Dictionnaires

NSI1

20 janvier 2022

Un nouveau type

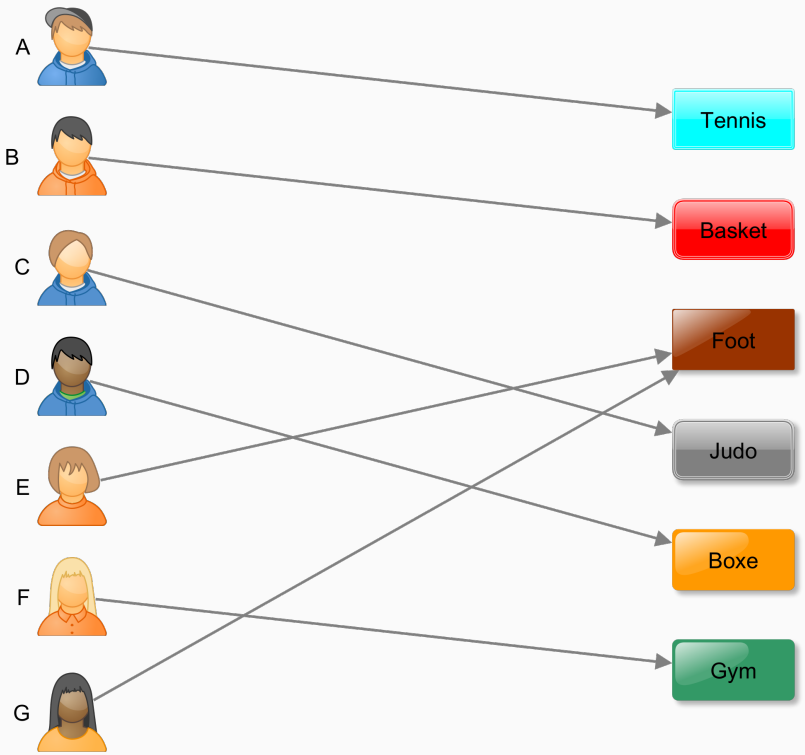
Situation

Situation

On demande à des jeunes quel est leur sport préféré.

Situation

On demande à des jeunes quel est leur sport préféré.Voici ce qu'on obtient :



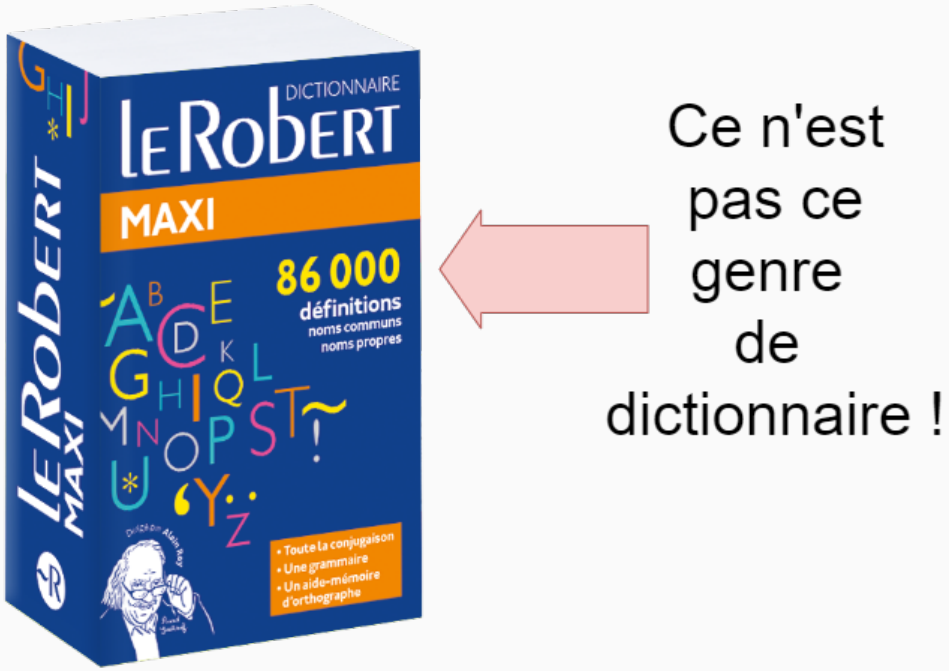
Un sport peut être cité par *plusieurs* jeunes.
En revanche, chaque jeune ne peut citer qu'*un seul* sport.

Dictionnaire

On pourrait utiliser une ou plusieurs listes pour représenter ces données mais il y a mieux : le *dictionnaire*.

Dictionnaire

On pourrait utiliser une ou plusieurs listes pour représenter ces données mais il y a mieux : le *dictionnaire*.



Un nouveau type

La variable `sport` est de type `dict` :

Un nouveau type

La variable `sport` est de type `dict` :

```
sport = {'A' : 'Tennis', 'B' : 'Basket',
         'C' : 'Judo', ' D' : 'Boxe',
         'E' : 'Foot', 'F' : 'Gym',
         'G' : 'Foot'}
```

Un nouveau type

La variable `sport` est de type `dict` :

```
sport = {'A' : 'Tennis', 'B' : 'Basket',
         'C' : 'Judo', ' D' : 'Boxe',
         'E' : 'Foot', 'F' : 'Gym',
         'G' : 'Foot'}
```

La syntaxe générale est :

```
variable = { cle1 : valeur1, cle2 : valeur2, ...}
```

Types autorisées

```
variable = { cle1 : valeur1, cle2 : valeur2, ...}
```

- Les valeurs peuvent être de n'importe quel type.

Types autorisées

```
variable = { cle1 : valeur1, cle2 : valeur2, ...}
```

- Les valeurs peuvent être de n'importe quel type.
- Les clés peuvent être

Types autorisées

```
variable = { cle1 : valeur1, cle2 : valeur2, ...}
```

- Les valeurs peuvent être de n'importe quel type.
- Les clés peuvent être
  - des `bool` , des `int` , des `float` ;

Types autorisées

```
variable = { cle1 : valeur1, cle2 : valeur2, ...}
```

- Les valeurs peuvent être de n'importe quel type.
- Les clés peuvent être
  - des `bool` , des `int` , des `float` ;
  - des `str` …

Types autorisées

```
variable = { cle1 : valeur1, cle2 : valeur2, ...}
```

- Les valeurs peuvent être de n'importe quel type.
- Les clés peuvent être
  - des `bool` , des `int` , des `float` ;
  - des `str` …
  - mais pas des `list` !

On peut tout de même utiliser des `tuples` en guise de clés :

Types autorisées

```
variable = { cle1 : valeur1, cle2 : valeur2, ...}
```

- Les valeurs peuvent être de n'importe quel type.
- Les clés peuvent être
  - des `bool` , des `int` , des `float` ;
  - des `str` …
  - mais pas des `list` !

On peut tout de même utiliser des `tuples` en guise de clés : les `tuples` ressemblent aux `list` mais sont non modifiables.

Types autorisées

```
variable = { cle1 : valeur1, cle2 : valeur2, ...}
```

- Les valeurs peuvent être de n'importe quel type.
- Les clés peuvent être
  - des `bool` , des `int` , des `float` ;
  - des `str` …
  - mais pas des `list` !

On peut tout de même utiliser des `tuples` en guise de clés : les `tuples` ressemblent aux `list` mais sont non modifiables.

`a = (1, 2, 3)` est un exemple de `tuple` .

Opérations sur les dictionnaires

Accéder à une valeur par sa clé

Pour connaître le sport préféré de `'A'`, c'est simple :

Accéder à une valeur par sa clé

Pour connaître le sport préféré de `'A'` , c'est simple :

```
>>> sport['A']
'Tennis'
```

Créer de nouveaux couples (`clé / valeur`)

Contrairement aux listes, il n'y a pas de méthode **`append`**.

Créer de nouveaux couples (`clé / valeur`)

Contrairement aux listes, il n'y a pas de méthode **append**.
Pour intégrer l'information « le sport préféré de H est le Rugby » on écrira simplement :

Créer de nouveaux couples (`clé / valeur`)

Contrairement aux listes, il n'y a pas de méthode **append**.
Pour intégrer l'information « le sport préféré de H est le Rugby » on écrira simplement :

```
sport['H']='Rugby'
```

Créer un dictionnaire vide et le peupler

On peut partir d'un dictionnaire vide

Créer un dictionnaire vide et le peupler

On peut partir d'un dictionnaire vide

```
d = dict()
```

Créer un dictionnaire vide et le peupler

On peut partir d'un dictionnaire vide

```
d = dict()
```

et remplir ses valeurs au fur et à mesure :

```
d['bonjour'] = 'hello'
```

Créer un dictionnaire vide et le peupler

On peut partir d'un dictionnaire vide

```
d = dict()
```

et remplir ses valeurs au fur et à mesure :

```
d['bonjour'] = 'hello'
```

```
d['crayon'] = 'pencil'
```

Créer un dictionnaire vide et le peupler

On peut partir d'un dictionnaire vide

```
d = dict()
```

et remplir ses valeurs au fur et à mesure :

```
d['bonjour'] = 'hello'
```

```
d['crayon'] = 'pencil'
```

```
d['se prélasser'] = 'to bask'
```

Créer un dictionnaire vide et le peupler

On peut partir d'un dictionnaire vide

```
d = dict()
```

et remplir ses valeurs au fur et à mesure :

```
d['bonjour'] = 'hello'
```

```
d['crayon'] = 'pencil'
```

```
d['se prélasser'] = 'to bask'
```

et cætera.

Parcourir l'ensemble des clés d'un dictionnaire

Parcourir l'ensemble des clés d'un dictionnaire

```
for cle in d.keys():
    print(cle)
```

Parcourir l'ensemble des clés d'un dictionnaire

```
for cle in d.keys():
    print(cle)
```

Ce script affiche

Parcourir l'ensemble des clés d'un dictionnaire

```
for cle in d.keys():
    print(cle)
```

Ce script affiche

```
bonjour
crayon
se prélasser
```

Parcourir l'ensemble des valeurs d'un dictionnaire

Parcourir l'ensemble des valeurs d'un dictionnaire

```
for valeur in d.values():
    print(valeur)
```

Parcourir l'ensemble des valeurs d'un dictionnaire

```
for valeur in d.values():
    print(valeur)
```

Ce script affiche

Parcourir l'ensemble des valeurs d'un dictionnaire

```
for valeur in d.values():
    print(valeur)
```

Ce script affiche

```
hello
pencil
to bask
```

Erreurs de clé

```
print(d['chien'])
```

Erreurs de clé

```
print(d['chien'])
```

Ce script produit une erreur :

Erreurs de clé

```
print(d['chien'])
```

Ce script produit une erreur :

```
KeyError : 'chien'
```

Un exemple

On veut créer un tableau de 10×10 cases avec la valeur 0 dedans.

Un exemple

On veut créer un tableau de 10×10 cases avec la valeur 0 dedans.

On peut bien sûr créer cela avec une liste de listes (en compréhension).

Un exemple

On veut créer un tableau de 10×10 cases avec la valeur 0 dedans.

On peut bien sûr créer cela avec une liste de listes (en compréhension).

On peut utiliser un dictionnaire :

Un exemple

On veut créer un tableau de 10×10 cases avec la valeur 0 dedans.

On peut bien sûr créer cela avec une liste de listes (en compréhension).

On peut utiliser un dictionnaire :

Un exemple

```
d = {(x, y) : 0 for x in range(1, 11) for y in range(1, 11)}
```

Un exemple

```
d = {(x, y) : 0 for x in range(1, 11) for y in range(1, 11)}
```

Avantages :

- plus simple à manipuler : on écrit `d[x, y]` au lieu de `d[x][y]` ;
- on n'est pas obligé de faire commencer les indices à zéro.

Inconvénients :

- prend plus de place en mémoire (on s'en fiche un peu);
- plus flexible entraîne plus de possibilité d'erreurs !

Utilisation des dictionnaires

Données structurées

Typiquement, pour stocker des données structurées :

Données structurées

Typiquement, pour stocker des données structurées :

```
reseau = {'nom'         : 'local',
          'ip'          : '192.168.1.0',
          'masque'      : '255.255.255.0',
          'passerelle' : '192.168.1.254'}
```

Tables

On utilise fréquemment des listes de dictionnaires...

Tables

On utilise fréquemment des listes de dictionnaires...

ou bien des dictionnaires de listes.