23 Les Dictionnaires

23.1 Introduction

Prenons l'exemple d'un répertoire téléhonique. Nous pouvons le mémoriser simplement comme un tableau (ou liste) de tableaux `[nom,numéro]`

[ ]:
```
liste_tel = [["Paul", 5234],
             ["Emile", 5345],
             ["Victor", 5186],
             ["Rose", 5678],
             ["Hélène", 5432]]
```

Si nous voulons appeler Rose, nous avons deux possibilités avec un tel tableau : * soit il faut savoir que les informations la concernant sont dans le quatrième élément de la liste (ce qui ne semble pas très pratique et réaliste)

[ ]:
```
print(liste_tel[3][1]) # il faut savoir que l'index de Rose est 3
```

- soit nous cherchons dans le tableau en partant du premier élément de la liste jusqu'à ce que nous trouvions Rose (ce qui revient à feuilleter son répertoire) : cela nécessite d'utiliser une boucle pour parcourir le tableau.

[ ]:
```
for element in liste_tel:
    if element[0] == 'Rose':
        print(element[1])
```

```
5678
```

Vous conviendrez que ce n'est pas pratique pour accéder à son numéro de téléphone. De même, la modification ou l'ajout d'un information nécessiterait de devoir feuilleter tout le répertoire. Il semblerait plus pratique d'associer un nom à un numéro, autrement dit d'associer à une **information** à une **clé**.

C'est ce que les dictionnaires permettent !

23.2 Les dictionnaires en Python

Un dictionnaire est un ensemble non ordonné de paires (clé, valeur) avec

On peut ajouter des couples (clé, valeur) à un dictionnaire, si la clé figure déjà dans le dictionnaire alors le couple est remplacé par le nouveau. Une **clé** peut être de type alphabétique, numérique, ou même de type construit sous certaines conditions. Les **valeurs** pourront être de tout type sans exclusion. En Python, le dictionnaire est un objet **mutable**, autrement dit, on peut le modifier.

23.2.1 Création d'un dictionnaire

Plusieurs méthodes permettent de créer soit un dictionnaire vide, soit de le noter en extension, soit par compréhension.

[ ]:
```
d1 = {}      # Création d'un dictionnaire vide
d2 = dict() # Création d'un dictionnaire vide (autre méthode)
d3 = {'poires': 5, 'bananes': 7, 'abricots' : 12} # création d'un dictionnaire␣
 ↪par extension
d4 = {k: k**2 for k in range(1, 10)} # création d'un dictionnaire par␣
 ↪compréhension

print(type(d1))
```

```
<class 'dict'>
```

[ ]:
```
print("d1 =>", d1)
print("d2 =>", d2)
print("d3 =>", d3)
print("d4 =>", d4)
```

```
d1 => {}
d2 => {}
d3 => {'poires': 5, 'bananes': 7, 'abricots': 12}
d4 => {1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81}
```

Il est même possible de créer un dictionnaire à partir d'une liste de couples.

[ ]:
```
liste = [('cle1','valeur1'),('cle2','valeur2')]
d5 = dict(liste)
liste_tel = [["Paul", 5234], ["Emile", 5345], ["Victor", 5186], ["Rose", 5678],␣
 ↪["Hélène", 5432]]
d6 = dict(liste_tel)

print("d5 =>", d5)
print("d6 =>", d6)
```

```
d5 => {'cle1': 'valeur1', 'cle2': 'valeur2'}
d6 => {'Paul': 5234, 'Emile': 5345, 'Victor': 5186, 'Rose': 5678, 'Hélène':
5432}
```

Important : Vous aurez noté que les dictionnaires Python se représentent entre accolades {}. Les différentes paires sont séparées par des virgules et sont de la forme `clé: valeur`.

23.2.1.1 Activité 23.1

Créez un dictionnaire appelé `notes` qui contient les paires (matières, moyenne) de vos trois spécialités. Affichez ensuite ce dictionnaire.

[ ]:
```
# à vous de jouer !
```

23.2.2 Accès, modification, ajout, suppression

L'**accès** à une valeur d'un dictionnaire se fait par sa clé.

[ ]:
```
d3 = {'poires': 5, 'bananes': 7, 'abricots' : 12}
d3['abricots']
```

[ ]:
```
12
```

Le dictionnaire étant un objet mutable on peut **modifier** la valeur associée à une clé ou **ajouter** une nouvelle association et afficher le dictionnaire modifié.

[ ]:
```
d = {'Paul': 5234, 'Emile': 5345, 'Victor': 5186, 'Rose': 5678, 'Hélène': 5432}
d['Rose'] = 4921     # clé existante donc modification de la valeur
d['Louane'] = 4118   # nouvelle clé donc ajout d'une nouvelle association
print(d)
```

```
{'Paul': 5234, 'Emile': 5345, 'Victor': 5186, 'Rose': 4921, 'Hélène': 5432,
'Louane': 4118}
```

Pour **supprimer** une association d'un dictionnaire on peut utilise le mot clé `del`.

[ ]:
```
print(d)
del d['Paul']
print(d)
```

```
{'Paul': 5234, 'Emile': 5345, 'Victor': 5186, 'Rose': 4921, 'Hélène': 5432,
'Louane': 4118}
{'Emile': 5345, 'Victor': 5186, 'Rose': 4921, 'Hélène': 5432, 'Louane': 4118}
```

23.2.3 Taille d'un dictionnaire

La fonction `len` renvoie la taille d'un dictionnaire.

[ ]:
```
d3 = {'poires': 5, 'bananes': 7, 'abricots' : 12}
len(d3)
```

[ ]:
```
3
```

23.2.3.1 Activité 23.2

On reprend le dictionnaire `notes` de l'activité 1.

1. Affichez la moyenne de NSI.

[ ]:
```

```

2. Modifiez votre moyenne de NSI qui a gagné 2 points. Affichez le dictionnaire.

[ ]:

3. Ajoutez la matière `Anglais` avec sa moyenne. Affichez le dictionnaire.

[ ]:

4. Affichez la taille du dictionnaire.

[ ]:

5. Supprimez une des trois spécialités et affichez le dictionnaire.

[ ]:

23.3 Les itérateurs pour les dictionnaires

Il est possible de parcourir un dictionnaire de trois manières :

- parcourir l'ensemble des **clés** avec la méthode `keys()` ;
- parcourir l'ensemble des **valeurs** avec la méthode `values()` ;
- parcourir l'ensemble des **paires clés-valeurs** avec la méthode `items()`.

On peut itérer sur un dictionnaire grâce à l'une de ces méthodes.

[ ]:
```
d = {'Paul': 5234, 'Emile': 5345, 'Victor': 5186, 'Rose': 5678, 'Hélène': 5432}
for prenom in d.keys():
    print(prenom)
```

```
Paul
Emile
Victor
Rose
Hélène
```

[ ]:
```
for num in d.values():
    print(num)
```

```
5234
5345
5186
5678
5432
```

[ ]:
```
for prenom, num in d.items():
    print(prenom, '->', num)
```

```
Paul -> 5234
Emile -> 5345
Victor -> 5186
Rose -> 5678
Hélène -> 5432
```

On peut aussi interroger l'appartenance d'une valeur ou d'une clé grâce au mot clé `in`.

[ ]:
```
'John' in d.keys()
```

[ ]:
```
False
```

[ ]:
```
'Paul' not in d.keys()
```

[ ]:
```
False
```

[ ]:
```
5186 in d.values()
```

[ ]:
```
True
```

23.3.0.1 Activité 23.3

On considère le dictionnaire `fruits` suivant.

[ ]:
```
fruits = {'poires': 5, 'pommes': 11, 'bananes': 7, 'abricots' : 12}
```

1. Affichez tous les fruits du dictionnaire.

[ ]:
```

```

2. Affichez toutes les quantités du dictionnaire.

[ ]:
```

```

3. Ecrivez un programme permettant d'obtenir l'affichage suivant.

```
Il reste 5 poires
Il reste 11 pommes
Il reste 7 bananes
Il reste 12 abricots
```

[ ]:
```

```

Les dictionnaires : EXERCICES

23.4 Exercice 23.1 :

On considère le dictionnaire suivant qui contient différents fruits ainsi que leurs quantités.

[ ]:
```
fruits = {"pommes": 8, "melons": 3, "poires": 6}
```

1. Quelle instruction permet d'accéder au nombre de melons ?

[ ]:
```

```

2. On a acheté 16 clémentines et utilisé 4 pommes pour faire une tarte. Quelles instructions permettent de mettre à jour le dictionnaire ?

[ ]:
```

```

23.5 Exercice 23.2 :

Répondez aux questions suivantes **sans exécuter les scripts proposés**. Vous les exécuterez pour vérifier vos réponses. 1. Qu'affiche le programme suivant ?

[ ]:
```
fruits = {'pommes': 4, 'melons': 3, 'poires': 6, 'clémentines': 16}
for c in fruits.keys():
    print(c)
```

2. Qu'affiche le programme suivant ?

[ ]:
```
fruits = {'pommes': 4, 'melons': 3, 'poires': 6, 'clémentines': 16}
for cle, valeur in fruits.items():
    print(cle, "->", valeur)
```

Réponse :

3. Qu'affiche le programme suivant ?

[ ]:
```
fruits = {'pommes': 4, 'melons': 3, 'poires': 6, 'clémentines': 16}
for v in fruits.values():
    print(v)
```

Réponse :

23.6 Exercice 23.3

On considère qu'il faut ajouter un fruit sur la liste des courses s'il en reste 4 ou moins. 1. Ecrivez un programme qui affiche la liste des courses en considérant le dictionnaire suivant.

[ ]:
```
fruits = {'pommes': 4, 'melons': 3, 'poires': 6, 'clémentines': 16}
# à compléter :
```

2. Ecrivez une fonction `liste_courses(fruits)` qui prend en paramètre un dictionnaire `fruits` et qui renvoie un tableau avec les fruits de la liste de courses.

[ ]:
```

```

23.7 Exercice 23.4 :

Voici deux dictionnaires :

[ ]:
```
athletes = {"Mike": (1.75, 68), "John": (1.89, 93), "Kate": (1.67, 62)}
sportifs = {"Mike": {"taille": 1.75,"poids": 68}, "John": {"taille": 1.
 ↪89,"poids": 93}, "Kate": {"taille": 1.67,"poids": 62}}
```

1. De quel type sont les clés des deux dictionnaires `athletes` et `sportifs`? De quels types sont les valeurs de ces deux dictionnaires ?

Réponse :

2. Quelle instruction permet d'accéder à la taille de Kate dans le dictionnaire `athletes` ?

[ ]:

3. Quelle instruction permet d'accéder à la taille de Kate dans le dictionnaire `sportifs` ?

[ ]:

23.8 Exercice 23.5 :

Le Scrabble est un jeu de société où l'on doit former des mots avec tirage aléatoire de lettres, chaque lettre valant un certain nombre de points. Le dictionnaire `scrabble` contient cette association entre une lettre et son nombre de points.

[ ]:
```
scrabble = {'A': 1, 'B': 3, 'C': 3, 'D': 2, 'E': 1, 'F': 4, 'G': 2, 'H': 4, 'I': 1, 'J': 8, 'K': 10, 'L': 1, 'M': 2, 'N': 1, 'O': 1, 'P': 3, 'Q': 8, 'R': 1, 'S': 1, 'T': 1, 'U': 1, 'V': 4, 'W': 10, 'X': 10, 'Y': 10, 'Z': 10}
```

Ecrivez une fonction `points(mot)` qui renvoie le nombre de points au scrabble de `mot`, qui est une chaîne de caractères majuscules.

Par exemple, le mot "ARBRE" doit rapporter 7 points, le mot "XYLOPHONE" doit rapporter 32 points.

[ ]:

23.9 Exercice 23.6

On considère la variable `personnages` suivante qui réunit quelques informations sur des personnalités (les âges sont fictifs, vous l'aurez compris).

[ ]:
```
personnages = [{'nom': 'Einstein', 'prénom': 'Albert', 'âge': '35', 'genre': 'm'},
               {'nom': 'Hamilton', 'prénom': 'Margaret', 'âge': '23', 'genre': 'f'},
               {'nom': 'Nelson', 'prénom': 'Ted', 'âge': '64', 'genre': 'm'},
               {'nom': 'Curie', 'prénom': 'Marie', 'âge': '41', 'genre': 'f'}]
```

1. Quel est le type de la variable `personnages`? Quel est le type des éléments de `personnages` ?

Réponse :

2. Quelle instruction permet d'accéder au dictionnaire de Ted Nelson ?

[ ]:

3. Quelle instruction permet d'accéder à l'âge de Ted Nelson ?

[ ]:

4. Dans le programme suivant, quel est le type de la variable `p` à chaque tour de boucle ? Quel est le rôle de ce programme ?

```
for p in personnages:
    if int(p['âge']) <= 40:
        print(p['nom'], p['prénom'])
```

Réponse :

5. Proposez un programme qui affiche uniquement les noms et prénoms des femmes du tableau `personnages`.

[ ]:

6. Ecrivez une fonction `age_moyen(personnages)` qui renvoie l'âge moyen des personnalités du tableau `personnages` entré en paramètre. On doit trouver 40,75 ans.

[ ]: