TIN HỌC CƠ SỞ

Phần 2: Ngôn ngữ lập trình Python

CHƯƠNG 13.

XỬ LÝ TẬP TIN TRONG PYTHON

Tin cơ sở (LT): 010100229802 - DHKL16A1HN, - DHKL16A2HN

GV. Cao Diệp Thắng

Mục tiêu chương

Nắm vững:

- ❑ Khái niệm, cách tạo, sử dụng module
- ❑ Khái niệm namespace và scope
- ❑ Package của python
- ❑ Thư viện chuẩn,

Nội dung

1. Khái niệm module
2. Xây dựng/import module
3. Namspace & Scope
4. Package
5. Python standar library

13.1. LÀM VIỆC VỚI TẬP TIN

- Tập tin (file) và thư mục (folder) là thành phần cơ bản của hệ thống lưu trữ dữ liệu bền vững, tuy có một vài hệ thống không sử dụng những khái niệm này
- Python cung cấp các phương thức cơ bản và cần thiết để thao tác với tập tin như là thành phần cơ bản của ngôn ngữ. Chúng ta có thể thực hiện các thao tác với tập tin bằng cách sử dụng file object.((một số ngôn ngữ lập trình khác xem xử lý tập tin là tính năng mở rộng)

Python chia các tác vụ tập tin làm hai nhóm:

1. Tác vụ quản lý: không ảnh hưởng đến nội dung tập tin (đổi tên, di chuyển, xóa, sao chép, phân quyền,...)
2. Tác vụ nội dung: có tương tác với nội dung tập tin (đọc, ghi,...)

Loại tác vụ quản lý: Python cung cấp nhiều hàm thuộc thư viện os (import os) để xử lý chỉ với 1-2 dòng lệnh

Loại tác vụ nội dung: Python thực hiện quy trình 3 bước:

1. Mở tập tin
2. Xử lý (đọc hoặc ghi)
3. Đóng tập tin tương tự như các ngôn ngữ lập trình khác

Như nhiều ngôn ngữ lập trình khác Python cũng chia tập tin làm hai loại:

1. **Tập tin văn bản**: chứa nội dung chủ yếu là text, các dấu trình bày (tab, xuống dòng, căn lề,...)
   - Python tự động xử lý việc lưu trữ dấu xuống dòng khác nhau giữa các hệ điều hành Linux/Unix và Windows
   - Python hỗ trợ việc tự động chuyển đổi mã hóa (encode) giữa các loại văn bản khác nhau
2. **Tập tin nhị phân**: Python xem như dãy các byte dữ liệu và thường thao tác theo các khối dữ liệu để tăng tốc độ xử lý.

13.2. MỞ - ĐÓNG TẬP TIN

13.2.1.Mở tập tin

Cú pháp:

```
file object = open(file_name[,access_mode][,buffering])
```

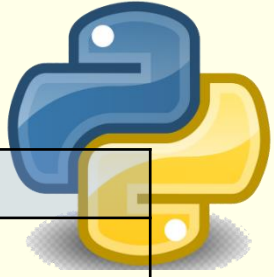
Ý nghĩa của các tham số trong hàm **open**()[5, 10,11,12]:

- **file_name**: tên tập tin sẽ truy cập
- **access_mode**: chế độ mở tập tin: read, write, append...Tham số này là tùy chọn và chế độ mặc đinh là đọc (**r**)
- **buffering**: 1: Dùng để thiết lập bộ đệm. Nếu giá trí thiết lập là 0 thì không sử dụng bộ nhớ đệm. Nếu giá trị là 1 thì thiết lập line bufferring. Nếu giá trị là số < 0 thì thiết lập kích thước bộ đệm mặc định

Ví dụ 13.1 mở tập tin bai_tho.txt

```
f = open("bai_tho.txt", "r")
```

Bảng 13.1. Danh sách access_mode khi mở tập tin



Giá trị	Ý nghĩa
r	Mở tập tin văn bản chỉ để đọc
r+	Mở tập tin văn bản để đọc và ghi
rb	Mở tập tin nhị phân chỉ để đọc
rb+, r+b	Mở tập tin nhị phân để đọc và ghi
w	Mở tập tin văn bản để ghi, nếu tập tin không tồn tại thì sẽ tạo mới
w+	Mở tập tin văn bản để đọc và ghi, nếu tập tin không tồn tại thì sẽ tạo mới
wb	Mở tập tin nhị phân để ghi, nếu tập tin không tồn tại thì sẽ tạo mới
wb+, w+b	Mở tập tin nhị phân để đọc và ghi, nếu tập tin không tồn tại thì sẽ tạo mới
a	Mở tập tin văn bản để ghi tiếp vào cuối nếu tập tin đã tồn tại, nếu tập tin không tồn tại thì sẽ tạo mới
a+	Mở tập tin văn bản để đọc và ghi tiếp vào cuối nếu tập tin đã tồn tại, nếu tập tin không tồn tại thì sẽ tạo mới
ab	Mở tập tin nhị phân để ghi tiếp vào cuối nếu tập tin đã tồn tại, nếu tập tin không tồn tại thì sẽ tạo mới
ab+, a+b	Mở tập tin nhị phân để đọc và ghi tiếp vào cuối nếu tập tin đã tồn tại, nếu tập tin không tồn tại thì sẽ tạo mới
x	Tạo tập tin văn bản mới để ghi, sinh lỗi nếu tập tin đã tồn tại
x+	Tạo tập tin văn bản mới để đọc và ghi, sinh lỗi nếu tập tin đã tồn tại
xb	Tạo tập tin nhị phân mới để ghi, sinh lỗi nếu tập tin đã tồn tại
xb+, x+b	Tạo tập tin nhị phân mới để đọc và ghi, sinh lỗi nếu tập tin đã tồn tại
b	Mở tập tin nhị phân để đọc
t	Mở tập tin văn bản để đọc (đây là giá trị mặc định của mode khi gọi hàm open)

Các thuộc tính của đối tượng file

Khi một file được mở và một đối tượng file được tạo ra, lập trình viên có thể lấy về các thông tin liên quan đến file thông qua các thuộc tính của đối tượng file. Xem danh sách thuộc tính trong bảng 13.2

Bảng 13.2. Danh sách thuộc tính đối tượng file

Thuộc tính	**Ý nghĩa**
file.**closed**	Trả lại **true** nếu file đã được đóng, nếu file vẫn còn mở thì trả lại **false**
file.**mode**	Trả lại chế độ truy nhập (access mode) của file đang mở.
file.**name**	Trả về tên file

Đóng tập tin

Cú pháp:

```
fileObject.close()
```

Ví dụ 13.2

```
f.close()
```

Ghi nhớ:

Thao tác đóng file rất quan trọng:

- Đẩy mọi dữ liệu trên vùng đệm xuống thiết bị lưu trữ.
- Cập nhật thông tin trên hệ thống file (filesize, last update,…)
- Giải phóng vùng dữ liệu dùng cho làm việc với file

Vì vậy khi không sử dụng đến file nữa, nên đóng file ngay!
Vậy việc quên đóng file có gây lỗi không? Câu trả lời là "Không có lỗi", hệ thống tự đóng tất cả các file đang mở khi kết thúc chương trình hoặc đối tượng tham chiếu đếm file được gán cho file khác. Tuy nhiên, sử dụng phương thức close() để đóng một file vẫn tốt hơn.

Ví dụ 13.2a.

```
f = open("test.txt",encoding = 'utf-8')# thực hiện các thao tác với tệp f.close()
```

Để đảm bảo hơn, người lập trình nên sử dụng khối lệnh **try**...**finally** (finally sẽ luôn luôn được thực thi bất chấp có hay không ngoại lệ) ở đây. Chúng ta viết lại ví dụ 13.2a như sau:

Ví dụ 13.2b.

```
try: f = open("test.txt",encoding = 'utf-8') # thực hiện các thao tác với tệp
finally: f.close()
```

Với cách sử dụng khối lệnh **try….finally** như trên, lập trình viên có thể yên tâm file được đóng đúng ngay cả khi phát sinh ngoại lệ khiến chương trình dừng đột ngột.

Một cách khác để đóng file là sử dụng câu lệnh **with**[15]. Lệnh with cho chúng ta bảo đảm rằng file luôn luôn được đóng mà không cần biết hay quan tâm những logic xử lý bên trong.Tiếp theo chúng ta xem xét ví dụ 13.2c dưới đây:

Ví dụ 13.2c.

```
with open("test.txt",encoding = 'utf-8') as f: # thực hiện các thao tác với tệp
```

13.2.3. Ghi và đọc tập tin

- Để ghi một file ta cần mở file bằng cú pháp để ghi, sử dụng mode write '**w**', append '**a**' hoặc mode tạo độc quyền '**x**' (xem danh mục chế độ trong bảng 13.1)
- Cần rất cẩn thận với chế độ '**w**', vì nó ghi đè lên nội dung nếu file đã tồn tại, các dữ liệu trước đó sẽ bị xóa.

a. Ghi tập tin với write()

Các hàm ghi dữ liệu ra file

- `write(data)`: ghi dữ liệu ra file, trả về số byte ghi được
  - Phương thức làm việc với cả file văn bản và file nhị phân
  - Nếu file văn bản thì data phải là kiểu str
  - Nếu file nhị phân thì data phải là khối byte (kiểu bytearray hoặc kiểu bytes)

  Cú pháp: `fileObject.write(data)`

- `writelines(data)`: ghi toàn bộ nội dung data vào file theo từng dòng
  - Chỉ làm việc với kiểu file văn bản
  - Dữ liệu data phải là danh sách các str
  - Nếu cố dùng kiểu dữ liệu khác sẽ phát sinh lỗi TypeError

  Cú pháp: `fileObject.writelines(data)`

Ví dụ 13.3a.

```
1 with open("khud_ab_12_22.txt",'w',encoding = 'utf-8') as f:
2     f.write("Chào Anh/Chị sinh viên \n")
3     f.write("Ngành Khoa học Dữ liệu\n\n")
4     f.write("Khoa Khoa học ứng dụng\n")
```

Kết quả: Tạo ra fille kiểu .txt là **khud_ab_12_22.txt** được lưu mặc định tại thư mục python36

C:\Users\PC\AppData\Local\Programs\Python\Python36\khud\khud_ab_12_22.txt, với nội dung là:

```
Chào cac Anh/Chi sinh vien
Ngành Khoa hoc du lieu

Khoa Khoa hoc ung dung.
```

Ví dụ 13.3b Mở và ghi nội dung vào cuối tập tin file1.txt

```
>>> f = open('file1.txt', 'a+')
>>> f.write("Hello, we are Python")
>>> f.close()
```

Kết quả:

```
Hello, we are Python
```

b. Đọc tập tin với read()

Các hàm đọc file

❑ **Hàm read(N):** đọc N byte tiếp theo

Cú pháp: `fileObject.read(N)`

Trong đó:

- N là số byte sẽ đọc khi mở tập tin, nếu không có thì đọc từ đầu đến cuối
- Nếu dữ liệu trong file không đủ N byte thì đọc đến cuối file
- Nếu tập tin mở ở chế độ văn bản thì trả về str
- Nếu tập tin mở ở chế độ nhị phân thì trả về dãy byte

Ví dụ 13.4a

```
1 f = open("bai_tho.txt",'r',encoding = 'utf-8')
2 a = f.read(25)     # đọc 25 kí tự đầu tiên
3 print('Nội dung 25 kí tự đầu là:\n', a)
4 b = f.read(35)     # đọc 35 kí tự tiếp theo
5 print('Nội dung 35 kí tự tiếp theo là:\n', b)
6 c = f.read()       # đọc phần còn lại
7 print('Nội dung phần còn lại là:\n', c)
```

```
Hom qua tat nuoc dau dinh
Nội dung 35 ký tự tiếp theo là:
Nội dung phần còn lại là:

Em duoc thi cho anh xin
Hay la em de lam tin trong nha

Chieu chieu ra ben Van Lau
Ai ngoi ai cau, ai sau ai tham
```

❑ ***Hàm* readline(N):** đọc một dòng từ file, tối đa là N byte, nếu không viết N thì trả về **str** là dữ liệu được đọc tới khi nào gặp kí tự hết dòng hoặc hết file

Cú pháp: `fileObject.readline(N)`

Trong đó:

- Dữ liệu trả về bao gồm cả kí tự xuống dòng \n, trừ tình huống đọc dòng cuối cùng của file

Ví dụ 13.4b

```
1 f = open("test.txt",'r',encoding = 'utf-8')
2 a = f.readline()
3 print('Nội dung dòng đầu: ', (a))
4 b = f.readline()
5 print('Nội dung dòng 2: ', (b))
6 c = f.readline()
7 print('Nội dung dòng 3: ', (c))
8 d = f.readline()
9 print('Nội dung dòng 4: ', (d))
```

Kết quả chạy ví dụ 13.4b.py

```
Nội dung dòng 1:  Hom qua tat nuoc dau dinh

Nội dung dòng 2:  Bo quen chiec ao tren canh hoa sen

Nội dung dòng 3:  Em duoc thi cho anh xin

Nội dung dòng 4:  Hay la em de lam tin trong nha
```

- Hàm **`readlines (N)`** : sử dụng readlines() đọc các dòng cho đến hết file và trả về một *danh sách các string*, nếu viết N thì sẽ xử lý tối đa là N byte và trả về giá trị rỗng khi kết thúc file.

Ví dụ 13.4c

```
1 f = open("bai_tho.txt",'r',encoding = 'utf-8')
2 a = f.readline()
3 print('Nội dung dòng đầu: ', (a))
4 b = f.readlines()
5 print('Nội dung các dòng còn lại: \n', (b))
6 c = f.readlines()
7 print('Nội dung các dòng còn lại: \n', (c))
```

hàm readline(), đọc hết dòng đầu tiên trả về chuỗi a

hàm readlines(), đọc các dòng đết hết file và trả về 1 danh sách b

hàm readlines(), trả về danh sách rỗng c = [] khi kết thúc file

Kết quả

```
Nội dung dòng đầu:  Hom qua tat nuoc dau dinh
Nội dung các dòng còn lại:
['Bo quen chiec ao tren canh hoa sen\n', 'Em duoc thi cho anh xin\n', 'Hay la em de lam tin trong nha\n', '\n', 'Chieu chieu ra ben Van Lau\n', 'Ai ngoi ai cau, ai sau ai tham']
Nội dung các dòng còn lại:
[]
```

c. *Con trỏ tập tin*

- Con trỏ tập tin là vị trí hiện thời sẽ đọc/ghi dữ liệu, tương tự như khi ta ghi dữ liệu lên màn hình.
- Một tập tin chỉ có một con trỏ tập tin.
- Khi mở tập tin ở chế độ "thêm cuối" (a – append), con trỏ tập tin tự động đặt ở cuối tập tin.
- Các chế độ mở tập tin khác luôn đặt con trỏ tập tin ở đầu.

Python cung cấp một số lệnh cho phép lấy vị trí và di chuyển con trỏ tập tin. Trong bảng 13.3 trình bày một số phương thức làm việc với con trỏ tập tin.[6, 9]

- *Chỉ nên sử dụng với tập tin nhị phân*
- *Không phải loại tập tin nào cũng lấy được vị trí hoặc dịch chuyển được con trỏ*

Bảng 13.3. Phương thức làm việc với con trỏ tập tin

Phương thức	Chức năng
`seek(offset, from)`	Thay đổi vị trị hiện tại của tập tin, trong đó: ▪ **offset** : số lượng byte được di chuyển, ▪ **from** vị trí từ nơi các byte sẽ được di chuyển (**0**: tính từ đầu tập tin (mặc định); **1**: tính từ vị trí hiện thời; **2**: tính từ cuối tập tin)
`seekable()`	Trả về **True** nếu tập tin là dạng truy cập ngẫu nhiên (dùng được phương thức **seek** ở trên)
`tell()`	Trả về vị trí con trỏ tập tin hiện tại (tính từ đầu tập tin)

Ví dụ 13.5

```
1  #Ví dụ về con trỏ tập tin, hàm tell(), seek()
2  f = open("bai_tho.txt", "r+")
3  a = f.read(12) # đọc 12 kí tự đầu tiên
4  print('Nội dung là: \n', a)
5  b = f.tell() # Kiểm tra vị trí hiện tại
6  print ('Vị trí hiện tại: ', b)
7  f.seek(0) # Đặt lại vị trí con trỏ tại vị trí đầu file
8  c = f.read(12)
9  print('Các ký tự đọc được sau khi về vị trí đâu tiên là: \n', c)
```

Kết quả:

```
Nội dung là:
 Hom qua tat
Vị trí hiện tại:  12
Các ký tự đọc được sau khi về vị trí đâu tiên là:
 Hom qua tat
```

Ví dụ 13.6. Đọc tập tin heights.txt là chiều cao các vận động viên (theo inch). Tính chiều cao trung bình theo mét

```
#Đọc tập tin heights.txt
f = open('heights.txt')
data = f.read() #biến data kiểu chuỗi
f.close()

#đổi sang list, phần tử kiểu chuỗi
lst = data.split(",")

#Tạo list height là các phần tử kiểu số thực
height = [float(item) for item in lst]

#Đổi đơn vị đo inch sang mét
height_m = [h*0.0254 for h in height]
#Tính chiều cao trung bình
mean = sum(height_m)/len(height_m)
print('Chiều cao trung bình :', round(mean,2))
```

Kết quả

```
Chiều cao trung bình : 1.87
```

d.Đổi tên và xóa tập tin: với module OS

Hàm đổi tên tập tin rename()

Cú pháp: **os.rename(current_file_name, new_file_name)**

Ví dụ 13.7 đổi tên tập tin **bai_tho.txt** thành **file1.txt**

```
import os
os.rename('bai_tho.txt', 'file1.txt')
```

e.Hàm xóa tập tin remove()

Cú pháp: **os.remove(file_name)**

Ví dụ 13.8: Xóa tập tin tin **file1.txt**

```
import os
os.remove('file1.txt')
```

13.3.1. Khái niệm tập tin CSV

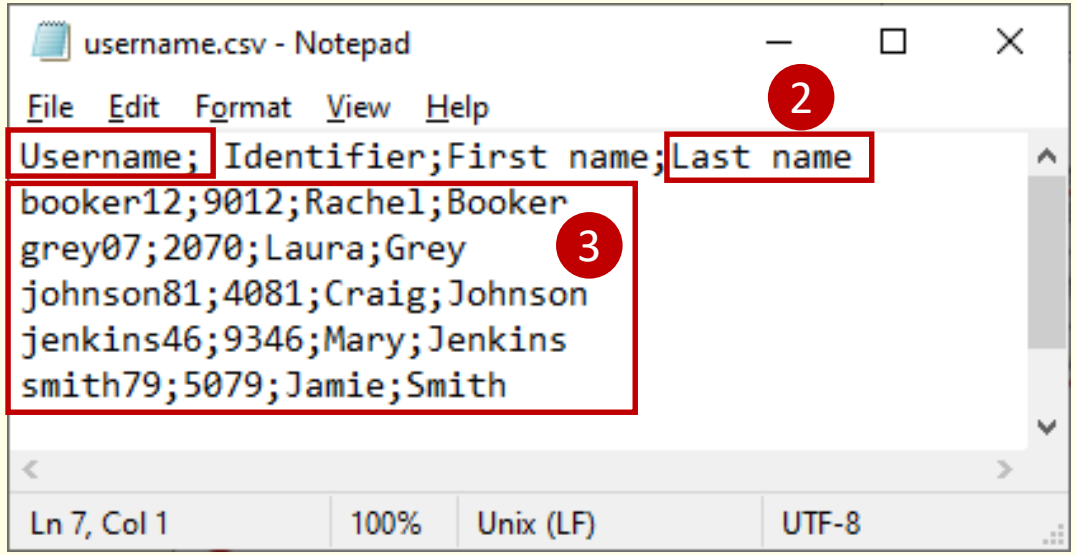
❑ CSV(Comma Separated Values) là một tệp tin bảng tính rất phổ biến và được sử dụng rộng rãi

- CSV (Comma Separated Values) là một loại định dạng là một định dạng nhập/xuất dữ liệu phổ biến dành cho bảng tính và CSDL (*spreadsheet & database*), trong đó, các giá trị được ngăn cách với nhau bằng dấu phẩy. Định dạng CSV thường xuyên được sử dụng để lưu các bảng tính *quy mô nhỏ* như danh bạ, danh sách lớp, báo cáo…
- CSV file được sử dụng để lưu trữ một lượng lớn các biến hoặc dữ liệu. Đó là các bảng tính đơn giản – ví dụ như **Excel** – chỉ có nội dung được lưu trữ trong plaintext.[10,11,12]

❑ Cấu tạo của file CSV

- Một văn bản CSV gồm nhiều dòng chứa các giá trị và các dấu phẩy. Dòng đầu tiên của văn bản CSV chứa tên của từng cột trong bảng tính, mỗi cột được xác định bằng 2 dấu phẩy/chấm phẩy (trừ cột đầu tiên và cuối cùng). Tất cả những dòng sau đó đều có cấu trúc tương tự, chứa các giá trị tương ứng của từng cột. Và văn bản là một dòng giá trị khác nhau trên bảng tính.(hình 13.1)

File username.csv mở trong Notepad



File username.csv mở trong excel

	A	B	C	D
1	Username	Identifier	First name	Last name
2	booker12	9012	Rachel	Booker
3	grey07	2070	Laura	Grey
4	johnson81	4081	Craig	Johnson
5	jenkins46	9346	Mary	Jenkins
6	smith79	5079	Jamie	Smith

1. Cột đầu tiên
2. Cột cuối cùng
3. Nội dung giá trị các cột, (tương ứng dòng đầu tiên)

https://support.staffbase.com/hc/en-us/articles/360007108391-CSV-File-Examples

❑ Điểm chung giữa file CSV và file Excel

- Cả CSV và Excel (.xls, .xlsx) đều là những tệp tin giúp lưu dữ liệu dưới dạng bảng tính. Cả hai đều có thể được mở bằng phần mềm bảng tính (phần mềm spreadsheet) như Microsoft Excel, Google Sheests, Polaris Office, Libre Office, .v.v…). Các hàm, công thức được hỗ trợ bởi những phần mềm spreadsheet đều sẽ áp dụng được trong cả 2 loại tập tin khi mở bằng phần mềm đó.

❑ Sự khác nhau giữa CSV và Excel

- *Dung lượng file CSV nhẹ hơn rất nhiều*

  Định dạng CSV chỉ chứa những giá trị đơn thuần, bao gồm số, chữ và các ký tự khác. CSV được xem như là một tệp tin phẳng (flat file), gần như là một file văn bản (txt) thông thường. Vì thế mà dung lượng của một file CSV rất nhẹ, nhẹ hơn nhiều so với Excel có cùng số lượng nội dung.
- *Quá trình tạo CSV ít phức tạp hơn*

  Cú pháp của một tệp tin CSV rất đơn giản, có thể nhanh chóng ghi lại dữ liệu mà không cần quan tâm đến định dạng chữ viết, font chữ hay màu sắc.
- *CSV không có khả năng thể hiện biểu đồ, không chứa định dạng*

  CSV không thể chứa được những giá trị nào khác ngoài chữ, số và ký tự. Hơn nữa, CSV cũng không thể xác định các các định dạng như ngày giờ, đơn vị tiền tệ, in đậm, in nghiêng…, chọn màu cho nội dung, chèn link, điều chỉnh vị trí văn bản (căn giữa, căn trái, căng phải, căn đều),.v.v…

❑ **CSV có thể được mở và chỉnh sửa nội dung với trình soạn thảo văn bản**

- Bên cạnh mở bằng các chương trình spreadsheet, tệp CSV có thể được mở và chỉnh được với các trình soạn thảo văn bản như notepad. Điều này giúp người sử dụng có thể đọc và chỉnh sửa tệp CSV mọi lúc mọi nơi dễ dàng hơn, nhất là trên smartphone, khi mà việc thao tác bằng các phần mềm spreadsheet khá bất tiện.

❑ Python cung cấp thư viện csv dùng để đọc và ghi dữ liệu dạng bảng ở định dạng CSV.

❑ Cần import csv khi làm việc với tập tin csv.

13.3.2. Thao tác trên tập tin (file) CSV

a. Mở - đọc file csv [10,11,12]

Cú pháp mở: `f=open(filename)`
Cú pháp đọc: `csv.reader(f)`
Cú pháp đóng: `f.close()`

Ví dụ 13.9: Mở - đọc từng dòng nội dung – đóng file

```
1   import csv
2   def read_csv_file(filename):
3       f=open(filename)
4       for row in csv.reader(f):
5           print(row)
6       f.close()
```

Ví dụ 13.10 Mở - đọc từng dòng nội dung và đưa vào list – đóng file

```
1   def read_csv_file_to_list(filename):
2       f=open(filename)
3       data = []
4       for row in csv.reader(f):
5           data.append(row)
6       f.close()
7       return data
```

Ví dụ 13.11: Mở - đọc từng dòng nội dung theo từng cột – đóng file

```
1   def read_csv_file_column(filename):
2       f=open(filename)
3       for row in csv.reader(f):
4           print(row[0], '\t', row[1])
5       f.close()
```

Ví dụ 13.12: Đọc file data_1**.csv** theo dòng và in ra dòng

```
1   from csv import reader
2   with open('data_1.csv', 'r') as read_obj:
3       csv_reader = reader(read_obj)
4       for row in csv_reader:
5           print(row)
```

Kết quả:
```
['Programming laguage;Designed by;Appeared;Extension']
['Python;Guido van Rossum;1991;.py']
['Java;James Gosling;1995;.java']
['C++;Bjarne Stroustrup;1983;.cpp']
```

Ví dụ 13.13: Đọc file **data_1.csv** theo dòng và in ra dòng

```
1   from csv import reader
2   with open('data_1.csv', 'r') as read_obj:
3       csv_reader = reader(read_obj)
4       for row in csv_reader:
5           print(row)
```

Chạy chương trình và kết quả thực hiện như trong ví dụ 13.12.

***Ghi nhớ**: **Context manager** thường được sử dụng để lock các tài nguyên (trường hợp mở và đóng file là một ví dụ kinh điển cho việc này).*

Ví dụ 13.14: đọc file data_1.csv ra list

```
1   from csv import reader
2
3   with open('data_1.csv', 'r') as read_obj:
4       csv_reader = reader(read_obj)
5       list_kq = list(csv_reader)
6
7   print(list_kq)
```

b. Mở - ghi file csv

Cú pháp mở: **`f = open(filename, filemode,newline='')`**

- Cú pháp ghi: **`csv.writer(f).writerow(content)`**

Ví dụ 13.15: mở - ghi danh sách – đóng file

```
1    def wite_csv_file(filename, listContent):
2        f=open(filename,'a',newline='')
3        for item in listContent:
4            csv.write(f).writerow(item)
5        f.close()
```

13.4. LÀM VIỆC VỚI THƯ MỤC

13.4.1.Tạo thư mục

Cú pháp: **os.mkdir**(“tên_thư_mục")

Ví dụ 13.16 : tạo thư mục bai_tap_13

```
import os
os.mkdir("bai_tap_13")
```

13.4.2.Thay đổi thư mục hiện hành

Cú pháp: os.**chdir**(“tên_mới")

Ví dụ 13.17 : thay đổi thư mục hiện tại test

```
import os
os.chdir("test")
```

13.4.3. Hiển thị thư mục

Cú pháp:

os.**getcwd**()

Ví dụ 13.18:

```
import os
os.getcwd()
```

13.4.4. Xóa thư mục

Cú pháp: os.**rmdir**(“tên_thư_mục")

Ví dụ 13.19 : xóa

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

Câu hỏi thảo luận

1. Nêu cú pháp cách mở/đóng tập tin trong python?
2. Con trỏ tập tin là gì?
3. Phân biệt file văn bản text (.txt) và file CSV?

Bài tập vận dụng.

Bài tập chương 13 trong TLHT