

6.1 Introduction

Les dictionnaires sont composés d'un nombre fini d'éléments auxquels on peut accéder par une clé qui fait partie de l'élément. Chaque élément est donc une paire : une clé et une valeur (*key* et *value* en anglais).

Les dictionnaires ne sont pas ordonnés, on ne peut pas rechercher un élément à partir de sa position (indice) dans le dictionnaire mais seulement à partir de la clé.

Comme les listes, ce sont des objets *itérables* car on peut parcourir leurs éléments à l'aide d'une boucle `for`.

6.1 Introduction 1

6.2 Syntaxe 1

6.3 Manipulation des dictionnaires 1

6.2 Syntaxe

- ▶ Un dictionnaire python est une succession de paires d'objets séparées par une virgule, délimité par des accolades `{}` et `}`.
- ▶ Une paire est composée d'une clé et d'une valeur.
- ▶ Un dictionnaire est dit mutable c'est-à-dire que l'on peut en modifier (voire en supprimer) un ou plusieurs éléments.
- ▶ Les clés du dictionnaire doivent être non mutable (les types `list` et `dict` sont interdits).
- ▶ Les valeurs du dictionnaire peuvent être des objets de type quelconque (`integer`, `float`, `string`, `list`, `tuple`, `boolean`).

```
dico={cle1 : valeur1, cle2: valeur2 ,cle3 : valeur3,..., clen :  
      valeurn}
```

Exemple –

- ▶ `velo={'guidon': 1, 'roue': 2, 'derailleur': 21, 'frein': 2}`
- ▶ `pikachu={'pokemon':'souris', 'taille': 0.4, 'poids': 6,
 'type':'electrik', 'talent':['statik','paratonnerre']}`

6.3 Manipulation des dictionnaires

Le dictionnaire vide est désigné par `{}`.

Création

La création d'un dictionnaire se fait en choisissant un nom et par les signes d'accolades `mon_dico={}`.

Taille

La taille du dictionnaire est donnée par la fonction prédéfinie `length` : `len(mon_dico)`.

Ajout d'un élément ou modification d'une valeur

L'ajout d'un élément clé:valeur ou la modification d'une valeur si la clé correspondante existe ont la même syntaxe.

`mon_dico['classe']='PTSI'` *Remarque* : La clé n'est pas modifiable (non mutable) alors que la valeur l'est.

Suppression d'un élément

La suppression d'un élément du dictionnaire est réalisée par la fonction `del` en précisant la clé de l'élément à supprimer. La paire clé:valeur est alors supprimée. Python `del(mon_dico['classe'])`

Lecture d'une valeur

Python `pikachu['taille']` # renvoie la valeur associée à la clé 'taille' du dictionnaire `pikachu`. Cette instruction peut être associée à `print` pour afficher la valeur ou affectée à une variable.

Parcours du dictionnaire

On peut parcourir un dictionnaire par ses clés, ses valeurs ou ses éléments clé:valeur. Pour parcourir la totalité du dictionnaire, on utilise une boucle bornée `for`.

| Instruction | Effet |
|--|---------------------------------------|
| <code>for cle in velo.keys():</code> | parcourt les clés du dictionnaire |
| <code>for valeur in velo.values():</code> | parcourt les valeurs du dictionnaire |
| <code>for cle,valeur in velo.items():</code> | parcourt les éléments du dictionnaire |

Liste des clés ou liste des valeurs

On peut récupérer les différentes clés ou les différentes valeurs du dictionnaire sous forme de liste.

| Instruction | Effet |
|---|---|
| <code>LesCles=list(velo.keys())</code> | <code>['guidon', 'roue', 'derailleur', 'frein']</code> est affecté à <code>LesCles</code> |
| <code>LesValeurs=list(velo.values())</code> | <code>[1, 2, 21, 2]</code> est affecté à <code>LesValeurs</code> |

Vérification d'une clé

Pour vérifier qu'une clé existe ou non dans un dictionnaire, on utilise le terme d'appartenance `in`.

| Instruction | Effet |
|--|---|
| <code>'guidon' in velo.keys()</code> | renvoie le booléen <code>True</code> si la clé 'guidon' est dans <code>velo</code> , <code>False</code> sinon |
| <code>'guidon' not in velo.keys()</code> | renvoie le booléen <code>False</code> si la clé 'guidon' est dans <code>velo</code> , <code>True</code> sinon |