# Gestion des fichiers

# Gestion des fichiers : ouverture ,lecture  et fermeture

Avant d'être en mesure de parcourir ou de rédiger dans un fichier, il est essentiel de l'initialiser à l'aide de la fonction open(). Cette fonction requiert le chemin du fichier ainsi que le mode d'ouverture en tant qu'arguments. Une fois que vous avez accompli vos opérations sur le fichier, il est primordial de le refermer en faisant appel à la méthode close() afin de libérer les ressources qui y sont liées.

```
# Ouverture du fichier en mode lecture
fichier = open("chemin/vers/fichier.txt", "r")

# Lecture du contenu du fichier
contenu = fichier.read()
print(contenu)

# Fermeture du fichier
fichier.close()
```

# Gestion des fichiers : ouverture ,lecture  et fermeture

Les divers modes d'ouverture de fichiers en Python sont les suivants :
- Mode lecture ('r') : Permet de lire le contenu d'un fichier existant.
- Mode écriture ('w') : Permet d'écrire dans un fichier, en le remplaçant s'il existe déjà.
- Mode ajout ('a') : Permet d'ajouter du contenu à la fin d'un fichier existant ou de le créer s'il n'existe pas.
- Mode lecture et écriture ('r+') : Permet la lecture et l'écriture dans un fichier existant.
- Mode écriture et création ('w+') : Permet l'écriture et la lecture dans un fichier, en le remplaçant s'il existe déjà.
- Mode ajout et création ('a+') : Permet l'ajout et la lecture dans un fichier existant ou la création d'un nouveau fichier.

# Gestion des fichiers : ouverture ,lecture  et fermeture

Vous pouvez lire un fichier ligne par ligne en utilisant la méthode readline() ou en itérant directement sur l'objet fichier à l'aide d'une boucle for. Cela est particulièrement utile pour les fichiers volumineux ou lorsque vous souhaitez traiter chaque ligne individuellement.

```
fichier = open("chemin/vers/fichier.txt", "r")

# Lecture ligne par ligne
ligne = fichier.readline()
while ligne:
    print(ligne)
    ligne = fichier.readline()

# Fermeture du fichier
fichier.close()
```

# Gestion des fichiers : lecture ligne par ligne

Vous pouvez lire un fichier ligne par ligne en utilisant la méthode readline() ou en itérant directement sur l'objet fichier à l'aide d'une boucle for. Cela est particulièrement utile pour les fichiers volumineux ou lorsque vous souhaitez traiter chaque ligne individuellement.

```
fichier = open("chemin/vers/fichier.txt", "r")

# Lecture ligne par ligne
ligne = fichier.readline()
while ligne:
    print(ligne)
    ligne = fichier.readline()

# Fermeture du fichier
fichier.close()
```

```
fichier = open("chemin/vers/fichier.txt", "r")

# Lecture ligne par ligne avec une boucle for
for ligne in fichier:
    print(ligne)

# Fermeture du fichier
fichier.close()
```

# Gestion des fichiers : ecriture

Pour écrire dans un fichier, vous devez ouvrir le fichier en mode écriture ("w") ou ajout ("a").
Utilisez la méthode write() pour écrire des données dans le fichier.
- L'écriture dans un fichier en mode écriture ("w") écrasera le contenu existant
- La l'écriture en mode ajout ("a") ajoutera le contenu à la fin du fichier.

```python
fichier = open("chemin/vers/fichier.txt", "w")

# Écriture dans le fichier
fichier.write("Contenu à écrire\n")
fichier.write("Autre contenu à écrire\n")

# Fermeture du fichier
fichier.close()
```

# Gestion des fichiers : ecriture ligne par ligne

La méthode writelines() permet d'écrire une liste de chaînes de caractères dans un fichier, chaque élément de la liste étant écrit sur une nouvelle ligne.

```
fichier = open("chemin/vers/fichier.txt", "w")
contenu = ["Ligne 1\n", "Ligne 2\n", "Ligne 3\n"]

fichier.writelines(contenu)

fichier.close()
```

# Gestion des fichiers : deplacement position

La méthode seek() est utilisée pour déplacer la position de lecture/écriture dans un fichier. Elle prend en argument un décalage en octets à partir du début du fichier. Vous pouvez ensuite utiliser les méthodes de lecture ou d'écriture pour accéder aux données à partir de la nouvelle position.

```
fichier = open("chemin/vers/fichier.txt", "r")

# Déplacer la position de lecture à 10 octets à partir du début du fichier
fichier.seek(10)

# Lire le contenu à partir de la nouvelle position
contenu = fichier.read()
print(contenu)

fichier.close()
```

# Gestion des fichiers : deplacement position

La méthode tell() est utilisée pour obtenir la position de lecture/écriture actuelle dans un fichier. Elle renvoie le nombre d'octets depuis le début du fichier jusqu'à la position actuelle.

```
fichier = open("chemin/vers/fichier.txt", "r")

# Lire le contenu
contenu = fichier.read()
print(contenu)

# Obtenir la position de lecture actuelle
position = fichier.tell()
print("Position de lecture actuelle :", position)

fichier.close()
```

# Gestion des fichiers : Utilisation avancée

```
with open("chemin/vers/fichier.txt", "r", encoding="utf-8") as fichier:
    contenu = fichier.read()
    print(contenu)
```

L'utilisation du « with » garantit que les ressources associées au contexte sont correctement libérées, même en cas d'exceptions ou d'erreurs survenant à l'intérieur du bloc de code. Cela évite d'avoir à se souvenir d'appeler manuellement la méthode close() pour fermer les ressources après leur utilisation.

# Gestion des fichiers : Utilisation avancée

```
with open("chemin/vers/fichier.txt", "r", encoding="utf-8") as fichier:
    contenu = fichier.read()
    print(contenu)
```

L'utilisation du « with » garantit que les ressources associées au contexte sont correctement libérées, même en cas d'exceptions ou d'erreurs survenant à l'intérieur du bloc de code. Cela évite d'avoir à se souvenir d'appeler manuellement la méthode close() pour fermer les ressources après leur utilisation.

# Gestion des fichiers : Utilisation avancée

```
with open("chemin/vers/fichier.txt", "r", encoding="utf-8") as fichier:
    contenu = fichier.read()
    print(contenu)
```

L'utilisation du « with » garantit que les ressources associées au contexte sont correctement libérées, même en cas d'exceptions ou d'erreurs survenant à l'intérieur du bloc de code. Cela évite d'avoir à se souvenir d'appeler manuellement la méthode close() pour fermer les ressources après leur utilisation.

# Gestion des fichiers : Module Csv lecture

Pour lire un fichier CSV, vous pouvez utiliser la fonction reader() du module csv. Elle prend en argument un objet fichier ouvert et renvoie un itérable qui permet de parcourir les lignes du fichier CSV

```
Nom,Âge,Ville
Alice,25,Paris
Bob,30,Londres
```

```python
import csv

with open("data.csv", "r") as fichier:
    lecteur_csv = csv.reader(fichier)
    for ligne in lecteur_csv:
        print(ligne)
```

```
['Nom', 'Âge', 'Ville']
['Alice', '25', 'Paris']
['Bob', '30', 'Londres']
```

# Gestion des fichiers : Module CSV ecriture

Pour écrire dans un fichier CSV, vous pouvez utiliser la fonction writer() du module csv. Elle prend en argument un objet fichier ouvert et renvoie un objet écrivain qui permet d'écrire des lignes dans le fichier CSV.

```python
import csv

donnees = [
    ['Nom', 'Âge', 'Ville'],
    ['Alice', '25', 'Paris'],
    ['Bob', '30', 'Londres']
]

with open("data.csv", "w", newline='') as fichier:
    ecrivain_csv = csv.writer(fichier)
    for ligne in donnees:
        ecrivain_csv.writerow(ligne)
```

```
Nom,Âge,Ville
Alice,25,Paris
Bob,30,Londres
```

# Gestion des fichiers : Module CSV

Spécifier un délimiteur personnalisé

```python
import csv

with open("data.csv", "r") as fichier:
    lecteur_csv = csv.reader(fichier, delimiter=';')
    for ligne in lecteur_csv:
        print(ligne)
```

```python
import csv

donnees = [
    ['Nom', 'Âge', 'Ville'],
    ['Alice', '25', 'Paris'],
    ['Bob', '30', 'Londres']
]

with open("data.csv", "w", newline='') as fichier:
    ecrivain_csv = csv.writer(fichier, delimiter='\t')
    for ligne in donnees:
        ecrivain_csv.writerow(ligne)
```

# Gestion des fichiers : Module CSV : lecture avec entete

La classe DictReader du module csv permet de lire un fichier CSV en utilisant les valeurs de la première ligne comme en-têtes de colonnes. Cela facilite l'accès aux données en utilisant les noms de colonnes plutôt que les indices.

```
Nom,Âge,Ville
Alice,25,Paris
Bob,30,Londres
```

```python
with open("data.csv", "r") as fichier:
    lecteur_csv = csv.DictReader(fichier)
    for ligne in lecteur_csv:
        print(ligne['Nom'], ligne['Âge'], ligne['Ville'])
```

# Gestion des fichiers : Module CSV : ecriture avec entete

```python
import csv

donnees = [
    {'Nom': 'Alice', 'Âge': '25', 'Ville': 'Paris'},
    {'Nom': 'Bob', 'Âge': '30', 'Ville': 'Londres'}
]

with open("data.csv", "w", newline='') as fichier:
    en_tetes = ['Nom', 'Âge', 'Ville']
    ecrivain_csv = csv.DictWriter(fichier, fieldnames=en_tetes)
    ecrivain_csv.writeheader()
    for ligne in donnees:
        ecrivain_csv.writerow(ligne)
```