

# Bài 19: XỬ LÝ FILE TRONG PYTHON

Xem bài học trên website để ủng hộ Kteam: [Xử lý file trong Python](#)

Mọi vấn đề về lỗi website làm ảnh hưởng đến bạn hoặc thắc mắc, mong muốn khóa học mới, nhằm hỗ trợ cải thiện Website. Các bạn vui lòng phản hồi đến Fanpage [How Kteam](#) nhé!

## Dẫn nhập

Trong bài trước, Kteam đã giới thiệu đến bạn các phương thức của [KIỂU DỮ LIỆU DICT](#) trong Python

Ở bài này Kteam sẽ giới thiệu với các bạn **CÁCH XỬ LÝ FILE** trong Python. Một trong những điều thiết yếu mà bất cứ ngôn ngữ lập trình nào bạn cũng đều phải tìm hiểu.

## Nội dung

Để đọc hiểu bài này tốt nhất bạn cần:

- Cài đặt sẵn [MÔI TRƯỜNG PHÁT TRIỂN CỦA PYTHON](#).
- Xem qua bài [CÁCH CHẠY CHƯƠNG TRÌNH PYTHON](#).
- Nắm [CÁCH GHI CHÚ](#) và [BIẾN TRONG PYTHON](#).
- [KIỂU DỮ LIỆU LIST](#), [KIỂU DỮ LIỆU TUPLE](#), [KIỂU DỮ LIỆU SET](#) trong Python.

Trong bài này, chúng ta sẽ cùng tìm hiểu những nội dung sau đây

- Khái quát về File trong Python
- Mở File trong Python

- Đóng File trong Python
  - Đọc File trong Python
  - Ghi File trong Python
  - Kiểm soát con trỏ File
  - Câu lệnh with
- 

## Khái quát về File trong Python

File là một thứ rất quen thuộc đối với những người sử dụng máy tính. Bạn thao tác, tạo lập file hằng ngày. Nó có thể là một bức hình, một văn bản tài liệu, một file thực thi và nhiều nhiều thứ khác nữa.

Trong Python, file có 2 loại:

### Text File

- Được cấu trúc như một dãy các dòng, mỗi dòng bao gồm một dãy các kí tự và một dòng tối thiểu là một kí tự dù cho dòng đó là dòng trống.
- Các dòng trong text file được ngăn cách bởi một kí tự newline và mặc định trong Python chính là kí tự [escape sequence newline](#) `\n`.

### Binary File

- Các file này chỉ có thể được xử lí bởi một ứng dụng biết và có thể hiểu được cấu trúc của file này.
  - Và chúng ta ở đây với mức độ cơ bản chỉ xử lí text file.
- 

## Mở File trong Python

Khỏi phải bàn, muốn thao tác với file, ta phải mở file. Mà muốn mở file, ta cũng cần phải có file.

Ở đây, Kteam sẽ tạo một file, và sau đó mở **CMD** ở ngay trong thư mục chứa file đó để không gặp nhiều khó khăn trong việc xử lí đường dẫn (Việc xử lí đường dẫn, Kteam sẽ giới thiệu cách xử lí bằng thư viện `os` trong tương lai).

Tên file sẽ là: `kteam.txt`

### Nội dung file:

```
How Kteam  
Free Education  
  
Share to better  
  
print('hello world!')
```

```
1 How Kteam  
2 Free Education  
3  
4 Share to better  
5  
6 print('hello world!')  
7
```

## Hàm open

Được rồi, bây giờ chúng ta sẽ mở file bằng cách sử dụng hàm open

### Cú pháp:

```
open(file, mode='r', buffering=-1, encoding=None, errors=None,  
      newline=None, closefd=True, opener=None)
```

**Công dụng:** Ở mức độ cơ bản, chúng ta sẽ chỉ quan tâm đến 2 parameter: **file** và **mode**.

Nếu các bạn muốn tìm hiểu rõ hơn về các parameter khác. Hãy dùng lệnh:

```
>>> help(open)
```


Ta sẽ bắt đầu bằng cách thử mở một file.

**Lưu ý:** Kteam xin được khuyến khích các bạn không sử dụng **interactive prompt** ở bài này.

```
>>> file_object = open('kteam.txt')
>>> file_object
<_io.TextIOWrapper name='kteam.txt' mode='r' encoding='cp1258'>
>>> type(file_object) # không cần quan tâm lắm
<class '_io.TextIOWrapper'>
```

**Lưu ý:** hàm open trả về một **file object**. Đây cũng là một **iterable**.

Tiếp đến là các mode mở file. Và cũng với mức độ cơ bản, Kteam sẽ cung cấp một số mode cơ bản liên quan đến text file.

MODE	USAGE
r	Mở để đọc. Đây là mode mặc định
r+	Mở để đọc và ghi
w	Mở để ghi. Trước đó, nó sẽ xóa hết nội dung của file hiện có. Nếu file không tồn tại, sẽ tạo ra một file với tên là tên file chúng ta truyền vào
w+	Mở để ghi và đọc. Trước đó cũng đã xóa hết nội dung của file hiện có. Nếu file không tồn tại, sẽ tạo ra một file với tên là tên file chúng ta truyền vào
a	Mở để ghi. Nếu file không tồn tại, sẽ tạo ra một file với tên là tên file chúng ta truyền vào
 a+	Mở để ghi và đọc Nếu file không tồn tại, sẽ tạo ra một file với tên là tên file chúng ta truyền vào

## Đóng File trong Python

Đây là việc chúng ta nên làm sau khi thao tác xong với file. Đó là đóng file.

Cú pháp:

```
<File>.close()
```

Tại sao chúng ta nên đóng file sau khi hoàn tất công việc với file?

- Giới hạn hệ điều hành. Chẳng hạn một hệ điều hành chỉ cho mở một số file nhất định cùng lúc thì nếu quên đóng file sẽ gây hao tổn. Đặc biệt là các file với dung lượng bự.
- Khi một file được mở, hệ điều hành sẽ khóa file đó lại, không cho các chương trình khác có thể xử lý trên file đó nữa nhằm đảm bảo tính nhất quán của dữ liệu.

Do đó hãy close file khi xong việc!

Dẫu vậy, nếu chương trình kết thúc. Tất cả các file đang mở cũng sẽ được đóng lại. Tuy nhiên việc đóng file vẫn là trách nhiệm nằm ở chúng ta.

```
>>> fobj = open('kteam.txt')
>>> fobj
<_io.TextIOWrapper name='kteam.txt' mode='r' encoding='cp1258'>
>>> fobj.close()
>>> fobj # sau khi đóng file, các phương thức xử lý sẽ không thể sử dụng được
<_io.TextIOWrapper name='kteam.txt' mode='r' encoding='cp1258'>
```

## Đọc File trong Python

Ta có một số phương thức có thể lấy được nội dung của file

### Phương thức read

Cú pháp:

```
<File>.read(size=-1)
```

**Công dụng:** Nếu `size` bị bỏ trống hoặc là một số âm. Nó sẽ đọc hết nội dung của file đồng thời đưa con trỏ file tới cuối file. Nếu không nó sẽ đọc tới `n` kí tự (với `n = size`) hoặc cho tới khi nội dung của file đã đọc xong.

- Sau khi đọc được nội dung, nó sẽ trả về dưới một dạng chuỗi.

- Nếu không đọc được gì, phương thức sẽ trả về một chuỗi có độ dài bằng 0

**Ví dụ:**

```
>>> fobj = open('kteam.txt')
>>> data = fobj.read()
>>> data
"How Kteam\nFree Education\n\nShare to better\n\nprint('hello world!')\n"
>>> print(data)
How Kteam
Free Education

Share to better

print('hello world!')

>>> fobj.read() # con trỏ file ở vị trí cuối cùng, bạn không thể đọc được gì nữa
"
>>> fobj.close() # nhớ đóng file
```

Dưới đây là một ví dụ về đọc từng số kí tự một

```
>>> fobj = open('kteam.txt')
>>> fobj.read(2)
'Ho'
>>> fobj.read(10)
'w Kteam\nFr'
>>> fobj.read(20)
'ee Education\n\nShare '
>>> fobj.read()
"to better\n\nprint('hello world!')\n"
>>> fobj.close()
```

## Phương thức readline

Cú pháp:

```
<File>.readline(size=-1)
```

**Công dụng:** Với parameter `size` thì hoàn toàn tương tự như phương thức `read`.

- Khác biệt ở chỗ, phương thức `readline` chỉ đọc một dòng có nghĩa là đọc tới khi nào gặp `newline` hoặc hết file thì ngừng.
- Con trỏ file cũng sẽ đi từ dòng này qua dòng khác.
- Kết quả đọc được trả về dưới dạng một chuỗi.
- Nếu không đọc được gì, phương thức sẽ trả về một chuỗi có độ dài bằng 0.

**Ví dụ:**

```
>>> fobj = open('kteam.txt')
>>> fobj.readline()
'How Kteam\n'
>>> fobj.readline(10)
'Free Educa'
>>> fobj.readline()
'tion\n'
>>> fobj.readline()
'\n'
>>> fobj.readline()
'Share to better\n'
>>> fobj.close()
```



## Phương thức readlines

Cú pháp:

```
<File>.readlines(hint=-1)
```

Ở mức độ cơ bản, ta không phải quan tâm đến parameter `hint`.

**Công dụng:** Phương thức này sẽ đọc toàn bộ file, sau đó cho chúng vào một list. Với các phần tử trong list là mỗi dòng của file.

- Con trỏ file sẽ được đưa tới cuối file. Khi đó, nếu bạn tiếp tục dùng `readlines`. Bạn sẽ nhận được một list rỗng.

**Ví dụ:**

```
>>> fobj = open('kteam.txt')
>>> list_content = fobj.readlines()
>>> list_content
['How Kteam\n', 'Free Education\n', '\n', 'Share to better\n', '\n', "print('hello\n\nworld!')\n"]
>>> list_content[2]
'\n'
>>> list_content[-1]
"print('hello world!')\n"
>>> fobj.close()
```

## Đọc file bằng constructor nhận iterable

Như đã nói, file object nhận được từ hàm `open` cũng là một **iterable**.

Thế nên, ta có thể sử dụng constructor list

```
>>> fobj = open('kteam.txt')
>>> list_content = list(fobj)
```

```
>>> list_content  
['How Kteam\n', 'Free Education\n', '\n', 'Share to better\n', '\n', "print('hello  
world!')\n"]  
>>> fobj.close()
```

Và cũng có thể là Tuple.

```
>>> fobj = open('kteam.txt')  
>>> tup_content = tuple(fobj)  
>>> tup_content  
('How Kteam\n', 'Free Education\n', '\n', 'Share to better\n', '\n', "print('hello  
world!')\n")  
>>> fobj.close()
```

Các constructor này cũng sẽ đưa con trỏ file xuống cuối file.

---

## Ghi File trong Python

Chúng ta có sự giúp đỡ của phương thức write để ghi nội dung vào file.

Và chúng ta cũng không cần phải tạo file. Vì các mode ghi sẽ giúp chúng ta tạo file.

### Phương thức write

Cú pháp:

```
<File>.write(text)
```

**Công dụng:** Phương thức này sẽ trả về số kí tự mà chúng ta ghi vào.

**Ví dụ:**

```
>>> fobj = open('kteam_2.txt', 'w')
>>> fobj.write('The first line\n') # thêm \n để kết thúc 1 dòng
15
>>> fobj.write('And last line too')
17
>>> fobj.close()
```

Mỗi lần sử dụng write. Con trỏ file sẽ được đặt ngay sau kí tự cuối cùng được ghi. Hãy lưu ý điều này, nó rất quan trọng đấy. Đặc biệt là khi bạn sử dụng các mode vừa đọc vừa ghi.

Nhưng, bạn sẽ gặp vấn đề như thế này khi sử dụng mode w. Ta hãy mở lại file khi này ta mới ghi một vài dòng vào nhé.

```
>>> fobj = open('kteam_2.txt')
>>> fobj.read()
'The first line\nAnd last line too'
>>> fobj.close()
>>> fobj = open('kteam_2.txt', 'w')
>>> fobj.write('\nnone more line')
14
>>> fobj.close()
>>> fobj = open('kteam_2.txt')
>>> fobj.read()
'\nnone more line'
>>> fobj.close()
```

Đó là nội dung file ban đầu của bạn sẽ bị mất đi. Đó là lí do chúng ta cần mới mode a.

Ta hãy mở lại file ta mới viết thêm một lần nữa.

```
>>> fobj = open('kteam_2.txt', 'a')
>>> fobj.write('\nthe second line')
16
>>> fobj.close()
>>> fobj = open('kteam_2.txt')
>>> fobj.read()
'\nnone more line\nthe second line'
>>> fobj.close()
```

## Kiểm soát con trỏ file

Bạn có thể thấy, con trỏ file rất quan trọng, nó dẫn đường cho việc đọc file, viết file. Và bạn cũng cần phải kiểm soát được nó.

Việc đó, ta sẽ nhờ tới phương thức seek

### Phương thức seek

Cú pháp:

```
<File>.seek(offset, whence=0)
```

Với Python 3.X. Một text file sẽ chỉ được sử dụng `whence = 0`. `whence = 1` hoặc `whence = 2` chỉ sử dụng với binary file.

Với Python 2.X thì bạn không phải quan tâm vấn đề này.

Do đó, ta cũng không cần quan tâm tới parameter `whence`.

**Công dụng:** Phương thức này giúp ta di chuyển con trỏ từ vị trí đầu file qua `offset` kí tự. Và parameter `offset` phải là một số tự nhiên.

- Nhờ phương thức này, ta có thể ghi nội dung từ bất cứ đâu trong file.
- Và từ đó ta có thể đọc lại file sau khi ta đưa con trỏ file xuống cuối file.

**Ví dụ:**

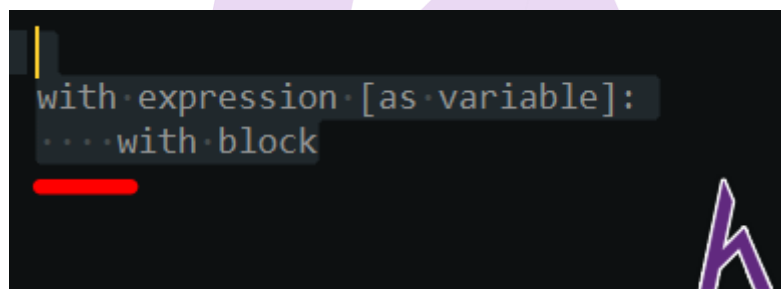
```
>>> fobj = open('kteam.txt')
>>> fobj.read()
"How Kteam\nFree Education\n\nShare to better\n\nprint('hello world!')\n"
>>> fobj.read()
""
>>> fobj.seek(0)
```

```
0
>>> fobj.read()
"How Kteam\nFree Education\n\nShare to better\n\nprint('hello world!')\n"
>>> fobj.seek(10)
10
>>> fobj.read()
"\nFree Education\n\nShare to better\n\nprint('hello world!')\n"
>>> fobj.close()
```

## Câu lệnh with

Cấu trúc cơ bản của câu lệnh with là

```
with expression [as variable]:
    with-block
```



Nhớ rằng **with-block** nằm thụt vào so với dòng **with expression** (theo chuẩn PEP8 là 4 space và là dùng space không dùng tab)

Câu lệnh này liên quan đến phương thức **\_\_enter\_\_** và **\_\_exit\_\_** của đối tượng. Do đó, ở đây Kteam sẽ nói cơ bản khi sử dụng file.

Đặc điểm của câu lệnh with khi sử dụng với file là. Khi kết thúc **with-block**. File sẽ được đóng.

```
>>> with open('kteam.txt') as fobj:
...     data = fobj.read()
...
```

```
>>> data
"How Kteam\nFree Education\n\nShare to better\n\nprint('hello world!')\n"
>>> fobj.read() # không thể đọc file, vì file đã đóng
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: I/O operation on closed file.
```

## Củng cố bài học

### Đáp án bài trước

Bạn có thể tìm thấy câu hỏi của phần này tại CÂU HỎI Củng Cố trong bài [KIỂU DỮ LIỆU DICT TRONG PYTHON – Phần 2](#).

1. Vì hai dict trở cùng vào một nơi. Cách khắc phục là ta dùng phương thức copy để có bản sao dict1.
2. Sẽ có lỗi ở

```
>>> d = {}
>>> d.update(3)
```

### Câu hỏi củng cố

1. Nêu sự khác nhau giữa mode r+ và w+
2. Tèo mở file dưới mode vừa đọc và ghi. Tèo đang thắc mắc là vì sao sau khi ghi xong rồi, mà Tèo vẫn không đọc được gì cả. Hãy giải đáp giúp Tèo.

```
>>> teo_file = open('teo.txt', 'w+')
>>> teo_file.write('Teo dep trai\n')
13
>>> teo_file.read()
```

"

Đáp án của phần này sẽ được trình bày ở bài tiếp theo. Tuy nhiên, Kteam khuyến khích bạn tự trả lời các câu hỏi để củng cố kiến thức cũng như thực hành một cách tốt nhất!

---

## Kết luận

Qua bài viết này, Bạn đã hiểu cơ bản về FILE TRONG PYTHON.

Ở bài viết sau, Kteam sẽ nói về [ITERATION & MỘT SỐ HÀM CƠ BẢN](#) hay được sử dụng.

Cảm ơn bạn đã theo dõi bài viết. Hãy để lại bình luận hoặc góp ý của mình để phát triển bài viết tốt hơn. Đừng quên **"Luyện tập – Thử thách – Không ngại khó"**.