

FORMATION PYTHON

GESTION DES FICHIERS EN PYTHON

Mustapha HAIN

GESTION DES FICHIERS EN PYTHON

En Python, la gestion de fichiers comprend les trois étapes suivantes:

- Ouvrir un fichier



- Traiter un fichier, (opérations de lecture ou d'écriture.)



- Fermer un fichier.



GESTION DES FICHIERS EN PYTHON

Ouvrir un fichier

Syntaxe :

```
fileobject = open(filename, mode)
```

Mode Description

'r' Ouvrez un fichier en lecture. (par défaut)

'w' Ouvrez un fichier pour l'écriture. Dans ce mode, si le fichier spécifié n'existe pas, il sera créé. Si le fichier existe, alors ses données sont détruites.

'x' Ouvrez un fichier pour une création exclusive. Si le fichier existe déjà, l'opération échoue.

'a' Ouvrez un fichier en mode ajout. Si le fichier n'existe pas, ce mode le créera. Si le fichier existe déjà, les nouvelles données seront ajoutées à la fin du fichier.

'+' Ouvrir un fichier pour la mise à jour (lecture et écriture)

GESTION DES FICHIERS EN PYTHON

Ouvrir un fichier?

Exemple 1 :

```
# mode lecture
```

```
f1 = open("etudiants.txt", "r")
```

```
# ouvrir un fichier binaire en mode écriture.
```

```
f2 = open("photo.jpg", "w")
```

GESTION DES FICHIERS EN PYTHON**Fermer un fichier**

Syntaxe :
f.close()

try:

```
f = open("etudiants.txt", "r", encoding = 'utf-8')
```

```
# traitements
```

finally:

```
f.close()
```

5

GESTION DES FICHIERS EN PYTHON**Ouvrir un fichier****Méthode 1**

```
file=open("f.txt","w+")  
file.write("Rami\n")  
file.write("Sara\n")  
file.write("Imad\n")  
file.close()
```

Méthode 2

```
with open("v.txt","a+") as file2:  
    file2.write("Rami\n")  
    file2.write("Sara\n")  
    file2.write("Imad\n")  
    file2.close()
```

6

GESTION DES FICHIERS EN PYTHON**Fichier et Listes**

```
liste=["a","b","c","d","e"]  
with open("w.txt","a") as file3:  
    for x in liste :  
        file3.write(x+"\n")  
    file3.close()  
#####  
with open("amis.txt","r+") as file4:  
    print(file4.readlines())  
    file4.close()
```

7


GESTION DES FICHIERS EN PYTHON**Déplacement des fichiers**

```
import os  
import shutil  
source="v.txt"  
target="data/v.txt"  
shutil.copy(source,target)  
os.remove(source)
```

8

GESTION DES FICHIERS EN PYTHON

fichiers csv





with open('data.csv', 'r') as file:

csv_reader = csv.reader(file)

for line in csv_reader:

print(line)





```

1 import csv
2
3 with open('data.csv', 'r') as file:
4     csv_reader = csv.reader(file)
5     for line in csv_reader:
6         print(line)

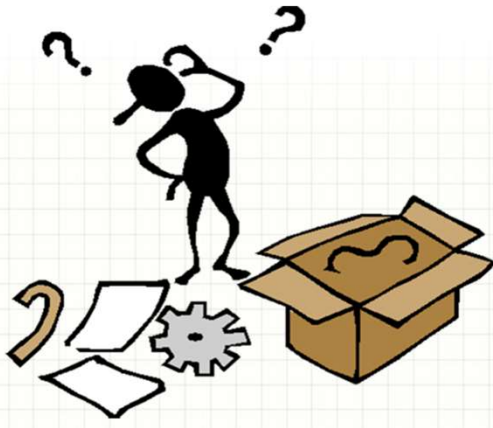
```

```

C:\Users\Admin\AppData\Local\Programs\Python\Python37-32\python.exe C:/Users/Admin/Pychar
['Passenger', 'Id', 'Survived', 'Pclass', 'Name', 'Sex', 'Age']
['1', '0', '3 Braund', ' Mr. Owen Harris ', 'male', ' 22']
['2', '1', '1 Cumings', ' Mrs. John Bradley (Florence Briggs Thayer)', ' female', '38']
['3', '1', '3 Heikkinen', ' Miss. Laina ', 'female', ' 26']
['4', '1', '1 Futrelle', ' Mrs. Jacques Heath (Lily May Peel)', 'female', '35']

```

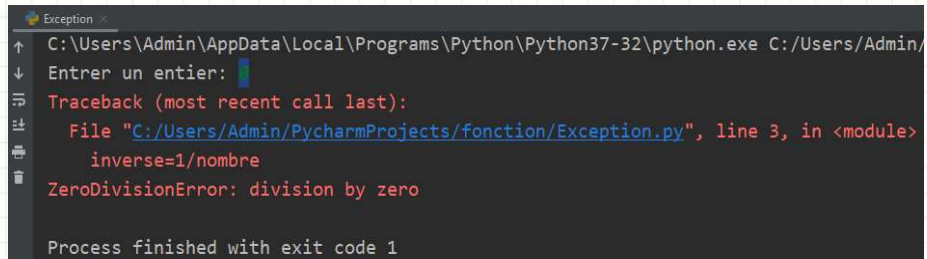
Atelier 5: GESTION DES FICHIERS EN PYTHON



Travaillons ensemble- Activité 1

LA GESTION DES EXCEPTIONS EN PYTHON

```
a=input("Entrer un entier: ")
nombre=int(a)
inverse=1/nombre
print("l'inverse de ",nombre," est ",inverse)
print("Next ...")
```



The screenshot shows a terminal window titled 'Exception' with the following content:

```
C:\Users\Admin\AppData\Local\Programs\Python\Python37-32\python.exe C:/Users/Admin/
Entrer un entier: 
Traceback (most recent call last):
  File "C:/Users/Admin/PycharmProjects/fonction/Exception.py", line 3, in <module>
    inverse=1/nombre
ZeroDivisionError: division by zero

Process finished with exit code 1
```

La gestion des exceptions permet à un programme de faire face aux erreurs d'exécution et continuer son exécution normale.

11

LA GESTION DES EXCEPTIONS EN PYTHON

Cela peut être fait en utilisant l'instruction « try ». Cette instruction suit la syntaxe suivante:

```
try:
    <corps>
except <TypeException>:
    <traitement>
finally :
    # Le bloc finally est toujours exécuté.
```

```
try:
    a=input("Entrer un entier: ")
    nombre=int(a)
    inverse=1/nombre
    print("l'inverse de ",nombre," est ",inverse)
except ZeroDivisionError:
    print("Erreur: Division par zéro!")
print("Next ...")
```

12

LA GESTION DES EXCEPTIONS EN PYTHON

Cela peut être fait en utilisant l'instruction « try ». Cette instruction suit la syntaxe suivante:

```
try:
    <corps>
except <TypeException>:
    <traitement>
finally :
    # Le bloc finally est toujours exécuté.
```

```
try:
    a=input("Entrer un entier: ")
    nombre=int(a)
    inverse=1/nombre
    print("l'inverse de %d est %f " % (nombre,inverse))
except ZeroDivisionError:
    print("Erreur: Division par zéro!")
except ValueError:
    print("Erreur: Valeur non-numérique")
```

13

LA GESTION DES EXCEPTIONS EN PYTHON

```
for file in ["existing_file.txt", "non_existing_file.txt"]:
    try:
        print("Trying to open : **", file, "**")
        f = open(file, 'r')
        print(" le fichier a été trouvé")
        print(" nombre de caractères :",len(f.read()))
    except FileNotFoundError:
        print(" le fichier n'a pas été trouvé")
    finally:
        f.close()
        print(" Fin de l'opération")
```

14

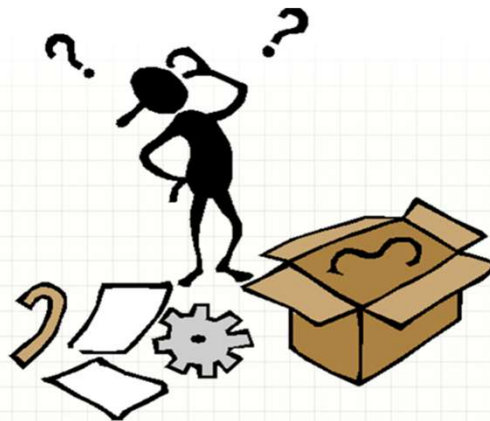
LA GESTION DES EXCEPTIONS EN PYTHON

Liste des erreurs d'exceptions:

- **IOError**: si le fichier ne peut pas être ouvert
- **KeyboardInterrupt**: lorsqu'une touche non requise est enfoncée par l'utilisateur
- **ValueError**: lorsque la fonction intégrée reçoit un argument incorrect
- **EOFError**: si End-Of-File est touché sans lire aucune donnée
- **ImportError**: s'il est impossible de trouver le module
- **MemoryError** : Cette erreur est générée lorsqu'une opération manque de mémoire.

15

Atelier 5: LA GESTION DES EXCEPTIONS EN PYTHON



Travaillons ensemble- Activité