

Đọc ghi file trong Python

NỘI DUNG BÀI VIẾT

- 1. Đọc ghi file trong Python
 - 1.1. Mode truy cập tệp trong Python
 - 1.2. Mở file trong Python
 - 1.3. Đóng file trong Python
 - 1.4. Đọc file trong Python
 - 1.4.1. Đọc toàn bộ file ra string
 - 1.4.2. Đọc từng dòng của file
 - 1.4.3. Đọc toàn bộ file ra list
 - 1.5. Ghi file trong Python
 - 1.6. Ghi dữ liệu vào cuối file
- 2. Đọc ghi JSON từ file
 - 2.1. Cách đọc file JSON
 - 2.2. Ghi dữ liệu từ điển ra file JSON

Đọc ghi file trong Python

Quá trình đọc ghi file trong [Python](#) trải qua 3 bước: **Mở file**, **Đọc hoặc ghi file** và cuối cùng là **đóng file**. Nhưng đầu tiên, chúng ta sẽ xem xét các **mode** khi làm việc với file trong Python nhé.

Mode truy cập tệp trong Python

Khá giống với các ngôn ngữ khác, việc đọc ghi file trong Python cũng có các mode làm việc tương ứng.

Mode	Giải thích
Read Only ('r')	Mode mặc định khi mở tệp. Ở chế độ này, tệp được mở ở chế độ chỉ được phép đọc dữ liệu và con trỏ tệp bắt đầu ở vị trí đầu tệp. Nếu tệp không tồn tại, sẽ gặp ngoại lệ FileNotFoundError.
Read & Write ('r+')	Mở file cho phép cả đọc và ghi. Vị trí con trỏ tệp ở vị trí đầu của tệp. Nếu tệp không tồn tại, sẽ gặp ngoại lệ FileNotFoundError.
Write Only ('w')	Mở file và chỉ cho phép ghi. Vị trí con trỏ tệp ở vị trí đầu của tệp. Nếu tệp không tồn tại, sẽ tự động tạo mới. Nếu tệp đã tồn tại, dữ liệu cũ sẽ bị ghi đè bằng dữ liệu mới.
Write & Read ('w+')	Mở file cho phép cả đọc và ghi. Vị trí con trỏ tệp ở vị trí đầu của tệp. Nếu tệp không tồn tại, sẽ tự động tạo mới. Nếu tệp đã tồn tại, dữ liệu cũ sẽ bị ghi đè bằng dữ liệu mới.
Append Only ('a')	Mở file cho phép ghi. File sẽ được tạo mới nếu chưa tồn tại. Con trỏ tệp sẽ ở cuối file nên sẽ tiếp tục ghi dữ liệu vào cuối nếu ban đầu đã có dữ liệu.
Append & Read ('a+')	Mở file cho phép ghi và đọc. File sẽ được tạo mới nếu chưa tồn tại. Con trỏ tệp sẽ ở cuối file nên sẽ tiếp tục ghi dữ liệu vào cuối nếu ban đầu đã có dữ liệu.

Mở file trong Python

Để mở file trong Python, chúng ta không cần phải import thêm thư viện nào cả, chúng ta sẽ sử dụng hàm `open()` built-in có cú pháp như sau:

```
0
1 file_object = open(r'đường_dẫn_tới_file', 'mode')
2
```

Lưu ý: Ký tự `r` trước đường dẫn tới file giúp bỏ qua các ký tự đặc biệt của string trong Python. Chẳng hạn, nếu không có ký tự `r` này thì `\t` trong đường dẫn `D:\text\myfile.txt` này sẽ bị coi là dấu tab, dẫn tới xảy ra lỗi không mong muốn.

Ví dụ:

```
0
1 # mở file để đọc, mode = 'r'
2 rf = open('data.txt', 'r')
3 # hoặc
4 rf = open('data.txt')
5
6 # mở file để ghi
7 wf = open('data.txt', 'w')
8
9 # mở file để ghi vào cuối
10 wf = open('data.txt', 'a')
11
```

Đóng file trong Python

Sử dụng hàm `close()` để đóng đối tượng tệp khi bạn đã làm việc xong với nó. Hãy luôn nhớ đóng file sau khi làm việc xong nếu không muốn phát sinh các vấn đề không mong muốn.

Ví dụ:

```
0
1 # mở file để đọc, mode = 'r'
2 rf = open('data.txt', 'r')
3 # hoặc
4 rf = open('data.txt')
5 # đóng file
6 rf.close()
7
8 # mở file để ghi
9 wf = open('data.txt', 'w')
10 # đóng file
11 wf.close()
12
13 # mở file để ghi vào cuối
14 wf = open('data.txt', 'a')
15 # đóng file
16 wf.close()
17
```

Một cách linh hoạt hơn (nên dùng) khi làm việc với file mà không cần phải lo khi nào đóng file. Hãy sử dụng từ khóa `with` như sau:

```

0
1 # Python sẽ tự động đóng file cho bạn
2 with open('/home/user/data.txt', 'r') as fp:
3     # doing smt
4
5 # tiếp tục các khối lệnh khác ...
6

```

Đọc file trong Python

Python cung cấp cho bạn 3 cách khác nhau để đọc nội dung từ tập tin. Mình sẽ đi trực tiếp vào ví dụ để các bạn có thể hiểu & áp dụng được ngay

Đọc toàn bộ file ra string

Sử dụng hàm `read()` để đọc toàn bộ nội dung file vào 1 biến string.

```

0
1 with open('data.txt') as rf:
2     content = rf.read()
3     print(content)
4     content = content + "\n\nFrom LTKK with love"
5     print(content)
6

```

Kết quả nhận được (cũng chính là nội dung file `data.txt`):

```

0
1 Kích thước cố định
2 Cần chỉ rõ kích thước trong khi khai báo
3 Kích thước thay đổi trong quá trình thêm/ xóa phần tử
4 Kích thước tối đa phụ thuộc vào bộ nhớ
5 typedef được dùng để định nghĩa một kiểu dữ liệu trong C.
6 malloc là hàm cấp phát bộ nhớ của C. Với C++ chúng ta dùng new
7 sizeof là hàm trả về kích thước của kiểu dữ liệu, dùng làm tham số cho hàm malloc
8 Kích thước cố định
9 Cần chỉ rõ kích thước trong khi khai báo
10 Kích thước thay đổi trong quá trình thêm/ xóa phần tử
11 Kích thước tối đa phụ thuộc vào bộ nhớ
12 typedef được dùng để định nghĩa một kiểu dữ liệu trong C.
13 malloc là hàm cấp phát bộ nhớ của C. Với C++ chúng ta dùng new
14 sizeof là hàm trả về kích thước của kiểu dữ liệu, dùng làm tham số cho hàm malloc
15
16 From LTKK with love
17

```

Đọc từng dòng của file

Sử dụng hàm `readline()` sẽ trả về 1 dòng nội dung của file. Chúng ta có thể duyệt qua từng dòng nội dung của file như sau:

```

0
1 with open('data.txt') as rf:
2     line = rf.readline()
3     index = 1
4     while line:
5         print('Line {}: {}'.format(index, line))
6         index += 1
7         line = rf.readline()
8

```

Kết quả:

```

0
1 Line 1: Kích thước cố định
2
3 Line 2: Cần chỉ rõ kích thước trong khi khai báo
4
5 Line 3: Kích thước thay đổi trong quá trình thêm/ xóa phần tử
6
7 Line 4: Kích thước tối đa phụ thuộc vào bộ nhớ
8
9 Line 5: typedef được dùng để định nghĩa một kiểu dữ liệu trong C.
10
11 Line 6: malloc là hàm cấp phát bộ nhớ của C. Với C++ chúng ta dùng new
12
13 Line 7: sizeof là hàm trả về kích thước của kiểu dữ liệu, dùng làm tham số cho hàm mal
14

```

Hàm `readline()` sẽ đọc 1 dòng, bao gồm cả ký tự newline (`\n`). Nên khi in ra, mỗi dòng có thể 1 dòng trống ở phía dưới như kết quả ở trên.

Đọc toàn bộ file ra list

Vẫn là đọc toàn bộ file nhưng hàm `readlines()` sẽ trả về một list tương ứng mỗi dòng dữ liệu là 1 phần tử của list.

```

0
1 with open('data.txt') as rf:
2     lines = rf.readlines()
3     for idx, line in enumerate(lines):
4         print(idx, line)
5

```

Kết quả:

```

0
1 0 Kích thước cố định
2
3 1 Cần chỉ rõ kích thước trong khi khai báo
4
5 2 Kích thước thay đổi trong quá trình thêm/ xóa phần tử
6
7 3 Kích thước tối đa phụ thuộc vào bộ nhớ
8
9 4 typedef được dùng để định nghĩa một kiểu dữ liệu trong C.
10
11 5 malloc là hàm cấp phát bộ nhớ của C. Với C++ chúng ta dùng new
12
13 6 sizeof là hàm trả về kích thước của kiểu dữ liệu, dùng làm tham số cho hàm malloc
14

```

Lưu ý: Bạn có thể dùng hàm `strip()` để xóa khoảng trắng thừa ở đầu & cuối string. Và ký tự newline ở cuối mỗi string cũng được coi là 1 khoảng trắng thừa.

Ghi file trong Python

Để ghi file trong Python, chúng ta có thể sử dụng hàm `write()` để ghi 1 biến string, hoặc dùng `writelines()` để ghi 1 list các chuỗi string.

```
0
1 texts = [
2     "0 Kích thước cố định",
3     "1 Cần chỉ rõ kích thước trong khi khai báo",
4     "2 Kích thước thay đổi trong quá trình thêm/ xóa phần tử",
5     "3 Kích thước tối đa phụ thuộc vào bộ nhớ",
6     "4 typedef được dùng để định nghĩa một kiểu dữ liệu trong C.",
7     "5 malloc là hàm cấp phát bộ nhớ của C. Với C++ chúng ta dùng new",
8     "6 sizeof là hàm trả về kích thước của kiểu dữ liệu, dùng làm tham số cho hàm malloc"
9 ]
10 with open('data.txt', 'w') as wf:
11     for text in texts:
12         wf.write(text + '\n')
13
```

Hoặc:

```
0
1 texts = [
2     "0 Kích thước cố định",
3     "1 Cần chỉ rõ kích thước trong khi khai báo",
4     "2 Kích thước thay đổi trong quá trình thêm/ xóa phần tử",
5     "3 Kích thước tối đa phụ thuộc vào bộ nhớ",
6     "4 typedef được dùng để định nghĩa một kiểu dữ liệu trong C.",
7     "5 malloc là hàm cấp phát bộ nhớ của C. Với C++ chúng ta dùng new",
8     "6 sizeof là hàm trả về kích thước của kiểu dữ liệu, dùng làm tham số cho hàm malloc"
9 ]
10 with open('data.txt', 'w') as wf:
11     wf.writelines(texts)
12
```

Lưu ý:

Hàm `writelines()` không tự động chèn thêm `\n` vào cuối mỗi dòng. Nên nếu chạy code trên thì các dòng sẽ ghi liền nhau thành 1 dòng duy nhất.

0 Kích thước cố định

1 Cần chỉ rõ kích thước trong khi khai báo

2 Kích thước thay đổi trong quá trình thêm/ xóa phần tử

3 Kích thước tối đa phụ thuộc vào bộ nhớ

4 typedef được dùng để định nghĩa một kiểu dữ liệu trong C.

5 malloc là hàm cấp phát bộ nhớ của C.

Với C++ chúng ta dùng new6 sizeof là hàm trả về kích thước của kiểu dữ liệu, dùng làm tham số cho hàm malloc

Ghi dữ liệu vào cuối file

Chúng ta chỉ cần đổi mode từ **w** sang **a** là được. Xem ví dụ:

```
0
1 texts = [
2     "0 Kích thước cố định",
3     "1 Cần chỉ rõ kích thước trong khi khai báo",
4     "2 Kích thước thay đổi trong quá trình thêm/ xóa phần tử",
5     "3 Kích thước tối đa phụ thuộc vào bộ nhớ",
6     "4 typedef được dùng để định nghĩa một kiểu dữ liệu trong C.",
7     "5 malloc là hàm cấp phát bộ nhớ của C. Với C++ chúng ta dùng new",
8     "6 sizeof là hàm trả về kích thước của kiểu dữ liệu, dùng làm tham số cho hàm malloc"
9 ]
10 with open('data.txt', 'w') as wf:
11     for text in texts:
12         wf.write(text + '\n')
13
14 with open('data.txt') as wr:
15     print("Number of line:", len(wr.readlines()))
16
17 # Ghi file vào cuối
18 with open('data.txt', 'a') as wf:
19     wf.write('=> From LTKK with love')
20
21 with open('data.txt') as wr:
22     texts = wr.readlines()
23     print("Number of line:", len(texts))
24     print(texts)
25
```

Kết quả:

```
0
1 Number of line: 7
2 Number of line: 8
3 ['0 Kích thước cố định\n', '1 Cần chỉ rõ kích thước trong khi khai báo\n', '2 Kích thước
4
```

Đọc ghi JSON từ file

Trong phần này, chúng ta sẽ làm việc với tệp tin JSON sử dụng ngôn ngữ Python. Giả sử chúng ta có file JSON như sau:

```
customer.json
1 [
2   {
3     "id": 1,
4     "email": "isidro_von@hotmail.com",
5     "first": "Torrey",
6     "last": "Veum",
7     "company": "Hilll, Mayert and Wolf",
8     "created_at": "2014-12-25T04:06:27.981Z",
9     "country": "Switzerland"
10  },
11  {
12    "id": 2,
13    "email": "frederique19@gmail.com",
14    "first": "Micah",
15    "last": "Sanford",
16    "company": "Stokes-Reichel",
17    "created_at": "2014-07-03T16:08:17.044Z",
18    "country": "Democratic People's Republic of Korea"
19  }
20 ]
```

Cách đọc file JSON

```
0
1 import json
2
3 with open('customer.json') as wr:
4     customers = json.load(wr)
5     for customer in customers:
6         print(customer, type(customer))
7
```

Kết quả:

```
0
1 {'id': 1, 'email': 'isidro_von@hotmail.com', 'first': 'Torrey', 'last': 'Veum', 'company': 'Hilll, Mayert and Wolf'}
2 {'id': 2, 'email': 'frederique19@gmail.com', 'first': 'Micah', 'last': 'Sanford', 'company': 'Stokes-Reichel'}
3
```

Ghi dữ liệu từ điển ra file JSON

```
0
1 import json
2
3 customers = [
4     {
5         "id": 1,
6         "email": "isidro_von@hotmail.com",
7         "first": "Torrey",
8         "last": "Veum",
9         "company": "Hilll, Mayert and Wolf",
10        "created_at": "2014-12-25T04:06:27.981Z",
11        "country": "Switzerland"
12    },
13    {
14        "id": 2,
15        "email": "frederique19@gmail.com",
16        "first": "Micah",
17        "last": "Sanford",
18        "company": "Stokes-Reichel",
19        "created_at": "2014-07-03T16:08:17.044Z",
20        "country": "Democratic People's Republic of Korea"
21    }
22 ]
23
24 with open('customer.json', 'w') as wr:
25     json.dump(customers, wr)
26
```

Lưu ý:

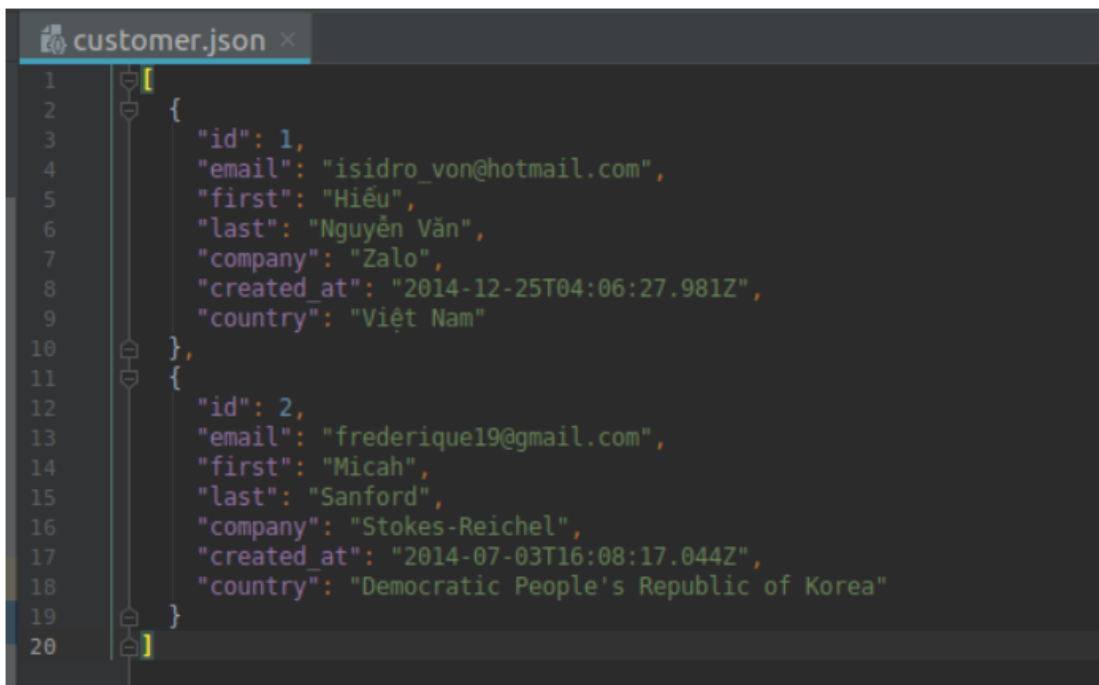
- Với dữ liệu tiếng Việt, hàm `dump()` và `dumps()` sẽ đưa toàn bộ về bảng mã ASCII. Do đó, nếu bạn muốn đọc được file JSON khi mở bằng Editor, thêm option `ensure_ascii = False`.
- Nếu muốn làm đẹp file JSON, hãy thêm option `indent=2` vào hàm `dump()` nhé. Trong đó, 2 số lượng dấu cách (space) dùng để format tệp tin.

```

0
1 import json
2
3 customers = [
4     {
5         "id": 1,
6         "email": "isidro_von@hotmail.com",
7         "first": "Hiếu",
8         "last": "Nguyễn Văn",
9         "company": "Zalo",
10        "created_at": "2014-12-25T04:06:27.981Z",
11        "country": "Việt Nam"
12    },
13    {
14        "id": 2,
15        "email": "frederique19@gmail.com",
16        "first": "Micah",
17        "last": "Sanford",
18        "company": "Stokes-Reichel",
19        "created_at": "2014-07-03T16:08:17.044Z",
20        "country": "Democratic People's Republic of Korea"
21    }
22 ]
23
24 with open('customer.json', 'w') as wr:
25     json.dump(customers, wr, ensure_ascii=False, indent=2)
26

```

Kết quả:



```

1 [
2   {
3     "id": 1,
4     "email": "isidro_von@hotmail.com",
5     "first": "Hiếu",
6     "last": "Nguyễn Văn",
7     "company": "Zalo",
8     "created_at": "2014-12-25T04:06:27.981Z",
9     "country": "Việt Nam"
10  },
11  {
12    "id": 2,
13    "email": "frederique19@gmail.com",
14    "first": "Micah",
15    "last": "Sanford",
16    "company": "Stokes-Reichel",
17    "created_at": "2014-07-03T16:08:17.044Z",
18    "country": "Democratic People's Republic of Korea"
19  }
20 ]

```

Như vậy, bài viết đã trình bày các nội dung cần thiết đến bạn đọc về nội dung đọc ghi file trong Python. Với những kiến thức căn bản này, bạn hoàn toàn có thể sử dụng đáp ứng yêu cầu trong công việc rồi. Chúc các bạn học tập tốt!

(Nguồn: Nguyễn Văn Hiếu - Lập trình không khó)