9 Dictionnaires

9.1 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`.

9.2 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, ... , 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']`

9.3 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'`.

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.
`del(mon_dico['classe'])`

Lecture d'une valeur

```
pikachu['taille'] # renvoie la valeur associée à la clé 'taille' du
    dictionnaire pikachu.
```

Parcours du dictionnaire

`for cle in velo` fonctionne aussi.

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
`for cle in velo.keys():`	Parcours des clés du dictionnaire
`for valeur in velo.values():`	Parcours des valeurs du dictionnaire
`for cle,valeur in velo.items():`	Parcours des 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
`LesCles=list(velo.keys())`	`['guidon','roue','derailleur','frein']` est affecté à `LesCles`
`LesValeurs=list(velo.values())`	`[1, 2, 21, 2]` est affecté à `LesValeurs`

Vérification d'une clé

1: `'guidon' in velo` fonctionne aussi.

Pour vérifier qu'une clé existe ou non dans un dictionnaire, on utilise le terme d'appartenance **`in`**. [1]

Instruction	Effet
`'guidon' in velo.keys()`	renvoie le booléen `True` si la clé 'guidon' est dans `velo`, `False` sinon
`'guidon' not in velo.keys()`	renvoie le booléen `False` si la clé 'guidon' est dans `velo`, `True` sinon