

## TP 11 : les dictionnaires en Python : La gestion des données, version maître du monde !



### Introduction : C'est quoi un dictionnaire en Python ?

Alors, imagine que tu as un tas de données, genre plein de trucs à gérer (les noms des élèves, les notes, les couleurs préférées, les pizzas qu'ils aiment...). Si tu utilises juste une liste, c'est comme essayer de retrouver une épingle dans une botte de foin. **Mais** si tu utilises un **dictionnaire**, c'est comme avoir une armoire bien rangée où tu sais exactement quel tiroir contient tes chaussettes dépareillées.

En gros, un **dictionnaire** en Python, c'est une **structure de données** où chaque élément est une paire **clé-valeur**. La clé, c'est ce que tu cherches, et la valeur, c'est ce que tu trouves.

### Les bases : Comment on déclare un dictionnaire ?

Très simple :

```
mon_dico = {  
    'nom': 'Alice',  
    'âge': 25,  
    'pays': 'Wonderland'  
}
```

Ici, on a trois paires clé-valeur :

- **'nom'** est la clé et **'Alice'** est la valeur.
- **'âge'** est la clé et **25** est la valeur.
- **'pays'** est la clé et **'Wonderland'** est la valeur.

### Accéder à une valeur (ou : "Eh, rends-moi mon biscuit!")

Pour accéder à une valeur, il suffit d'utiliser la clé :

```
print(mon_dico['nom']) # Résultat : Alice
```

Et si tu demandes une clé qui n'existe pas... Python panique :

```
print(mon_dico['poids']) # KeyError !
```

Astuce : Utilise `.get()` pour éviter les catastrophes :

```
print(mon_dico.get('poids', 'Clé inconnue')) # Résultat : Clé inconnue
```

### Ajouter ou modifier des éléments (aka : "Range ce bazar!")

Modifier une valeur ou ajouter une nouvelle paire clé-valeur est aussi simple que de passer l'aspirateur (ou presque) :

```
mon_dico['âge'] = 26 # Modification de la valeur  
mon_dico['hobby'] = 'Chasse aux lapins' # Ajout d'une nouvelle clé
```

## Supprimer des éléments (ou : "Dégage l'élément!")

Pour supprimer une clé et sa valeur, c'est un peu comme nettoyer une étagère :

```
del mon_dico['pays']
```

## Parcourir un dictionnaire (ou : "Explorons cette jungle de données!")

Utiliser des boucles avec des dictionnaires, c'est comme jouer à Indiana Jones dans un temple de données :

- Parcourir juste les clés :

```
for clé in mon_dico:  
    print(clé)
```

- Parcourir les valeurs :

```
for valeur in mon_dico.values():  
    print(valeur)
```

- Parcourir les paires clé-valeur :

```
for clé, valeur in mon_dico.items():  
    print(f'Clé: {clé}, Valeur: {valeur}')
```

## Les dictionnaires imbriqués (ou : "Inception version Python")

Tu peux avoir un dictionnaire dans un dictionnaire (un peu comme des poupées russes) :

```
classe = {  
    'élève1': {'nom': 'Alice', 'note': 15},  
    'élève2': {'nom': 'Bob', 'note': 18}  
}
```

Pour accéder à une sous-valeur :

```
print(classe['élève1']['nom']) # Résultat : Alice
```

Savoir si un nom est présent dans un sous-dictionnaire :

```
def nom_existe(classe, nom_recherche):  
    for eleve, infos in classe.items():  
        if infos['nom'] == nom_recherche:  
            return True  
    return False
```

## Quelques méthodes utiles :

- `len()` pour connaître la taille du dico :

```
print(len(mon_dico)) # Combien de paires clé-valeur ?
```

- **keys()**, **values()**, **items()** pour obtenir les clés, les valeurs, ou les deux !

```
print(mon_dico.keys()) # Toutes les clés  
print(mon_dico.values()) # Toutes les valeurs
```