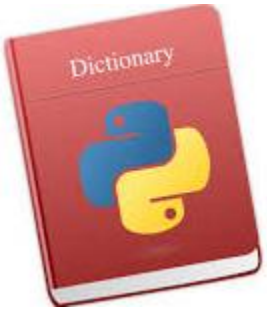
Chapitre 13 - Les dictionnaires en python



Les listes permettent de stocker des données lorsqu'elles sont nombreuses. Les éléments stockés dans une liste sont ordonnés. On y accède en utilisant un numéro qu'on appelle l'**indice** ou l'**index** de l'élément.

Un **dictionnaire** en Python va aussi permettre de rassembler des éléments mais ceux-ci seront identifiés par une **clé**. On peut faire l'analogie avec un dictionnaire de français où on accède à une définition avec un mot.

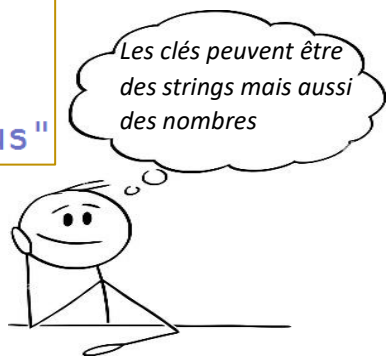
Contrairement aux listes qui sont délimitées par des crochets, on utilise des **accolades** pour les dictionnaires.

Remarque : Dans la plupart des langages de programmation, les listes sont appelées « *tableau associatif* ».

1- COMMENT CREER UN DICTIONNAIRE :

a. CREER UN DICTIONNAIRE VIDE ET LE REMPLIR :

```
hanks = {}
hanks["nom"] = "HANKS"
hanks["prenom"] = "Tom"
hanks["naissance"] = 1956
hanks["email"] = "tom.hanks@hollywood.us"
```



b. CREER UN DICTIONNAIRE AVEC SON CONTENU :

```
wright = {
    "nom": "WRIGHT",
    "prenom":"Robin",
    "dateNaissance" : 1966,
    "email" : "robin.wright@hollywwood.us"
}
```

c. MIXTE DES 2 METHODES PRECEDENTES :

```
sinise = {
    "nom": "SINISE",
    "prenom":"Gary",
    "dateNaissance" : 1955
}
sinise["email"] = "gary.sinise@hollywwood.us"
```

2- COMMENT LIRE UN ELEMENT DU DICTIONNAIRE :

```
>>> hanks["nom"]
'HANKS'
```

```
>>> hanks["email"]
'tom.hanks@hollywood.us'
```

3- COMMENT MODIFIER UN ELEMENT DU DICTIONNAIRE :

```
>>> hanks["email"] = "tom.hanks@gmail.com"

>>> hanks["email"]
'tom.hanks@gmail.com'
```

4- COMMENT PARCOURIR LES ELEMENTS D'UN DICTIONNAIRE PAR CLES :

```
for cles in hanks :
    print(cles)
```

donne

```
>>> (executing file "cours.py")
nom
prenom
naissance
email
```

```
for cles in hanks :
    print(hanks[cles])
```
donne

```
>>> (executing file "cours.py")
HANKS
Tom
1956
tom.hanks@hollywood.us
```

a. COMMENT SAVOIR SI UNE CLE EST DANS UN DICTIONNAIRE *:*

```
>>> "nom" in hanks
True
```

```
>>> "telephone" in hanks
False
```

5- COMMENT PARCOURIR LES ELEMENTS D'UN DICTIONNAIRE CLES :VALEURS

```
for cles in hanks :
    print(f" clé :  {cles} , valeur : {hanks[cles]}")
```

Variante

```
for key,value  in hanks.items():
    print( f" clé :  {key} , valeur : {value}")
```

Donne

```
clé :  nom , valeur : HANKS
clé :  prenom , valeur : Tom
clé :  datedenaissance , valeur : 1956
clé :  email , valeur : tom.hanks@hollywoood.us
```

6- UN DICTIONNAIRE PEUT-IL CONTENIR DES DICTIONNAIRES :

Un dictionnaire contient des éléments repérés par des clés.
Ces éléments peuvent être eux-mêmes des dictionnaires.
Ces éléments peuvent aussi être des listes ou des objets quelconques.

```
forestGump = {
    "forest" : hanks,
    "jenny" : wright,
    "lieutenantDan" : sinise
}
```

La commande `>>> forestGump["jenny"]` exécutée dans la console donne le résultat suivant :

La commande `>>> forestGump["jenny"]["prenom"]` exécutée dans la console donne le résultat suivant :

7- COMMENT SUPPRIMER DES VALEURS

```
>>> del forestGump["jenny"]
```

```
>>> forestGump.clear()
```

```
>>> del forestGump
```

8- EXERCICES :

Exercice 1 : Problème de casse ?

En informatique, la casse désigne le fait de distinguer les lettres majuscules des lettres minuscules.

Le code ci-contre permet de créer 2 dictionnaires :

Ce code est exécuté. Pour chacune des commandes données dans le tableau ci-dessous, indiquer le résultat de l'exécution :

```
def casse() :
    minuscules = 'abcdefghijklmnopqrstuvwxyz'
    majuscules = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
    upper = {}
    lower = {}
    for i in range(26) :
        l = minuscules[i]
        L = majuscules[i]
        upper[l] = L
        lower[L] = l
    return upper , lower

# main --------------------
upper , lower = casse()
```

Commande	Résultat
`upper['a']`	
`upper['A']`	
`lower['A']`	
`len(upper)`	
`for c in upper :` `  print(c , end = " ")`	
`for c in upper :` `  print(c , upper[c] , end = " ")`	
`upper.clear()` `print(upper)`	

Exercice 2 : Changer les lettres d'un mot ?

On complète le code précédent :

```
def casse() :
    minuscules = 'abcdefghijklmnopqrstuvwxyz'
    majuscules = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
    upper = {}
    lower = {}
    for i in range(26) :
        l = minuscules[i]
        L = majuscules[i]
        upper[l] = L
        lower[L] = l
    return upper , lower

def majuscule(mot) :
    new = ''
    for c in mot :
        new = new + upper[c]
    return new

# main ---------------------
upper , lower = casse()
m = majuscule("bonjour")
```

1- Quelle valeur contient la variable *m* après exécution ?

2- Compléter le tableau ci-dessous donnant le contenu des variables au cours de l'exécution :

mot	***new***	***c***

Exercice 3 : L'informatique ça peut servir !

Quel résultat à l'écran donne l'exécution de ce code ?

```
def anglais():
    uk = {}
    uk['je'] = 'i'
    uk['aime'] = 'love'
    uk['toi'] = 'you'
    return uk

def allemand():
    d = {'je':'ich' , 'aime' : 'liebe' , 'toi' : 'dich'}
    return d

def italien():
    d = {}
    d['je'] = "io"
    d['aime'] = 'amore'
    d['toi'] = 'voi'
    return d

def traduction(m1 , m2 , m3):
    print(uk[m1],uk[m2],uk[m3])
    print(g[m1],g[m2],g[m3])
    print(it[m1],it[m2],it[m3])

# main
uk = anglais()
g = allemand()
it = italien()
traduction("je","aime","toi")
```