

>>> dico
{'classe': 'MPSI', 'notes info': [17, 12, 10], 'age': 19}
```

1.II.3 Applications

Parmi les applications des dictionnaires, deux nous seront très utiles par la suite :

- Le comptage
- La détermination de chemins (Dijkstra et A star)

1.II.3.a Comptage

Cherchons le nombre d'apparition de chaque entier de 0 à 9 dans une liste d'entiers compris entre 0 et 9

```
L = [1,2,1,3,5,7,9,1,4,5,3,6,5,2,8,7,3,6,4,9,2,1,5,3,5,8,4,4,5,3,0,1,8,0]
dico = {}
for i in range(10): # On cherche les entiers de 0 à 9
    dico[i] = 0
for elem in L:
    dico[elem] += 1
print(dico)
```

```
{0: 2, 1: 5, 2: 3, 3: 5, 4: 4, 5: 6, 6: 2, 7: 2, 8: 3, 9: 2}
```

Remarque : Sans l'initialisation, on récupère le même dictionnaire, mais les clés sont dans l'ordre dans lequel apparaissent pour la première fois chaque entier :

```
L = [1,2,1,3,5,7,9,1,4,5,3,6,5,2,8,7,3,6,4,9,2,1,5,3,5,8,4,4,5,3,0,1,8,0]
dico = {}
for elem in L:
    if elem in dico:
        dico[elem] += 1
    else:
        dico[elem] = 1
print(dico)
```

```
{1: 5, 2: 3, 3: 5, 5: 6, 7: 2, 9: 2, 4: 4, 6: 2, 8: 3, 0: 2}
```

1.II.3.b Chemin

Soit un ensemble d'étapes depart:arrivee d'un voyage débutant à Paris et finissant à Strasbourg listées dans un ordre aléatoire :

```
dico = {'paris':'lyon','bordeaux':'saint
etienne','lyon':'marseille','brest':'rouen','bastia':'lille','nice':'bres
t','vienne':'valence','paris':'lyon','rouen':'bordeaux','montpelier':'bas
tia','valence':'montpelier','lyon':'marseille','saint
etienne':'vienne','marseille':'nice','lille':'strasbourg'}
```

On peut lister le trajet dans l'ordre depuis le départ ainsi :

```
Depart = 'paris'
Arrivee = 'strasbourg'
Intermediaire = Depart
while Intermediaire != Arrivee:
    Intermediaire = dico[Intermediaire]
    print(Intermediaire)
```

```
lyon
marseille
nice
brest
rouen
bordeaux
saint etienne
vienne
valence
montpelier
bastia
lille
strasbourg
```

Remarques :

- On pourrait changer le départ et l'arrivée pour obtenir une portion du trajet
- Dans Dijkstra, nous aurons une dictionnaire sous la forme arrivee:depart et il sera aussi simple de remonter le chemin depuis l'arrivée connue