



# BÀI 08

## File Handling

### Đọc và Ghi tập tin

## Tóm Tắt Nội Dung

---

Trong bài học này chúng ta sẽ đi tìm hiểu lần lượt các nội dung

- 1 Tổng quan về xử lí file tập tin
- 2 Phương thức đọc file
- 3 Phương thức ghi và tạo mới file
- 4 Phương thức xóa file, folder

## 8.1 Tổng quan về xử lý file tập tin



### Khái niệm file handling (xử lý tệp tin)

- ◆ Trong Python là quá trình làm việc với các tệp tin trên hệ thống máy tính. Điều này bao gồm đọc tệp, ghi vào tệp và thực hiện các thao tác khác liên quan đến tệp. Python cung cấp một số phương thức và module hỗ trợ làm việc với tệp rất dễ dàng.
- ◆ Python có thể làm việc với nhiều loại tệp khác nhau, bao gồm:
  1. **Tệp văn bản** (Text files): Đây là loại tệp thông thường chứa dữ liệu dưới dạng văn bản
  2. **Tệp nhị phân** (Binary files): Đây là các tệp không chứa dữ liệu văn bản, mà thường là dữ liệu nhị phân, hình ảnh, âm thanh, hoặc các loại dữ liệu khác không thuộc kiểu văn bản
  3. **Tệp CSV** (CSV files): CSV (Comma Separated Values) là một loại tệp văn bản đặc biệt, được sử dụng để lưu trữ dữ liệu dưới dạng bảng, trong đó các giá trị cách nhau bởi dấu phẩy

## 8.1 Tổng quan về xử lý file tập tin



### Khái niệm file handling (xử lý tệp tin)

4. **Tệp JSON** (JSON files): JSON (JavaScript Object Notation) là một định dạng dữ liệu phổ biến dựa trên văn bản, thường được sử dụng để truyền và lưu trữ dữ liệu dạng cấu trúc
5. **Tệp XML** (XML files): XML (eXtensible Markup Language) là một định dạng dữ liệu được sử dụng để lưu trữ thông tin dưới dạng cây cấu trúc
6. **Tệp SQLite** (SQLite files): SQLite là một hệ quản lý cơ sở dữ liệu nhẹ và phổ biến được tích hợp sẵn trong Python
7. **Các loại tệp khác**: Ngoài các loại tệp nêu trên, Python cũng có thể làm việc với các loại tệp tùy chỉnh hoặc định dạng tệp đặc biệt khác thông qua việc sử dụng các thư viện phù hợp hoặc tự viết mã xử lý tùy chỉnh

## 8.1 Tổng quan về xử lý file tập tin

---



### Khái niệm file handling (xử lý tệp tin)

- ◆ Trong phạm vi bài học này, chúng ta chỉ tìm hiểu kiểu tệp tin văn bản thuần.
- ◆ Tệp văn bản thường có phần mở rộng **.txt** hay **.dat**

## 8.1 Tổng quan về xử lí file tập tin



### Phương thức **open()** - Mở tệp tin

- ◆ Việc đọc hay ghi dữ liệu từ tệp tin sẽ thông qua một đầu đọc (file head). Khi đọc hoặc ghi dữ liệu sẽ đọc hay ghi dữ liệu trực tiếp từ vị trí của đầu đọc
- ◆ Khi đọc xong hay ghi xong đầu đọc sẽ dịch chuyển về cuối tệp tin đúng một khoảng bằng lượng thông tin (tính theo byte) đã đọc (ghi) được.



- ◆ Lệnh **open("tên\_file.txt", "r")** sẽ mở tệp sẵn sàng đọc. Đầu đọc sẽ chuyển về vị trí gốc là 0.
- ◆ Lệnh **open("tên\_file.txt", "w")** sẽ mở tệp, xóa tất cả nội dung đang có và đưa đầu đọc về vị trí đầu tệp. Nếu tệp chưa tồn tại thì tạo mới

## 8.2 Phương thức đọc tập tin



### Phương thức **open()** - Mở tập tin

Để làm việc với tệp, trước tiên chúng ta cần mở tệp. Sử dụng hàm **open()** để mở một tệp trong Python. Hàm này trả về một đối tượng tệp tin, cho phép bạn thực hiện các thao tác đọc/ghi trên tệp.

```
file = open("ten_file.txt", ["mode"])
```

- ◆ **ten\_file.txt**: là tên tệp tin muốn mở
- ◆ **mode**: tùy chọn, là chế độ mở tệp với các tùy chọn sau:
  - r**: là Read (đọc, mở), giá trị mặc định
  - w**: là Write (ghi đè), nếu tệp chưa tồn tại thì tạo mới
  - a**: là Append (ghi nối vào cuối tệp), nếu tệp chưa tồn tại thì tạo mới
  - x**: là Create (tạo mới tệp), trả về lỗi nếu tệp đã tồn tại
  - t**: là text mode, giá trị mặc định

## 8.2 Phương thức đọc tập tin



### Phương thức **open()** - Mở tệp tin

Ví dụ: để mở file `demo_file.txt` bạn chỉ cần code như sau:

```
file = open("demo_file.txt")
```

Dòng code trên sẽ tương đương với, vì **rt** là mặc định

```
file = open("demo_file.txt", "rt")
```



## 8.2 Phương thức đọc tập tin



### Phương thức **read()** – Đọc toàn bộ nội dung tập tin

Ví dụ bạn có một file `demo_file.txt` với nội dung:

```
Hello! Welcome to demo_file.txt  
This file is for testing purposes.  
Good Luck!
```

Để lấy được nội dung này và hiển thị ra cho người xem bạn code như sau

```
file = open("demo_file.txt", "r")  
content = f. f.read() #Lấy tất cả nội dung  
print(content)  
print(f.read(5)) #Chỉ lấy 5 kí tự đầu tiên
```

Đầu tiên mở file ra, sau đó dùng phương thức **read()** để đọc nội dung tập tin

## 8.2 Phương thức đọc tập tin



### Phương thức **read()** – Đọc toàn bộ nội dung tập tin

Chúng ta cùng tìm hiểu cách đọc dữ liệu trong Python qua 2 lần gọi `read()`

Đầu tệp

Vị trí đầu đọc đang đọc tới



Cuối tệp

Trong lần gọi `read()` đầu tiên, toàn bộ nội dung được lưu vào biến nhớ `content` (`content` là một `str`) , khi đó đầu đọc nằm ở vị trí cuối tệp

Nếu gọi `read()` lần thứ 2: `content = f.read()` thì `content` bây giờ là rỗng.  
Lí do là lệnh `read()` luôn đọc dữ liệu từ vị trí đầu đọc trở đi

## 8.2 Phương thức đọc tập tin



Làm sao biết đầu đọc đang ở vị trí nào ?

Hàm `tell()` sẽ cho chúng ta biết vị trí của đầu đọc tính từ vị trí đầu tệp tin.

```
f = open("demo_file.txt", "r")  
print(f.tell()) #Mở tệp đầu đọc ở vị trí số 0 (đầu tệp)  
  
content = f.read()  
print(f.tell()) #Đọc xong đầu đọc ở vị trí số 110 (cuối tệp)  
  
f.close() #đóng tệp, kết thúc phiên làm việc với file hiện tại
```

Phương thức **`close()`** được gọi sau khi bạn mở file và ghi xong rất quan trọng, nó giúp: Giải phóng tài nguyên, Lưu trữ dữ liệu, Tránh lỗi khi làm việc sau khi tệp đã được đóng.

## 8.2 Phương thức đọc tập tin



### Phương thức readline()

Phương thức này được sử dụng để đọc một dòng văn bản từ tệp. Mỗi lần gọi phương thức readline(), nó sẽ đọc một dòng ký tự đầu tiên của tệp văn bản và di chuyển con trỏ tệp đến dòng kế tiếp. Khi không còn dòng nào để đọc, readline() trả về một chuỗi rỗng

```
f = open("demo_file.txt", "r")  
line1 = f.readline()  
print(line1) # In ra dòng đầu tiên của tệp  
  
line2 = f.readline()  
print(line2) # In ra dòng thứ 2 của tệp
```

## 8.2 Phương thức đọc tập tin



### Phương thức readlines()

Phương thức này được sử dụng để đọc tất cả các dòng văn bản từ tệp và trả về một danh sách (kiểu list) các chuỗi, trong đó mỗi phần tử trong danh sách là một dòng văn bản từ tệp

```
f = open("demo_file.txt", "r")
lines = f.readlines()
print(type(lines)) #in ra được: list
print(lines) # In ra toàn bộ nội dung của tệp ở dạng list
```

Khi bạn sử dụng readlines(), toàn bộ nội dung của tệp sẽ được đọc và lưu trữ trong bộ nhớ. Nếu tệp quá lớn, điều này có thể ảnh hưởng đến hiệu năng và tiêu tốn nhiều tài nguyên.

## 8.2 Phương thức ghi và tạo mới file



### Phương thức **write()** - Ghi tệp tin đã tồn tại

Để ghi nội dung vào một tệp tin đã tồn tại trước hết bạn phải mở nó với phương thức `open()` và chọn 1 trong 2 chế độ (mode):

**a:** là Append (ghi nối vào cuối tệp), nếu tệp chưa tồn tại thì tạo mới

**w:** là Write (ghi đè), nếu tệp chưa tồn tại thì tạo mới

Ví dụ mở file `demo_file.txt` và ghi tiếp nội dung vào

```
f = open("demo_file.txt", "a")  
f.write("Now the file has more content!")  
f.close()
```

## 8.2 Phương thức ghi và tạo mới file



### Phương thức **write()** - Ghi tệp tin đã tồn tại

Ví dụ mở file `demo_file.txt` và **ghi đè** nội dung đã có

```
file = open("demo_file.txt", "w")  
f.write("Now the file has more content!")  
f.close()
```

Bạn kiểm tra nội dung đã được thay đổi chưa với cách code như sau:

```
file = open("demo_file.txt", "r")  
print(f.read())
```

Đầu tiên mở file ra, sau đó dùng phương thức `read()` để đọc nội dung tệp tin

## 8.2 Phương thức ghi và tạo mới file



### Phương thức **write()** - Ghi tệp tin đã tồn tại

Ghi nội xuống dòng

```
f = open("demo_file.txt", "w")
msg1 = 'Đây là dòng 1 \n'
msg2 = 'Đây là dòng 2 \n'
msg3 = 'Đây là dòng 3 \n'
f.write(msg1)
f.write(msg2)
print(msg3, file = fileout) #Ghi nội dung với lệnh print()
f.close()
```

Để ghi nhiều dòng và muốn xuống dòng bạn có thể sử dụng kí tự \n vào cuối mỗi dòng như ví dụ trên

Bạn cũng có thể sử dụng lệnh `print()` với tham số thứ 2 (file) như trên để ghi nội dung



## 8.2 Phương thức ghi và tạo mới file



### Phương thức **writelines()** - Ghi tệp tin đã tồn tại

Đây là một tùy chọn khác cho việc ghi file, khi bạn có nội dung cần ghi ở dạng list các chuỗi văn bản

```
f = open("demo_file.txt", "w")  
mylist = ["apple", "banana", "cherry"]  
f.writelines(mylist)  
f.close()
```

Với cách này bạn không thể sử dụng kí tự \n để xuống dòng

## 8.2 Phương thức ghi và tạo mới file



### Tạo mới tệp tin

Để tạo mới một tệp tin trong Python, bạn sử dụng phương thức `open()` với một trong 3 tùy chọn:

**w**: là Write (ghi đè), nếu tệp chưa tồn tại thì tạo mới

**a**: là Append (ghi nối vào cuối tệp), nếu tệp chưa tồn tại thì tạo mới

**x**: là Create (tạo mới tệp tin), trả về lỗi nếu tệp đã tồn tại

Tạo mới file `myfile.txt` (tạo file với nội dung trống)

```
file = open("myfile.txt", "x")
```

Tạo mới file `myfile.txt` nếu nó chưa tồn tại, có thể thêm nội dung sau khi tạo file

```
file = open("myfile.txt", "w")
```

## 10.3 Phương thức xóa tệp tin, folder



### Xóa tệp tin

Để xóa được một tệp tin (file) bạn cần đến OS module, sau đó dùng phương thức `os.remove()` như sau:

```
import os
os.remove("demofile.txt")
```

Để tránh lỗi xảy ra khi xóa file bạn nên kiểm tra sự tồn tại của file trước

```
import os
if os.path.exists("demofile.txt"): #kiểm tra xem tồn tại không
    os.remove("demofile.txt") #xóa nếu tồn tại
else:
    print("File không tồn tại")
```

## 10.4 Phương thức xóa tệp tin, folder



### Tạo folder

Tạo một folder rỗng trong Python với đường dẫn chỉ định

```
import os  
os.mkdir("path/to/new_directory")
```



### Xóa folder

Để xóa được một folder bạn cần đến OS module, sau đó dùng phương thức `os.rmdir()` như sau:

```
import os  
os.rmdir("myfolder")
```

Và lưu ý là nó chỉ xóa được khi trong folder đó rỗng

## 10.5 Phương thức xóa tệp tin, folder



### Đổi tên file

Đổi tên một file trong Python với đường dẫn chỉ định

```
import os  
os.rename("path/to/old_filename.txt", "path/to/new_filename.txt")
```



### Đổi tên folder

Dùng chung hàm với đổi file

```
import os  
os.rename("path/to/old_folder_name", "path/to/new_folder_name")
```

## 10.6 Phương thức xóa tệp tin, folder

---



### Nguồn tham chiếu

- ◆ Xử lý file trong Python – w3School

[https://www.w3schools.com/python/python\\_file\\_handling.asp](https://www.w3schools.com/python/python_file_handling.asp)

- ◆ Các phương thức xử lý trong module os của Python

[https://www.w3schools.com/python/module\\_os.asp](https://www.w3schools.com/python/module_os.asp)

# Tổng kết lại bài

---

- 1 Nắm được tổng quan về xử lý file tập tin
- 2 Nắm được phương thức đọc file
- 3 Nắm được phương thức ghi và tạo mới file
- 4 Nắm được phương thức xóa file, folder