Haute École Bruxelles-Brabant
École Supérieure d'Informatique
Bachelor en Informatique

2023 – 2024
DEV1 – DEV1L

DEV1 – DEV1L – Laboratoires Python

TD 13 – Dictionnaires

Les buts du TD sont :

- ▷ spécifier les syntaxes associées aux dictionnaires (syntaxe littérale, méthodes associées)
- ▷ mettre en oeuvre un dictionnaire (définir, modifier et parcourir)

Table des matières

1 Rappels **2**
1.1 Le concept 2
1.2 Syntaxe résumée 2
1.2.1 Opérations de base 2
1.2.2 D'autres opérations utiles 2
1.3 Usages idiomatiques 3

2 Découverte **4**

3 Exercices **5**

Version du document
13 novembre 2023

1 Rappels

1.1 Le concept

Un dictionnaire en Python est une *structure de données* dont le but est d'associer des *valeurs* à des *clefs*, de sorte à retrouver facilement la valeur à partir de la clef. Un dictionnaire contient donc des paires clef-valeur.

Pourquoi « dictionnaire »

On l'appelle *dictionnaire* en prenant l'exemple d'un « vrai dictionnaire » (de langue française, par exemple). Dans ce cas, les mots de la langue sont les clefs ; les définitions associées sont les valeurs. Le point crucial est : en connaissant un mot, on retrouve facilement la définition associée.

Quelques contraintes :

- À chaque clef du dictionnaire, il y a exactement une valeur associée.
- Les clefs doivent être des objets *hachables*. Retenons que ceci inclut les entiers, les flottants, les chaînes et les tuples, mais pas, par exemple, les listes.
- Les valeurs par contre, peuvent être de n'importe quel type, en ce compris des listes.

1.2 Syntaxe résumée

1.2.1 Opérations de base

Description	Exemple de syntaxe
création littérale	`{clef1: value1, clef2: value2}`
création en compréhension	`{expression1: expression2 for une_variable in un_itérable}`
dictionnaire vide	`{}` ou `dict()`
accès via `[]`	`mon_dictionnaire[clef]`
modifier ou ajouter une paire	`mon_dictionnaire[clef] = valeur`
tester l'existence d'une clef	`clef in mon_dictionnaire`
parcourir les clefs	`for key in mon_dictionnaire:`
parcourir les paires	`for key, value in mon_dictionnaire.items():`

1.2.2 D'autres opérations utiles

accès via `.get`	`mon_dictionnaire.get(clef, valeur_par_default)`
nombre de clefs	`len(mon_dictionnaire)`
liste des clefs	`list(mon_dictionnaire)`
liste des valeurs	`list(mon_dictionnaire.values())`
supprimer une clef (et sa valeur)	`del mon_dictionnaire[clef]`
récupère et supprime une valeur	`mon_dictionnaire.pop(clef, default_value)`
supprime tout le contenu	`mon_dictionnaire.clear()`
copie	`ma_copie = mon_dictionnaire.copy()`

1.3 Usages idiomatiques

Parmi les usages classiques des dictionnaires :

▷ créer une association du type mot-définition. Par exemple :

association.py
```
profs_dev1 = {
    "A111": "bej",  "A112": "mbr",  "A121": "sre",  "A122": "dbo",  "A131": "abe",
    "A132": "clg",  "A211": "bej",  "A212": "cuv",  "A221": "sre",  "A222": "sdr",
    "A231": "hal",  "A232": "nri",  "A311": "bis",  "A312": "clg",  "A321": "cuv",
    "A322": "hal",  "A331": "srv",  "A332": "gba",  "A341": "pbt",  "A342": "tni"
}
```

▷ compter des valeurs. Par exemple :

compter.py
```
groupes_par_prof = {
    "abe": 1, "bej": 2, "bis": 1, "clg": 2, "cuv": 2,
    "dbo": 1, "gba": 1, "hal": 2, "mbr": 1, "nri": 1,
    "pbt": 1, "sdr": 1, "sre": 2, "srv": 1, "tni": 1
}
```

▷ créer un "objet" avec des propriétés. Par exemple :

objets.py
```
une_personne = {
    "prénom": "Guillaume",
    "nom": "Apollinaire",
    "naissance": (26, 8, 1880)
}

autre_personne = {
    "prénom": "Jean-Jacques",
    "nom": "Goldmann",
    "naissance": (11, 10, 1951)
}
```

▷ combiner certaines idées ci-dessus pour créer une « base de données ». Par exemple :

bdd.py
```
etudiants = {
    52104: {
        "prénom": "Guillaume",
        "nom": "Apollinaire",
        "groupe": "A321"
    },
    45371: {
        "prénom": "Jean-Jacques",
        "nom": "Goldmann",
        "groupe": "A123"
    }
}
```

(Dans cet exemple, les clefs du dictionnaire *etudiants* sont les matricules. L'important est que chaque clef identifie exactement un élément de la base de données.)

2 Découverte

Dans cette section, nous suggérons d'utiliser l'interpréteur python pour *tester*.

Exercice 1 Tests libres

Essayez par vous même de définir un dictionnaire, afficher quelques valeurs.

Exercice 2 Tests dirigés

Si on suppose les objets de la section 1.3 correctement initialisés, quel code écrire pour...

- ▷ obtenir le prof du groupe A311

- ▷ obtenir le nombre de groupes de Mme Cuvelier (*cuv*).

- ▷ obtenir le *groupe* de l'étudiant ayant le matricule 52104

- ▷ afficher tou · tes les enseignant · es de dev1 (sans doublons)

Exercice 3 Tests de réflexion

Que se passe-t-il si on essaye de

- ▷ donner deux valeurs à une même clef ?

- ▷ donner la même valeur à deux clefs différentes ?

- ▷ accéder (par exemple afficher) à une clef qui n'existe pas dans le dictionnaire ?

- ▷ utiliser une liste comme clef ?

▷ utiliser, dans le même dictionnaire, des clefs de types différents (par exemple une chaîne et un booléen) ?

3 Exercices

Exercice 4 Exercice simple

Écrire un dictionnaire qui à chaque jour (une chaîne : lundi, mardi, etc.), associe sa traduction en anglais. (au besoin, aidez-vous d'un… dictionnaire.)

Exercice 5 Construire les fréquences

Étant donné une liste de chaînes, écrire une fonction qui établit le dictionnaire des fréquences de chaque chaîne présente.

En d'autres termes, créez un dictionnaire dont les clefs sont les chaînes de la liste, et dont les valeurs « comptent les occurrences » de chaque clef dans la liste.

Par exemple :

compter2.py

```
profs_dev1 = [
    "bej", "mbr", "sre", "dbo", "abe",
    "clg", "bej", "cuv", "sre", "sdr",
    "hal", "nri", "bis", "clg", "cuv",
    "hal", "srv", "gba", "pbt", "tni"
]

# résultat attendu:

{
    "abe": 1, "bej": 2, "bis": 1, "clg": 2, "cuv": 2,
    "dbo": 1, "gba": 1, "hal": 2, "mbr": 1, "nri": 1,
    "pbt": 1, "sdr": 1, "sre": 2, "srv": 1, "tni": 1
}
```

Exercice 6 Argmax

Étant donné un dictionnaire dont les valeurs sont des nombres, écrivez une fonction qui détermine la liste des clefs dont la valeur associée est la plus grande.

Par exemple :

argmax.py

```
data = {
    "abe": 1, "bej": 2, "bis": 1, "clg": 2, "cuv": 2,
    "dbo": 1, "gba": 1, "hal": 2, "mbr": 1, "nri": 1,
    "pbt": 1, "sdr": 1, "sre": 2, "srv": 1, "tni": 1
}

# résultat attendu:

 ["bej", "clg", "cuv", "hal", "sre"]
```

Exercice 7 Combinaison des précédents

Écrire un programme qui demande des mots à l'utilisateur · ice (un mot par ligne, et une ligne vide termine l'entrée), puis affiche le ou les mots [1] les plus fréquemment entrés.

1. En cas d'égalité il y aura plusieurs mots.