

# Leçon : Structures de données en Python

## Introduction


En Python, il existe plusieurs structures de données intégrées qui sont essentielles pour organiser et stocker les données. Les quatre principales structures de données sont les listes, les tuples, les dictionnaires et les ensembles (sets). Cette leçon couvrira chacune de ces structures avec des exemples pratiques.

## Listes

Une liste est une collection ordonnée et modifiable. Elle permet de stocker des éléments de différents types (entiers, flottants, chaînes de caractères, etc.).

### Exemple de liste contenant des types différents

python

 Copy code


```
ma_liste = [1, 2, 4, 8.9, 'r', True, 'abc']  
print(ma_liste)
```

### Accéder aux éléments d'une liste

Les éléments d'une liste sont indexés, ce qui signifie que vous pouvez accéder à chaque élément en utilisant son index (commençant à 0).

#### Accéder au premier élément


python

 Copy code

```
premier_element = ma_liste[0]  
print(premier_element)
```

## Accéder au dernier élément

python


 Copy code

```
dernier_element = ma_liste[-1]
print(dernier_element)
```

## Découpage de liste

Le découpage permet de créer une nouvelle liste contenant une partie de la liste originale.

python

 Copy code


```
sous_liste = ma_liste[1:4]
print(sous_liste)
```

## Tuples

Un tuple est similaire à une liste, mais il est immuable, ce qui signifie que ses éléments ne peuvent pas être modifiés après la création.

### Exemple de tuple

python


 Copy code

```
mon_tuple = (1, 2, 3, 'a', 'b', 'c')
print(mon_tuple)
```

### Accéder aux éléments d'un tuple

Comme les listes, les éléments d'un tuple sont indexés.

python

 Copy code


```
premier_element_tuple = mon_tuple[0]
print(premier_element_tuple)
```

# Dictionnaires

Un dictionnaire est une collection non ordonnée de paires clé-valeur. Chaque clé doit être unique.

## Exemple de dictionnaire


python

 Copy code

```
mon_dictionnaire = {'nom': 'Alice', 'âge': 25, 'ville': 'Paris'}  
print(mon_dictionnaire)
```

## Accéder aux valeurs via les clés


python

 Copy code

```
nom = mon_dictionnaire['nom']  
print(nom)
```

## Ajouter une nouvelle paire clé-valeur

python

 Copy code


```
mon_dictionnaire['profession'] = 'Ingénieur'  
print(mon_dictionnaire)
```

# Ensembles (Sets)

Un ensemble est une collection non ordonnée d'éléments uniques.

## Exemple d'ensemble


python

 Copy code

```
mon_ensemble = {1, 2, 3, 4, 5}  
print(mon_ensemble)
```

## Ajouter un élément à un ensemble


python

 Copy code

```
mon_ensemble.add(6)  
print(mon_ensemble)
```

## Ajouter un élément à un ensemble


python

 Copy code

```
mon_ensemble.add(6)
print(mon_ensemble)
```

## Enlever un élément d'un ensemble

python

 Copy code

```
mon_ensemble.remove(3)
print(mon_ensemble)
```

## Conclusion

Les structures de données en Python sont puissantes et flexibles, permettant de gérer efficacement les données dans divers scénarios. En comprenant et en utilisant les listes, tuples, dictionnaires et ensembles, vous pouvez organiser et manipuler vos données de manière optimale.