

## PARTIE 3 : Les dictionnaires

### 1. Définitions et propriétés



#### DÉFINITION 1 :

Les dictionnaires en Python permettent d'associer des valeurs à des clés. A partir d'une clé, on peut alors accéder directement à la valeur qui lui est associée.

✓ EXEMPLE 1 : On considère un répertoire téléphonique qui associe à des noms des numéros de téléphone.

```
contacts = {"Alain": "0708091011", "Bernard": "0610090807", "Charles": "0609080710"}
```

Le dictionnaire contacts a été défini en extension mais on aurait pu à partir d'un dictionnaire vide ajouter des éléments au fur et à mesure, par exemple :

```
contacts = {}
contacts["Alain"] = "0708091011"
contacts["Bernard"] = "0610090807"
contacts["Charles"] = "0609080710"
```



**PROPRIÉTÉ 1 :** Il n'y a pas de structure d'ordre dans un dictionnaire d'ailleurs à l'affichage les éléments n'apparaissent pas forcément dans l'ordre d'insertion dans le dictionnaire.



**PROPRIÉTÉ 2 :** Les clés et leurs valeurs sont ici des chaînes de caractères. Ils sont donc déclarés entre guillemets. Il est aussi possible d'avoir d'autres types.

### 2. Interaction avec les dictionnaires

#### 2.1. Accéder aux valeurs d'un dictionnaire

```
>>> contacts["Alain"]
'0708091011'
```

### 3. Parcourir les clés ou les valeurs d'un dictionnaire

```
# Parcourir les clés d'un dictionnaire
>>> contacts.keys()
dict_keys(['Bernard', 'Charles', 'Alain'])

# Parcourir les valeurs d'un dictionnaire
>>> contacts.values()
dict_values(['0610090807', '0609080710', '0606060606'])
```

Les dictionnaires sont itérables, on peut donc les parcourir sur les clés, les valeurs ou les couples clé/valeur.

```
# Parcours de dictionnaire sur les clés
for cle in contacts.keys():
    print(cle)
```

```
>>>
Alain
Bernard
Charles
```

```
# Parcours de dictionnaire sur les valeurs
for val in contacts.values():
    print(val)
```

```
>>>
0708091011
0610090807
0609080710
```

```
# Parcours de dictionnaire sur les clés et les valeurs
for (cle ,val) in contacts.items():
    print(cle ,"->",val)
```

```
>>>
Alain -> 0708091011
Bernard -> 0610090807
Charles -> 0609080710
```

#### 4. Vérifier si une valeur ou une clé est contenue dans le dictionnaire

```
# Vérification si une clé est incluse dans le dictionnaire
'Charles' in contacts.keys() # Renvoie True
```

```
# Vérification si une valeur est incluse dans le dictionnaire
'0708091011' in contacts.values() # Renvoie True
```

#### 5. Dictionnaires par compréhension

Pour les dictionnaires, la syntaxe est équivalente aux listes. Il faut préciser la clé et la valeur pour chaque élément.


✓ **EXEMPLE 2** : Dictionnaire contenant pour les clés les nombres entiers de 2 à 10 et comme valeur associées le carré de la clé.

```
>>> dico = {k: k ** 2 for k in range(2 ,11)}

>>> dico
{2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100}
```

## 6. Opérations sur les dictionnaires

Opération	Résultat
<code>s[k]</code>	Renvoie la valeur associée à la clé <code>k</code>
<code>x in s.values()</code>	Renvoie <code>True</code> si une valeur de <code>s</code> est égale à <code>x</code> , <code>False</code> sinon
<code>x not in s.values()</code>	Renvoie <code>True</code> si aucune valeur de <code>s</code> n'est égale à <code>x</code> , <code>False</code> sinon
<code>x in s.keys()</code>	Renvoie <code>True</code> si une clé de <code>s</code> est égale à <code>x</code> , <code>False</code> sinon
<code>s[k] = v</code>	Modifie la valeur <code>v</code> associée à la clé <code>k</code> ou l'ajoute si elle n'existe pas déjà
<code>s.get(k, v)</code>	Renvoie la valeur associée à la clé <code>k</code> . Si la clé <code>k</code> n'existe pas, renvoie la valeur <code>v</code>
<code>s.pop(k)</code>	Enlève du dictionnaire la clé <code>k</code> et renvoie la valeur associée

 Exercice 10 : On considère le dictionnaire suivant :


```
dico2 = {"a" : True, "b" : False , "c" : False}
```

1. Quelle est la valeur de `dico2[1]` ?
2. Quelle est la valeur de `dico2["a"]` ?
3. Quelle instruction permet de modifier le dictionnaire afin que sa nouvelle valeur soit :  
**`dico2 = "a" : True, "b" : False, "c" : False, "e" : True?`**
4. Que permet d'afficher le script suivant ?

```
for cle in dico2.keys():
    print(cle , end=" ")
```

5. Que permet d'afficher le script suivant ?

```
for truc in dico2. items () :
    print ( truc , end = " ")
```

 Exercice 11 : on modélise des informations (nom, taille, poids) sur des Pokémons de la façon suivante :

```
exemples_pokemons = {'Bulbizarre' : (70 ,7), 'Herbizarre' : (100 ,13), 'Abo' : (200 ,7), 'Jungko' : (170 ,52)}
```

Par exemple, Bulbizarre est un Pokémon qui mesure 70 cm et pèse 7 kg.

1. Ajouter à cette structure de données le Pokémon Goupix qui mesure 60 cm et pèse 10 kg. Noter ici l'instruction tapée :


2. On donne le code suivant :

```
def lePlusGrand(pokemon):
    grand = None
    taille_max = None
    for (nom , (taille ,poids)) in pokemon.items():
        if taille_max is None or taille > taille_max:
            taille_max = taille
            grand = nom
    return (grand , taille_max)
```

Écrire le code de la fonction `lePlusLeger()` qui renvoie un tuple dont la première composante est le nom du Pokémon le plus léger et la deuxième composante son poids.

3. Écrire le code de la fonction `taille()` qui prend en paramètre un dictionnaire de Pokémons ainsi que le nom d'un Pokémon et qui renvoie la taille de ce Pokémon.

```
assert taille(exemples_pokemons , 'Abo') == 200
assert taille(exemples_pokemons , 'Jungko') == 170
assert taille(exemples_pokemons , 'Dracaufeu ') is None
```

 Exercice 12 : Au zoo de Beauval, il y a 5 éléphants d'Asie, 17 écureuils d'Asie, 2 pandas d'Asie ...

On représente et inventaire à l'aide d'un dictionnaire de la façon suivante :

```
zoo_Beauval={'éléphant' : ('Asie' ,5), 'écureuil' : ('Asie' ,17), 'panda' : ('Asie' ,2), 'hippopotame' : ('Afrique' ,7), 'girafe' : ('Afrique' ,4)}
```

On fait de même avec le zoo de La Flèche :

```
zoo_LaFleche={'ours' : ('Europe' ,4), 'tigre' : ('Asie' ,7), 'girafe' : ('Afrique' ,11), 'hippopotame' : ('Afrique' ,3)}
```

On souhaite créer une fonction `nombreTotal()` qui prend en paramètre un zoo ainsi que le nom d'un continent et qui renvoie le nombre d'animaux originaire de ce continent dans ce zoo. par exemple :

```
assert nombreTotal(zoo_LaFleche , 'Afrique') == 14
assert nombreTotal(zoo_Beauval , 'Asie') == 24
```

1. Ecrire le code de cette fonction

2. On souhaite créer une fonction `nombre()` qui prend en paramètre un zoo ainsi que le nom d'un animal et qui renvoie le nombre de représentants de cet animal dans ce zoo. Par exemple :

```
assert nombre(zoo_LaFleche , 'panda') == None
assert nombre(zoo_Beauval , 'panda') == 2
```

Ecrire le code de cette fonction