


Модуль csv - чтение и запись CSV файлов

Python имеет встроенную библиотеку `csv`, которая предоставляет функции для чтения и записи данных в формате CSV. Давайте рассмотрим, как использовать эту библиотеку.

Чтение CSV-файлов

Для чтения CSV-файлов сначала импортируйте библиотеку `csv` и используйте функцию `csv.reader()`:

1	<code>import csv</code>
2	<code>with open('example.csv','r') as csvfile:</code>
3	<code> csvreader = csv.reader(csvfile)</code>
4	<code> for row in csvreader:</code>
5	<code> print(row)</code>
6	

 В этом примере мы открываем файл `example.csv` на чтение и передаем его функции `csv.reader()`. Затем мы проходимся по каждой строке и выводим ее.

Чтение

Пример чтения файла в формате CSV (файл `csv_read.py`):

```
import csv

with open('sw_data.csv') as f:
    reader = csv.reader(f)
    for row in reader:
        print(row)
```

Вывод будет таким:

```
$ python csv_read.py
['hostname', 'vendor', 'model', 'location']
['sw1', 'Cisco', '3750', 'London']
['sw2', 'Cisco', '3850', 'Liverpool']
['sw3', 'Cisco', '3650', 'Liverpool']
['sw4', 'Cisco', '3650', 'London']
```

В первом списке находятся названия столбцов, а в остальных соответствующие значения.

Обратите внимание, что сам csv.reader возвращает итератор:

```
In [1]: import csv
```

```
In [2]: with open('sw_data.csv') as f:
...: reader = csv.reader(f)
...: print(reader)
...:
<_csv.reader object at 0x10385b050>
```

При необходимости его можно превратить в список таким образом:

```
In [3]: with open('sw_data.csv') as f:
...: reader = csv.reader(f)
...: print(list(reader))
...:
['hostname', 'vendor', 'model', 'location'], ['sw1', 'Cisco', '3750', 'London'], ['sw2', 'Cisco', '3850', 'Liverpool'], ['sw3', 'Cisco', '3650', 'Liverpool'], ['sw4', 'Cisco', '3650', 'London']
```

Чаще всего заголовки столбцов удобнее получить отдельным объектом. Это можно сделать таким образом (файл csv_read_headers.py):

```
import csv

with open('sw_data.csv') as f:
    reader = csv.reader(f)
    headers = next(reader)
    print('Headers: ', headers)
    for row in reader:
        print(row)
```

Иногда в результате обработки гораздо удобнее получить словари, в которых ключи - это названия столбцов, а значения - значения столбцов.

Для этого в модуле есть **DictReader** (файл csv_read_dict.py):

```
import csv

with open('sw_data.csv') as f:
    reader = csv.DictReader(f)
    for row in reader:
        print(row)
        print(row['hostname'], row['model'])
```

Вывод будет таким:

```
$ python csv_read_dict.py
{'hostname': 'sw1', 'vendor': 'Cisco', 'model': '3750', 'location': 'London, Globe Str 1 '}
sw1 3750
{'hostname': 'sw2', 'vendor': 'Cisco', 'model': '3850', 'location': 'Liverpool'}
sw2 3850
{'hostname': 'sw3', 'vendor': 'Cisco', 'model': '3650', 'location': 'Liverpool'}
sw3 3650
{'hostname': 'sw4', 'vendor': 'Cisco', 'model': '3650', 'location': 'London, Grobe Str 1'}
sw4 3650
```

Запись

Аналогичным образом с помощью модуля csv можно и записать файл в формате CSV (файл csv_write.py):

```
import csv

data = [['hostname', 'vendor', 'model', 'location'],
['sw1', 'Cisco', '3750', 'London, Best str'],
['sw2', 'Cisco', '3850', 'Liverpool, Better str'],
['sw3', 'Cisco', '3650', 'Liverpool, Better str'],
['sw4', 'Cisco', '3650', 'London, Best str']]

with open('sw_data_new.csv', 'w') as f:
    writer = csv.writer(f)
    for row in data:
        writer.writerow(row)

with open('sw_data_new.csv') as f:
    print(f.read())
```

В примере выше строки из списка сначала записываются в файл, а затем содержимое файла выводится на стандартный поток вывода.

Вывод будет таким:

```
$ python csv_write.py
hostname,vendor,model,location
sw1,Cisco,3750,"London, Best str"
sw2,Cisco,3850,"Liverpool, Better str"
sw3,Cisco,3650,"Liverpool, Better str"
sw4,Cisco,3650,"London, Best str"
```

Обратите внимание на интересную особенность: строки в последнем столбце взяты в кавычки, а остальные значения - нет.

Так получилось из-за того, что во всех строках последнего столбца есть запятая. И кавычки указывают на то, что именно является целой строкой. Когда запятая находится в кавычках, модуль csv не воспринимает её как разделитель.

Иногда лучше, чтобы все строки были в кавычках. Конечно, в данном случае достаточно простой пример, но когда в строках больше значений, то кавычки позволяют указать, где начинается и заканчивается значение.

Модуль csv позволяет управлять этим. Для того, чтобы все строки записывались в CSV-файл с кавычками, надо изменить скрипт таким образом (файл csv_write_quoting.py):

```
import csv

data = [['hostname', 'vendor', 'model', 'location'],
['sw1', 'Cisco', '3750', 'London, Best str'],
['sw2', 'Cisco', '3850', 'Liverpool, Better str'],
['sw3', 'Cisco', '3650', 'Liverpool, Better str'],
['sw4', 'Cisco', '3650', 'London, Best str']]

with open('sw_data_new.csv', 'w') as f:
    writer = csv.writer(f, quoting=csv.QUOTE_NONNUMERIC)
    for row in data:
        writer.writerow(row)

with open('sw_data_new.csv') as f:
    print(f.read())
```

Теперь вывод будет таким:

```
$ python csv_write_quoting.py
"hostname","vendor","model","location"
"sw1","Cisco","3750","London, Best str"
"sw2","Cisco","3850","Liverpool, Better str"
"sw3","Cisco","3650","Liverpool, Better str"
"sw4","Cisco","3650","London, Best str"
```

Теперь все значения с кавычками. И поскольку номер модели задан как строка в изначальном списке, тут он тоже в кавычках.

Кроме метода `writerow`, поддерживается метод `writerows`. Ему можно передать любой итерируемый объект.

Например, предыдущий пример можно записать таким образом (файл csv_writerows.py):

```
import csv
```

```

data = [['hostname', 'vendor', 'model', 'location'],
['sw1', 'Cisco', '3750', 'London, Best str'],
['sw2', 'Cisco', '3850', 'Liverpool, Better str'],
['sw3', 'Cisco', '3650', 'Liverpool, Better str'],
['sw4', 'Cisco', '3650', 'London, Best str']]

with open('sw_data_new.csv', 'w') as f:
writer = csv.writer(f, quoting=csv.QUOTE_NONNUMERIC)
writer.writerows(data)

with open('sw_data_new.csv') as f:
print(f.read())

```

DictWriter

С помощью DictWriter можно записать словари в формат CSV.

В целом DictWriter работает так же, как writer, но так как словари не упорядочены, надо указывать явно в каком порядке будут идти столбцы в файле. Для этого используется параметр fieldnames (файл csv_write_dict.py):

```

import csv

data = [{
'hostname': 'sw1',
'location': 'London',
'model': '3750',
'vendor': 'Cisco'
}, {
'hostname': 'sw2',
'location': 'Liverpool',
'model': '3850',
'vendor': 'Cisco'
}, {
'hostname': 'sw3',
'location': 'Liverpool',
'model': '3650',
'vendor': 'Cisco'
}, {
'hostname': 'sw4',
'location': 'London',
'model': '3650',
'vendor': 'Cisco'
}]

```

```
with open('csv_write_dictwriter.csv', 'w') as f:
writer = csv.DictWriter(
f, fieldnames=list(data[0].keys()), quoting=csv.QUOTE_NONNUMERIC)
writer.writeheader()
for d in data:
writer.writerow(d)
```

Указание разделителя

Иногда в качестве разделителя используются другие значения. В таком случае должна быть возможность подсказать модулю, какой именно разделитель использовать.

Например, если в файле используется разделитель ; (файл sw_data2.csv):

```
hostname;vendor;model;location
sw1;Cisco;3750;London
sw2;Cisco;3850;Liverpool
sw3;Cisco;3650;Liverpool
sw4;Cisco;3650;London
```

Достаточно просто указать, какой разделитель используется в reader (файл csv_read_delimiter.py):

```
import csv

with open('sw_data2.csv') as f:
reader = csv.reader(f, delimiter=';')
for row in reader:
print(row)
```