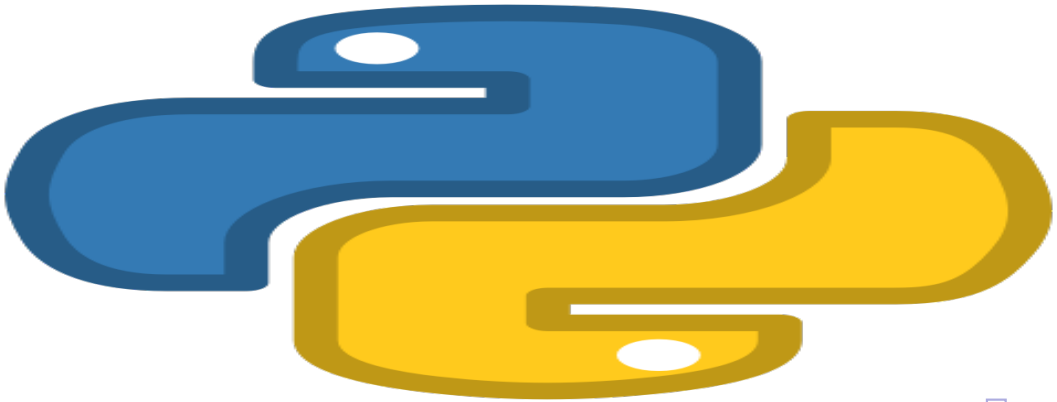
Python : gestion de fichiers

Achref El Mouelhi

Docteur de l'université d'Aix-Marseille
Chercheur en programmation par contrainte (IA)
Ingénieur en génie logiciel

elmouelhi.achref@gmail.com



Plan

Python

Les fichiers

- outil utilisé pour stocker et/ou échanger les données
- pouvant être en écriture ou en lecture

Python

Les fichiers

- outil utilisé pour stocker et/ou échanger les données
- pouvant être en écriture ou en lecture

3 étapes pour la manipulation de fichiers en Java

- Création ou ouverture
- Utilisation : écriture ou lecture
- Fermeture

Python

Plusieurs modes d'ouverture en **Python**

- `r` : ouverture en lecture seule (read) [**mode par défaut**]
- `w` : ouverture en écriture seule (write). Si le fichier n'existe pas, il sera créé. Le nouveau contenu écrasera le précédent.
- `a` : ouverture en écriture (append). Si le fichier n'existe pas, il sera créé. Le nouveau contenu sera ajouté à la fin du fichier.
- `r+` : ouverture en lecture et écriture. Le fichier doit exister au préalable.
- `w+` : ouverture en lecture et écriture. Si le fichier n'existe pas, il sera créé. Le nouveau contenu écrasera le précédent.
- `a+` : ouverture en lecture et ajout à la fin du fichier. Si le fichier n'existe pas, il sera créé.

Python

Démarche

- Créez un répertoire `cours-fichiers` dans votre espace de travail
- Lancez **VSC** et allez dans `File > Open Folder...` et choisissez `cours-fichiers`
- Dans `cours-fichiers`, créez un fichier `main.py` et un répertoire `files`

Python

Pour créer un fichier en mode écriture

```
mon_fichier = open("files/fichier.txt", "w")
```

Python

Pour créer un fichier en mode écriture

```
mon_fichier = open("files/fichier.txt", "w")
```


Remarques

- Si le fichier n'existe pas, il sera créé
- Le fichier sera créé dans le répertoire `files` (par défaut à la racine du projet)

Python

La variable créée est une instance de la classe `TextIOWrapper`

```
mon_fichier = open("files/fichier.txt", "w")

print(type(mon_fichier))
# affiche <class '_io.TextIOWrapper'>
```

Python

La variable créée est une instance de la classe `TextIOWrapper`

```
mon_fichier = open("files/fichier.txt", "w")

print(type(mon_fichier))
# affiche <class '_io.TextIOWrapper'>
```

Pour écrire dans le fichier (le contenu précédent sera écrasé), on utilise la méthode `write`

```
mon_fichier.write('bonjour')
```

Python

La variable créée est une instance de la classe `TextIOWrapper`

```
mon_fichier = open("files/fichier.txt", "w")

print(type(mon_fichier))
# affiche <class '_io.TextIOWrapper'>
```

Pour écrire dans le fichier (le contenu précédent sera écrasé), on utilise la méthode `write`

```
mon_fichier.write('bonjour')
```


Remarques

- La méthode `write()` n'accepte que le chaînes de caractères
- Pour ajouter un autre type de données, il faut faire conversion

Python

N'oublions pas de fermer le fichier

```
mon_fichier.close()
```

Python

On peut utiliser `with ... as` pour faire l'ouverture et fermeture

```
with open('files/fichier.txt', 'w') as mon_fichier:
    print(mon_fichier.write('bonjour'))

# affiche 7 (nombre de caractères écrits)
```

Python

Pour ouvrir un fichier en lecture (si le fichier n'existe pas, une exception sera levée)

```
mon_fichier = open("files/fichier.txt", "r")
```

Python

Pour ouvrir un fichier en lecture (si le fichier n'existe pas, une exception sera levée)

```
mon_fichier = open("files/fichier.txt", "r")
```

Pour lire le contenu

```
contenu = mon_fichier.read()

print(contenu)
# affiche bonjour
```

Python

Pour ouvrir un fichier en lecture (si le fichier n'existe pas, une exception sera levée)

```
mon_fichier = open("files/fichier.txt", "r")
```

Pour lire le contenu

```
contenu = mon_fichier.read()

print(contenu)
# affiche bonjour
```


Remarques

- La méthode `read()` lit tout le contenu du fichier
- On peut utiliser `split()` pour parcourir le contenu ligne par ligne ou mot par mot

Python

N'oublions pas de fermer le fichier

```
mon_fichier.close()
```

Python

On peut utiliser `with ... as` pour faire l'ouverture et fermeture

```
with open('files/fichier.txt', 'r') as mon_fichier:
    print(mon_fichier.read())

# affiche bonjour
```

Python

Pour lire le fichier ligne par ligne

```
with open('./fichier.txt', 'r') as mon_fichier:
    for ligne in mon_fichier:
        print(ligne)
```

Python

Pour spécifier le nombre de caractère à lire à chaque appel

```
nombre_caracteres = 8

with open('./fichier.txt', 'r') as mon_fichier:
    while True:
        fragment = mon_fichier.read(nombre_caracteres)
        if not fragment:
            break
        print(fragment)
```

Python

Pour spécifier le nombre de caractère à lire à chaque appel

```
nombre_caracteres = 8

with open('./fichier.txt', 'r') as mon_fichier:
    while True:
        fragment = mon_fichier.read(nombre_caracteres)
        if not fragment:
            break
        print(fragment)
```


Ou

```
nombre_caracteres = 8

with open('./fichier.txt', 'r') as mon_fichier:
    while (fragment := mon_fichier.read(nombre_caracteres)):
        print(fragment)
```

Python

Pour lire le fichier ligne par ligne, on utilise `readline()`

```
with open('./fichier.txt', 'r') as mon_fichier:
    while (line := mon_fichier.readline()):
        print(line)
```

Python

`readlines()` **stocke toutes les lignes en mémoire**

```
with open('./fichier.txt', 'r') as mon_fichier:
    lines = mon_fichier.readlines()
    for line in lines:
        print(line)
```

Python

Étant donné le fichier `file.txt` ayant le contenu suivant

```
15
12
17
13
10
16
```

Python

Étant donné le fichier `file.txt` ayant le contenu suivant

```
15
12
17
13
10
16
```


Exercice 1

Écrivez un script **Python** qui permet de calculer la moyenne des valeurs définies dans `file.txt`.

Python

Étant donné le fichier `phrases.txt` ayant le contenu suivant

```
Une première phrase.
Et voici une deuxième.
Et encore une troisième. Ciao.
```

Python

Étant donné le fichier `phrases.txt` ayant le contenu suivant

```
Une première phrase.
Et voici une deuxième.
Et encore une troisième. Ciao.
```


Exercice 2

Écrivez un script **Python** qui permet de lire les données du fichier `phrases.txt` et d'afficher le nombre total de mots, de lignes et de phrases.

Python

Étant donné le fichier `notes.txt` ayant le contenu suivant

```
wick 17 13 15
dalton 20 9 13
maggio 16 12 20
baggio 8 7 9
```

Python

Étant donné le fichier `notes.txt` ayant le contenu suivant

```
wick 17 13 15
dalton 20 9 13
maggio 16 12 20
baggio 8 7 9
```

Exercice 3

Écrivez un script **Python** qui permet de lire les données du fichier `notes.txt`, calculer la moyenne de chaque personne et l'écrire avec le nom dans `moyennes.txt`.

Python

Considérons le fichier `fichier.json` suivant

```
{
    "nom": "wick",
    "prenom": "john",
    "age": "45"
}
```

Python

Pour lire le contenu de `fichier.json` et le stocker dans un dictionnaire

```
import json

contenu = {}

with open('files/fichier.json') as f:
    contenu = json.load(f)

print(contenu)
# affiche {'nom': 'wick', 'prenom': 'john', 'age': '45'}

print(contenu['nom'])
# affiche wick

print(contenu.get('nom'))
# affiche wick
```

Python

Et si le fichier `fichier.json` était structuré ainsi

```
{
    "personnes": [
        {
            "nom": "wick",
            "prenom": "john",
            "age": "45"
        },
        {
            "nom": "wick",
            "prenom": "john",
            "age": "45"
        }
    ]
}
```

Python

Pour lire le contenu de `fichier.json`

```
import json

contenu = {}

with open('files/fichier.json') as f:
    contenu = json.load(f)


# Parcourir et afficher les clés-valeurs
for personne in contenu['personnes']:
    for cle, valeur in personne.items():
        print(f"{cle}: {valeur}")
    print("-" * 20)  # Ajoute une ligne de séparation entre les
        personnes
```

Python

Pour écrire dans `fichier.json`, on utilise la méthode `dump`

```
import json

entry = { "ville": "New York"}

with open('files/fichier.json', 'w') as f:

    json.dump(entry, f)
```

Python

Pour écrire dans `fichier.json`, on utilise la méthode `dump`

```
import json

entry = { "ville": "New York"}

with open('files/fichier.json', 'w') as f:

    json.dump(entry, f)
```


Remarque

Le contenu précédent a été écrasé.

Python

Et si on remplaçait `w` par `a`

```
import json

entry = { "ville": "New York"}

with open('files/fichier.json', 'a') as f:
    json.dump(entry, f)
```

Python

Et si on remplaçait `w` par `a`

```
import json

entry = { "ville": "New York"}

with open('files/fichier.json', 'a') as f:
    json.dump(entry, f)
```


Le résultat est

```
{
    "nom": "wick",
    "prenom": "john",
    "age": "45"
}{"ville": "New York"}
```

Python

Exercice

écrire un code **Python** qui permet d'ajouter une quatrième clé `ville` dans `fichier.json` avec la valeur `New York`

Python

Correction

```
import json

contenu = {}

with open('files/fichier.json') as f:
    contenu = json.load(f)

contenu['ville'] = 'New York'

with open('files/fichier.json', 'w') as f:
    json.dump(contenu, f)
```

Python

Considérons le fichier `fichier.xml` suivant

```
<personnes>
    <personne>
        <nom>Doe</nom>
        <prenom>John</prenom>
        <age>45</age>
    </personne>
    <personne>
        <nom>Dalton</nom>
        <prenom>Jack</prenom>
        <age>50</age>
    </personne>
</personnes>
```

Python

Pour lire le contenu de `fichier.xml`

```
import xml.etree.ElementTree as ET

# Lecture depuis un fichier XML
tree = ET.parse('files/fichier.xml')

root = tree.getroot()

for personne in root.findall('personne'):
    for info in personne:
        print(f"{info.tag}: {info.text}")
```

Python

Pour écrire dans `fichier.xml`

```
import xml.etree.ElementTree as ET

# Lecture depuis un fichier XML
tree = ET.parse('files/fichier.xml')

root = tree.getroot()

# Création d'un nouvel élément personne
nouvelle_personne = ET.SubElement(root, 'personne')

# Ajout des informations de la nouvelle personne
nom = ET.SubElement(nouvelle_personne, 'nom')
nom.text = 'Smith'

prenom = ET.SubElement(nouvelle_personne, 'prenom')
prenom.text = 'Jane'

age = ET.SubElement(nouvelle_personne, 'age')
age.text = '35'

# Enregistrement des modifications dans le fichier XML
tree.write('files/fichier.xml')
```

Python

Vérifiez le nouveau contenu du `fichier.xml`

```
<personnes>
    <personne>
        <nom>Doe</nom>
        <prenom>John</prenom>
        <age>45</age>
    </personne>
    <personne>
        <nom>Dalton</nom>
        <prenom>Jack</prenom>
        <age>50</age>
    </personne>
    <personne>
        <nom>Smith</nom>
        <prenom>Jane</prenom>
        <age>35</age>
    </personne>
</personnes>
```

Python

Considérons le fichier `fichier.csv` suivant

```
Nom;Prénom;Âge
Ford;James;30
Maggio;Sophie;45
Linus;Benjamin;35
```

Python

Pour lire le contenu de `fichier.xml`

```
import csv

# Ouverture du fichier en mode lecture
with open('fichier.csv', mode='r', newline='', encoding='utf-8') as fichier:
    contenu = csv.reader(fichier)

    # Lecture du contenu ligne par ligne
    for ligne in contenu:
        print(ligne)
```

Python

Pour écrire dans `fichier.csv`

```
import csv

# Données à écrire dans le fichier CSV
donnees = [
    ['Cooper', 'Alan', 25],
    ['Jhonsson', 'Peter', 35]
]

# Chemin vers le fichier CSV
fichier_csv = 'fichier.csv'

# Ouverture du fichier en mode écriture
with open(fichier_csv, mode='a', newline='', encoding='utf-8') as fichier:
    # le delimiter par défaut est ,
    writer = csv.writer(fichier, delimiter=';')

    # écriture des données ligne par ligne
    writer.writerows(donnees)
```

Python

Vérifiez le nouveau contenu du `fichier.xml`

```
Nom;Prénom;Âge
Ford;James;30
Maggio;Sophie;45
Linus;Benjamin;35
Cooper;Alan;25
Jhonsson;Perter;35
```