Dictionnaires : notions de 1ère année

Lycée Thiers

Dictionnaires : associer des clés à des valeurs

Un dictionnaire associe une valeur à chaque clé.

```
1 notes = {"Alice": 15, "Bob": 12, "Chloe": 17}
```

- Les clés sont ici des noms ; les valeurs sont des notes.

```
1 notes["Bob"]      # 12
```

Dictionnaires : associer des clés à des valeurs

Un dictionnaire associe une valeur à chaque clé.

```
1 notes = {"Alice": 15, "Bob": 12, "Chloe": 17}
```

- Les clés sont ici des noms ; les valeurs sont des notes.
- Une clé apparaît au plus une fois.

```
1 notes["Bob"]      # 12
```

Dictionnaires : associer des clés à des valeurs

Un dictionnaire associe une valeur à chaque clé.

```
1 notes = {"Alice": 15, "Bob": 12, "Chloe": 17}
```

- Les clés sont ici des noms ; les valeurs sont des notes.
- Une clé apparaît au plus une fois.
- Plusieurs clés peuvent avoir la même valeur.

```
1 notes["Bob"]      # 12
```

Dictionnaires : associer des clés à des valeurs

Un dictionnaire associe une valeur à chaque clé.

```
1 notes = {"Alice": 15, "Bob": 12, "Chloe": 17}
```

- Les clés sont ici des noms ; les valeurs sont des notes.
- Une clé apparaît au plus une fois.
- Plusieurs clés peuvent avoir la même valeur.
- On accède à une valeur par sa clé.

```
1 notes["Bob"]      # 12
```

À quoi servent les dictionnaires ?

- Associer des informations à des identifiants : notes, coordonnées, caractéristiques d'un objet.

Une liste permet d'accéder à un élément par sa position ; un dictionnaire permet d'accéder à une valeur par sa clé.

```
1 capitales = {"France": "Paris", "Italie": "Rome"}
2 capitales["Italie"]     # "Rome"
```

À quoi servent les dictionnaires ?

- Associer des informations à des identifiants : notes, coordonnées, caractéristiques d'un objet.
- Compter les occurrences de mots, de lettres ou de nombres.

Une liste permet d'accéder à un élément par sa position ; un dictionnaire permet d'accéder à une valeur par sa clé.

```
1 capitales = {"France": "Paris", "Italie": "Rome"}
2 capitales["Italie"]     # "Rome"
```

À quoi servent les dictionnaires ?

- Associer des informations à des identifiants : notes, coordonnées, caractéristiques d'un objet.
- Compter les occurrences de mots, de lettres ou de nombres.
- Mémoriser des résultats déjà calculés.

Une liste permet d'accéder à un élément par sa position ; un dictionnaire permet d'accéder à une valeur par sa clé.

```
1 capitales = {"France": "Paris", "Italie": "Rome"}
2 capitales["Italie"]     # "Rome"
```

À quoi servent les dictionnaires ?

- Associer des informations à des identifiants : notes, coordonnées, caractéristiques d'un objet.
- Compter les occurrences de mots, de lettres ou de nombres.
- Mémoriser des résultats déjà calculés.
- Regrouper des données selon une catégorie.

Une liste permet d'accéder à un élément par sa position ; un dictionnaire permet d'accéder à une valeur par sa clé.

```
1 capitales = {"France": "Paris", "Italie": "Rome"}
2 capitales["Italie"]     # "Rome"
```

Créer, consulter et modifier

```
1  d = {}                       # dictionnaire vide
2  d = {"Alice": 15, "Bob": 12}
3
4  d["Alice"]                   # consulter : 15
5
6  d["Chloe"] = 17              # ajouter une association
7  d["Bob"] = 14                # modifier une valeur
8
9  len(d)                       # nombre de cles : 3
10
11 del d["Alice"]               # supprimer une association
```

Attention : consulter ou supprimer une clé absente avec ces commandes provoque une erreur `KeyError`.

Tester la présence d'une clé

```
1 notes = {"Alice": 15, "Bob": 12}
2
3 "Bob" in notes          # True
4 "Chloe" in notes        # False
5 "Chloe" not in notes    # True
6 12 in notes             # False
```

Le test `in` porte sur les clés.

On peut tester la présence d'une clé avant de consulter sa valeur :

```
1 if "Chloe" in notes:
2     print(notes["Chloe"])
```

Consulter avec une valeur par défaut

`d.get(cle, defaut)` renvoie :

- la valeur associée à la clé si elle est présente ;

```
1 notes = {"Alice": 15, "Bob": 12}
2
3 notes.get("Bob", 0)     # 12
4 notes.get("Chloe", 0)   # 0
```

Exemple : compter les occurrences des éléments d'une liste.

```
1 effectifs = {}
2 for x in [2, 5, 2, 2, 5]:
3     effectifs[x] = effectifs.get(x, 0) + 1
4 # effectifs vaut {2: 3, 5: 2}
```

`get` ne modifie pas le dictionnaire.

Consulter avec une valeur par défaut

`d.get(cle, defaut)` renvoie :

- la valeur associée à la clé si elle est présente ;
- la valeur par défaut sinon.

```
1 notes = {"Alice": 15, "Bob": 12}
2
3 notes.get("Bob", 0)     # 12
4 notes.get("Chloe", 0)   # 0
```

Exemple : compter les occurrences des éléments d'une liste.

```
1 effectifs = {}
2 for x in [2, 5, 2, 2, 5]:
3     effectifs[x] = effectifs.get(x, 0) + 1
4 # effectifs vaut {2: 3, 5: 2}
```

`get` ne modifie pas le dictionnaire.

Parcourir un dictionnaire

```
1  notes = {"Alice": 15, "Bob": 12, "Chloe": 17}
2
3  # Parcourir les cles
4  for nom in notes:
5      print(nom)
6
7  # Parcourir les valeurs
8  for note in notes.values():
9      print(note)
10
11 # Parcourir les couples (cle, valeur)
12 for nom, note in notes.items():
13     print(nom, note)
```

Le parcours suit l'ordre d'insertion des clés, sans les trier.

Ne pas ajouter ni supprimer de clés pendant un parcours.

Coût des opérations courantes

On note n le nombre de clés du dictionnaire.

Opération	Coût usuel
Nombre de clés : `len(d)`	$O(1)$
Consultation : `d[cle]`, `d.get(cle)`	$O(1)$ en moyenne
Présence : `cle in d`	$O(1)$ en moyenne
Modification d'une valeur	$O(1)$ en moyenne
Ajout d'une clé	$O(1)$ amorti en moyenne
Suppression d'une clé	$O(1)$ en moyenne
Parcours complet	$O(n)$

- Amorti : le coût est réparti sur une suite d'ajouts.

Coût des opérations courantes

On note n le nombre de clés du dictionnaire.

Opération	Coût usuel
Nombre de clés : `len(d)`	$O(1)$
Consultation : `d[cle]`, `d.get(cle)`	$O(1)$ en moyenne
Présence : `cle in d`	$O(1)$ en moyenne
Modification d'une valeur	$O(1)$ en moyenne
Ajout d'une clé	$O(1)$ amorti en moyenne
Suppression d'une clé	$O(1)$ en moyenne
Parcours complet	$O(n)$

- Amorti : le coût est réparti sur une suite d'ajouts.
- Ces coûts supposent des clés dont le traitement coûte $O(1)$, par exemple de petits entiers.

Coût des opérations courantes

On note n le nombre de clés du dictionnaire.

Opération	Coût usuel
Nombre de clés : `len(d)`	$O(1)$
Consultation : `d[cle]`, `d.get(cle)`	$O(1)$ en moyenne
Présence : `cle in d`	$O(1)$ en moyenne
Modification d'une valeur	$O(1)$ en moyenne
Ajout d'une clé	$O(1)$ amorti en moyenne
Suppression d'une clé	$O(1)$ en moyenne
Parcours complet	$O(n)$

- Amorti : le coût est réparti sur une suite d'ajouts.
- Ces coûts supposent des clés dont le traitement coûte $O(1)$, par exemple de petits entiers.
- Les opérations annoncées en moyenne peuvent coûter $O(n)$ dans le pire cas.