Les dictionnaires

Un dictonnaire (type dict) est un type construit (données composites) mutable de la même manière qu'une liste. La principale différence est que les indices ne sont pas obligatoirement des entiers (0, 1, 2, 3 ...) et peuvent être de type str, float.
Ces indices ne sont pas ordonnés et s'appellent des clés. A chaque clé correspond une valeur.
Ce type de données sera privilégié lors de manipulation de tableau de données non homogènes (mélangeant des chaînes de caractères et des entiers).

1. Construction d'un dictionnaire

Un dictionnaire peut être construit à partir d'un dictinnaire vide, noté { } et en y ajoutant des entrées avec des affectations de la forme d[clé] = valeur

```
>>> dico = {}
>>> dico['yes'] = 'oui'
>>> dico['no'] = 'non'
>>> dico['and'] = 'et'
>>> dico['or'] = 'ou'
>>> dico
{'or' : 'ou', 'and' : 'et', 'no' : 'non', 'yes' : 'oui'}
```

Il est aussi possible de définir directement entre accolades, l'ensemble de clés : valeurs séparés par des virgules.

```
>>> dico = {'or' : 'ou', 'and' : 'et', 'no' : 'non', 'yes' : 'oui'}
>>> dico
{'and' : 'et', 'or' : 'ou', 'no' : 'non', 'yes' : 'oui'}
```

Remarque : L'ordre d'insertion n'est pas important. Le dictionnaire est affiché en présentant les clés dans un ordre arbitraire, qui n'est ni l'ordre d'insertion, ni l'ordre alphabétique.

2. Manipulation des dictionnaires

Accès à une valeur

L'accès à une valeur s'effectue à l'aide de sa clé avec la construction d[cle]

```
>>> dico = {'or' : 'ou', 'and' : 'et'}
>>> dico['and']
'et'
```

Si on tente d'obtenir la valeur associée à une clé qui n'est pas dans le dictionnaire, l'erreur suivante apparaît :

```
>>> dico['ou']
Traceback (most recent call last):
  File "<pyshell>",line 1, in <module>
KeyError: 'ou'
```

Afin d'éviter cette erreur qui a pour effet d'arrêter le programme, il est possible d'utiliser la méthode get. Cette méthode renvoie la valeur associée à la clé si elle existe, sinon elle renvoie la valeur None.

```
>>> dico = {'or' : 'ou', 'and' : 'et',
'no' : 'non'}
>>> dico.get('and')
'et'
>>> dico.get('ou')
None
```

Test de valeur

On peut tester si le dictionnaire possède une entrée pour une certaine clé avec la construction clé in d

```
>>> dico = {'or' : 'ou', 'and' : 'et'}
>>> 'and' in dico
True
>>> 'yes' in dico
False
```

Suppression d'éléments

La fonction del() supprime sur place un élément du dictionnaire repéré par sa clé.

```
>>> dico = {'or' : 'ou', 'and' : 'et'}
>>> del(dico['or'])
>>> dico
 {'and' : 'et'}
```

Longueur d'un dictionnaire

La fonction len() permet comme tout type construit (string, liste, tuple ...), de connaître la longueur d'un dictionnaire

```
>>>dico = {'or' : 'ou', 'and' : 'et'}
>>> len(dico)
2
```

Parcours d'un dictionnaire

On peut parcourir toutes les clés d'un dictionnaire avec la boucle for , l'ordre de parcours étant arbitraire. On peut ainsi afficher le contenu d'un dictionnaire dico avec la boucle

```
for i in dico :
     print('la clé', i, 'est associée
     à la valeur', dico[i])
```

3.Opérations sur les dictionnaires

Des opérateurs spécifiques de filtrage de données existent afin de comparer deux dictionnaires. Ces opérateurs sont associés à des méthodes qui donnent accès à l'ensemble des couples clé/valeur (méthode item) ou plus spécifiquement à l'ensemble des clés (méthode keys) ou des valeurs (méthode values).

```
>>> dico = {'or' : 'ou', 'and' : 'et'}
>>> dico.items()
dict_items([('or' : 'ou'), ('and' : 'et')])
>>> dico.keys()
dict_keys(['or','and'])
>>> dico.values()
dict_values(['ou','et'])
```

Comparaison de dictionnaires

L'opérateur & permet de trouver des clés communes à deux dictionnaires

```
>>> dico1 = {'one': 1,'two': 2}
>>> dico2 = {'two': 2, 'three': 3}
>>> dico1.keys() & dico2.keys()
{'two'}
```

De la même manière on peut comparer des couples clé / valeurs communs.

```
>>> dico1.items() & dico2.items()
{('two', 2)}
```

Concaténation de dictionnaires

La méthode update() permet de fusionner sur place deux dictionnaires

```
>>> dico1.update(dico2)
>>> dico1
{'one': 1, 'two': 2, 'three': 3}
```