Exercices branchés/débranchés : Les dictionnaires en Python

mars 2025

Les questions rouge sont considérées comme plus difficiles ou demandent des codes plus longs.
Le PPE (parcours par élément : `for elt in dico`) sera largement privilégié : il s'agit d'*un parcours par clé*.

Soit le tableau associatif suivant :

Nom	Age
Aude	25
Patrice	26
Franck	25

1. Définir le dictionnaire `professeur` qui *implémente* le tableau associatif précédent.
2. Ecrire les lignes de code permettant :
   — d'afficher l'âge de Patrice
   — d'afficher le nombre de professeurs dans le tableau
   — d'ajouter l'entrée suivante `"Sophie" : 24`
   — de supprimer la clé `"Franck"` (et donc la valeur associée)
   — de modifier l'entrée `"Aude"` afin de lui associer `23`
3. Dans une console, quel type d'erreur l'instruction suivante renvoie-t-elle ?

   ```
   professeur ["Isabelle"]
   ```

4. On dispose d'un dictionnaire compilant des prénoms d'élèves (clé) et des moyennes (valeur). Par exemple :

   ```
   exemple = {"Lucy" : 10, "Luke" : 9.5, "Mark": 15.5, "Helene" : 12, "Olivia" : 8}
   ```

   Ecrire une fonction `parcours` acceptant comme unique paramètre un dictionnaire `dico` (type `dict`). La fonction affiche sur une seule ligne le prénom et la moyenne des élèves dont la moyenne est supérieure ou égale à 10 puis retourne `"Fini"` :

Console Python

```
>>> parcours({"Lucy" : 10, "Luke" : 9.5, "Mark": 15.5, "Helene" : 12, "Olivia" : 8})
eleve Lucy a 10
eleve Mark a 15.5
eleve Helene a 12
'Fini'
```

5. Ecrire une fonction `carre` qui accepte un seul paramètre : un entier positif `n`. La fonction retourne un dictionnaire contenant comme clés tous les entiers `i` inférieurs à `n` : les valeurs associées sont les nombres `i` élevés au carré.

Console Python

```
>>> carre(5)
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16}
```

6. Ecrire une fonction `unicode` qui accepte deux paramètres :
   - `debut` un nombre entier (`int`)
   - `fin` un nombre entier (`int`)

   La fonction retourne un dictionnaire contenant tous les couples (`cle : valeur`) où la clé est un caractère et la valeur son code unicode.
   Les paramètres `debut` et `fin` définissent les codes de début et de fin.

Console Python

```
>>> unicode(60, 80)
{'<': 60, '=': 61, '>': 62, '?': 63, '@': 64, 'A': 65, 'B': 66, 'C': 67, 'D': 68, 'E': 69,
↪  'F': 70, 'G': 71, 'H': 72, 'I': 73, 'J': 74, 'K': 75, 'L': 76, 'M': 77, 'N': 78, 'O': 79,
↪  'P': 80}
```

7. Pour créer un compte sur un site WEB, un utilisateur doit donner un prénom, un login et un mot de passe. Ensuite, pour accéder au site, l'utilisateur ne doit fournir que le couple *(login,mdp)* afin de s'identifier ; Par ailleurs, le site stocke tous les identifiants et la date de dernière connexion dans un dictionnaire selon le format suivant :

```
{"login" :
        {"prenom" : <prenom>,
          "mdp" : <mot_de_passe>,
          "date": <date_de_derniere_connexion>
         },...
}
```

Voici le contenu du dictionnaire d aujourd'hui :

```
>>> d = {"LouLouCMoi" :
...         {"prenom" :"Lou",
...           "mdp" : "3juillet2005@!",
...           "date": "16/11/2022"
...          },
...      "oznE":
...          {"prenom" :"Enzo",
...           "mdp" : "Mbappe78",
...           "date": "5/2/2023"
...          },
...
...     "LaReine":
...          {"prenom" :"Olivia",
...           "mdp" : "N3wt0nJ0hN",
...           "date": "5/2/2023"
...          }
... }
```

(a) Quelle propriété des clés de dictionnaires est utile dans cet exemple ?

(b) Ecrire une ligne de code permettant d'afficher le nombre d'utilisateurs actuellement inscrits.

(c) Ecrire une fonction auth qui accepte trois arguments `dico` (type `dict`), `login` (type `str`) et `mdp` (type `str`) :

- si le couple *(login, mdp)* existe, alors on retourne la chaîne formatée :
  `"Bienvenu(e) {nom} ! Comment allez-vous depuis le {date} ?"`
- si le *login* existe mais le *mot de passe* ne correspond pas, alors on retourne la chaîne :
  `"Mot de passe INCORRECT"`
- si le login n'existe pas, alors on retourne :
  `"Identifiant {login} est inconnu"`

Console Python

```
>>> auth(d, "oznE", "Mbappe78")
'Bienvenu(e) Enzo ! Comment allez-vous depuis le 5/2/2023 ?'
>>> auth(d, "oznE", "Mbappe13")
'Mot de passe INCORRECT'
>>> auth(d, "thebrat", "Mbappe78")
'Identifiant thebrat est inconnu'
```

(d) Que pourrait-il se passer si un nouvel utilisateur créait un compte avec le login `"oznE"` ?

8. La fonction native `sorted` accepte *un iterable* comme unique paramètre et retourne une liste triée des valeurs de l'*iterable* (ordre croissant). La console suivante illustre le fonctionnement de cette fonction :

Console Python

```
>>> sorted("badcjf")
['a', 'b', 'c', 'd', 'f', 'j']
>>> sorted([3, 2, 1, 0])
[0, 1, 2, 3]
>>> sorted((3, 2, 1, 0))
[0, 1, 2, 3]
```

On rappelle que la méthode `.keys()` retourne un *iterable* contenant les valeurs des clés d'un dictionnaire.

Ecrire une fonction `dict_vers_list_triee` qui accepte comme unique paramètre, un dictionnaire `dico` dont toutes les clés sont des nombres entiers (type `int`).
La fonction retourne une liste de *2-uplets* (tuple de taille 2) contenant tous les couples *(cle, valeur)* du dictionnaire. Cette liste doit être triées par clé croissante.

Console Python

```
>>> dict_vers_list_triee ({2: "Luce", 1: "Angèle", 4: "Marion", 3: " Debbie"})
[(1, 'Angèle'), (2, 'Luce'), (3, ' Debbie'), (4, 'Marion')]
```

9. Ecrire une fonction `compte_valeurs` qui accepte comme unique paramètre `dico` de type `dict`. Il s'agit d'un dictionnaire dont toutes les valeurs sont des entiers (type `int`).
   La fonction retourne un dictionnaire dont les clé sont les valeurs de `dico` et les valeurs sont les nombres d'apparition dans le dictionnaire `dico`. Par exemple :

Console Python

```
>>> compte_valeurs({"a": 1, "b": 4, "c": 1, "d": 4, "e": 1, "f": 2})
{1: 3, 4: 2, 2: 1}
```

10. Ecrire une fonction `construire` qui accepte deux paramètres de type `list` :
    - `key` une liste de clés de type `str`
    - `value` une liste de clés de type `str`

    Les deux listes entrées comme paramètres ont la même longueur. La fonction retourne un dictionnaire `{cle : valeur}` construit par compréhension de sorte que le clés et les valeurs possèdent le même indice dans chacune des listes :

Console Python

```
>>> construire(["maillot", "short", "chaussette"], ["rouge", "bleu", "blanc"])
{'maillot': 'rouge', 'short': 'bleu', 'chaussette': 'blanc'}
```