TP N°8: Les dictionnaires

Un *dictionnaire* en python est un type de données *non ordonné*, *mutable* et appartenant au type *dict*. Le dictionnaire est une sorte de liste mais au lieu d'utiliser des index, on utilise des *clés*, c'est à dire des valeurs autres que numériques.

Il permet de stocker des couples « **clé** : **valeur** ».

- Les **clés** pouvaient être des entiers, des chaînes et «quelques autres types non-mutables» , les tuples sont un de ces types. Toutes les clés doivent être distinctes.
- Les **valeurs** peuvent être de n'importe quel type Python. Toutes les valeurs possibles peuvent être stockés dans un dictionnaire.

Création

```
d1 = dict(); d2 = {} # dictionnaires vides
d3 = {"un":"one", "deux":"two", "trois":"three"}
d4 = {"p1": (1, 2.5) , "p2": (5.,7.8)} # créer un dictionnaire non vide
L=[(1, 2.5) , (5.,7.8)]
d5 = dict(L) # créer un dictionnaire à partir d'une liste de couples : d5 = {1: 2.5, 5.0: 7.8}
```

Remarques :

- Le dictionnaire vide se note `dict()` ou `{ }`.
- Les clés de dictionnaire doivent être non-mutables. Les tuples sont non-mutables mais si vous avez un tuple contenant des listes, il est considéré comme mutable et n'est pas utilisable comme clé de dictionnaire. Seuls les tuples de chaînes, de nombres ou d'autres tuples utilisable comme clé peuvent être utilisés comme clé de dictionnaire.

Fonctions et méthodes associées aux dictionnaires

```
d1 = {"un":"one", "deux":"two", "trois":"three"} # dictionnaire initial
d1["un"] ; d1.get("un")         # Récupérer la valeur de la clé "un"
d2 = {"etudiants" : [ { "nom" : "Ali" , "age" : 20} , {"nom" : "Ahmed" , "age" : 21 } ],
      "matieres" : { "Informatique" : {"coeff" : 4,  "nb_heures" : 2 } ,
                     "Français"     : {"coeff" : 4,  "nb_heures" : 2 } ,
                     "Analyse"      : {"coeff" : 10, "nb_heures" : 6 }
                   }
     }
d2["etudiants"][0]["nom"] # "Ali"
d2["matieres"]["Informatique"]["coeff"] # 4
d1["quatre"] = "four"      # ajouter/modifier(si la clé existe) le couple "quatre":"four"
d1.__setitem__("cinq","five")
d3 = {"six":"six", "sept":'seven'}
d1.update(d3)        # ajouter un dictionnaire d2 à un dictionnaire existant d.
d1.pop("trois")       # retourne la valeur "three" associée à la clé "trois"
                     # et supprime le couple "trois":"three" du dictionnaire.
del d1["quatre"]      # Supprimer le couple de clé "quatre"
len(d1)               # renvoie le nombre d'éléments d'un dictionnaire
"deux" in d1          # Tester avec in et not in si une clé appartient ou non à un dictionnaire.
d1 == d3              # Comparer le contenu de deux dictionnaires avec ==  et !=
d1.clear()            # Vider le dictionnaire d
```

Remarques :

- Si l'on demande la valeur associée à une clé inexistante, une exception `KeyError` est levée.
- Il est aussi possible de demander si la clé existe, avec l'opérateur `in` ou via la méthode `has_key()`. Exemple : `'onze' in d`, `d.has_key('onze')`

Exercice 1

1. Créer le dictionnaire d2 proposé précédemment.
2. Y ajouter un nouveau étudiant dont le nom est "Sami" et l'age est 19 ans.
3. Quelle est la taille du dictionnaire des matières.
4. Afficher les matières dont le coefficient égal à 10.

Itération sur les dictionnaires

Un dictionnaire étant un objet itérable, on peut itérer dessus. Par défaut, on itère sur les clés, mais on peut parcourir un dictionnaire avec les méthodes `.keys()`, `.values()` ou `.items()` qui renvoient une liste constituée des *clefs*, des *valeurs*, ou de *tuples (clef,valeur)*.

```
for k in d: # Par défaut, on itère sur les clés,  k représente une clé
    print("clé :", k, " --- valeur :", d[k])

for k in  d.keys(): # k représente une clé
    print(d[k])

for v in d.values(): # v représente une valeur
    print(v)

for k,v in d.items(): # items() : retourne des tuples (k = clé, v = valeur)
    print(k,v)
```

Exercice 2

Écrivez un programme python qui échange les clés et les valeurs d'un dictionnaire (ce qui permettra par exemple de transformer un dictionnaire anglais/français en un dictionnaire français/anglais). On suppose que le dictionnaire ne contient pas plusieurs valeurs identiques.
Exemple : `{"un":"one", "deux":"two"}` → `{"one":"un", "two":"deux"}`

Exercice 3

Écrivez un script qui crée un mini-système de base de données fonctionnant à l'aide d'un dictionnaire, dans lequel vous mémoriserez les noms d'une série de copains, leurs âges et leurs tailles.
Votre script devra comporter deux parties : la première pour le remplissage du dictionnaire, et la seconde pour sa consultation.

- Dans la partie de remplissage, utilisez une boucle pour accepter les données entrées par l'utilisateur. Dans le dictionnaire, le nom de l'élève servira de clé d'accès, et les valeurs seront constituées de tuples (âge, taille), dans lesquels l'âge sera exprimé en années (donnée de type entier), et la taille en mètres (donnée de type réel).
- La partie de consultation comportera elle aussi une boucle, dans laquelle l'utilisateur pourra fournir un nom quelconque pour obtenir en retour le couple (âge, taille) correspondant.
  Le résultat devra être une ligne de texte bien formatée, telle par exemple :
  `« Nom :  Ali ben Ahmed - âge :  15 ans - taille :  1.74 m ».`
  Pour obtenir ce résultat, utilisez la méthode `format()` pour le formatage des chaînes de caractères.

Exercice 4

1. Vous avez à votre disposition un texte quelconque. Écrivez un script qui analyse ce texte et enregistre dans un dictionnaire les occurrences de chacune des lettres de l'alphabet dans ce texte .
2. Modifiez le script ci-dessus afin qu'il mémorise dans un dictionnaire l'emplacement exact de chacun des mots (compté en nombre de caractères à partir du début).
   Lorsqu'un même mot apparaît plusieurs fois, tous ses emplacements doivent être mémorisés : chaque valeur de votre dictionnaire doit donc être une liste d'emplacements.