


# Classement de Strings et Tuples en Python

## Classement de Strings

*Nous avons une liste de prénoms que nous souhaitons trier selon différents critères.*

*Liste de prénoms*

python


 Copy code

```
names = [  
    "Alice",  
    "Bob",  
    "Charlie",  
    "Diana",  
    "Ethan",  
    "Fiona",  
    "George",  
    "Hannah",  
    "Ian",  
    "Jasmine"  
]
```

## Classement par le premier caractère

*Nous définissons une fonction `classement` qui retourne le premier caractère du prénom, puis utilisons cette fonction comme clé de tri.*

python


 Copy code

```
def classement(name):  
    return name[0]  
  
liste_triee = sorted(names, key=classement)  
print(liste_triee)  
# Output: ['Alice', 'Bob', 'Charlie', 'Diana', 'Ethan', 'Fiona', 'George', 'Hannah']
```

## Classement par le dernier caractère

Cette fois, la fonction `classement` retourne le dernier caractère du prénom.

python


 Copy code

```
def classement(name):  
    return name[-1]  
  
liste_triee = sorted(names, key=classement)  
print(liste_triee)  
# Output: ['Diana', 'Fiona', 'Bob', 'Alice', 'Charlie', 'George', 'Jasmine', 'Hanna
```

## Classement par la longueur du prénom

La fonction `classement` retourne la longueur du prénom.

python


 Copy code

```
def classement(name):  
    return len(name)  
  
liste_triee = sorted(names, key=classement)  
print(liste_triee)  
# Output: ['Bob', 'Ian', 'Alice', 'Diana', 'Ethan', 'Fiona', 'George', 'Hannah', 'O
```

## Classement de Tuples Nous avons une liste de tuples, où chaque tuple contient un prénom et un âge. Nous allons trier cette liste par âge.

### Liste de personnes avec âge

python


 Copy code

```
people = [  
    ("Alice", 28),  
    ("Bob", 34),  
    ("Charlie", 22),  
    ("Diana", 29),  
    ("Ethan", 31),  
    ("Fiona", 26),  
    ("George", 40),  
    ("Hannah", 25),  
    ("Ian", 38),  
    ("Jaasmine", 27)  
]
```

## Classement par âge (croissant)

La fonction `c_lassement` retourne l'âge de la personne.

python

 Copy code


```
def c_lassement(personne):  
    return personne[1]  
  
personnes_triees = sorted(people, key=c_lassement)  
print(personnes_triees)  
# Output: [('Charlie', 22), ('Hannah', 25), ('Fiona', 26), ('Jaasmine', 27), ('Alicia', 28)]
```

”

## Classement par âge (décroissant)

En passant l'argument `reverse=True` à `sorted`, nous pouvons trier en ordre décroissant.

python


 Copy code

```
personnes_triees = sorted(people, key=c_lassement, reverse=True)  
print(personnes_triees)  
# Output: [('George', 40), ('Ian', 38), ('Bob', 34), ('Ethan', 31), ('Diana', 29), ('Charlie', 22), ('Hannah', 25), ('Fiona', 26), ('Jaasmine', 27), ('Alicia', 28)]
```

## Trouver le tuple avec le plus grand âge

Utilisation de `max` avec la fonction `c_lassement`

python


 Copy code

```
personne_max = max(people, key=c_lassement)  
print(personne_max)  
# Output: ('George', 40)
```

## Trouver le tuple avec le prénom le plus court

La fonction `c_lassement` retourne la longueur du prénom.

python

 Copy code


```
def c_lassement(personne):  
    return len(personne[0])  
  
personne_min = min(people, key=c_lassement)  
print(personne_min)  
# Output: ('Bob', 34)
```

# Fonctions Lambda

Les fonctions lambda permettent de définir des fonctions anonymes de manière concise.

## Définition normale d'une fonction


python

 Copy code

```
def say_hello(name):  
    return "Bonjour " + name  
  
print(say_hello("florian"))  
# Output: Bonjour florian
```

## Définition d'une fonction en lambda


python

 Copy code

```
say_hello = lambda name: "Bonjour " + name  
print(say_hello("florian"))  
# Output: Bonjour florian
```

## Application des lambdas pour le tri de tuples

python

 Copy code


```
# Liste des personnes avec leur âge
people = [
    ("Alice", 28),
    ("Bob", 34),
    ("Charlie", 22),
    ("Diana", 29),
    ("Ethan", 31),
    ("Fiona", 26),
    ("George", 40),
    ("Hannah", 25),
    ("Ian", 38),
    ("Jaasmine", 27)
]

# Trier les personnes par âge en utilisant une fonction lambda
people_tries = sorted(people, key=lambda personne: personne[1])

# Afficher la liste triée de manière lisible
for person in people_tries:
    print(f"Name: {person[0]}, Age: {person[1]}")
```

Sortie attendue :

yaml

 Copy code

```
Name: Charlie, Age: 22
Name: Hannah, Age: 25
Name: Fiona, Age: 26
Name: Jaasmine, Age: 27
Name: Alice, Age: 28
Name: Diana, Age: 29
Name: Ethan, Age: 31
Name: Bob, Age: 34
Name: Ian, Age: 38
Name: George, Age: 40
```