GeeksforGeeks

A computer science portal for geeks

Custom Searcl

Practice GATE CS Placements Videos Contribute

Login/Register

Quick Links for Python

Recent Articles

MCQ / Quizzes

Practice Problems

Basics

Introduction

New Generation Language

Keywords, Set 1 Set 2

Explore More...

Variables

Variables,Expressions & Functions

Global and Local Variables

Type Conversion

Explore More...

Operators

Increment and Decrement Operator

▲

▲

▲

▲

Working with csv files in Python

This article explains how to load and parse a CSV file in Python.

3.5

First of all, what is a CSV ?

CSV (Comma Separated Values) is a simple **file format** used to store tabular data, such as a spreadsheet or database. A CSV file stores tabular data (numbers and text) in plain text. Each line of the file is a data record. Each record consists of one or more fields, separated by commas. The use of the comma as a field separator is the source of the name for this file format.

For working CSV files in python, there is an inbuilt module called **csv**.

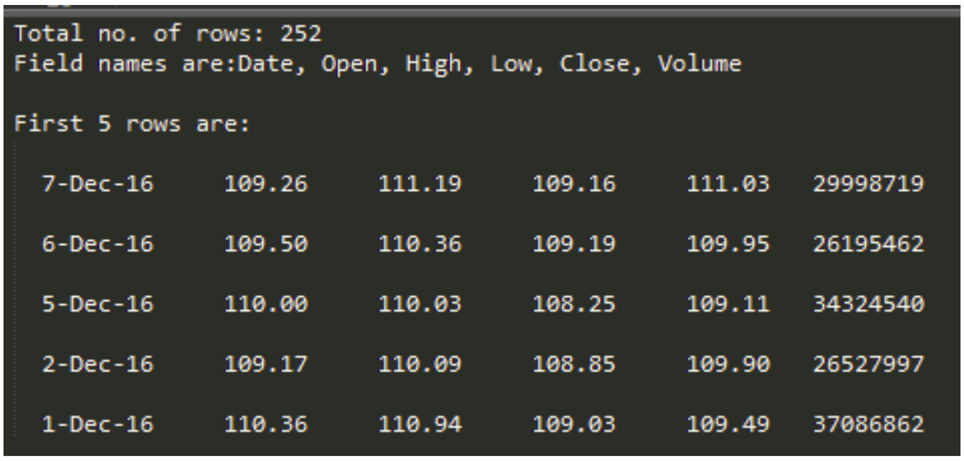
Reading a CSV file

```
# importing csv module
import csv

# csv file name
filename = "aapl.csv"

# initializing the titles and rows list
fields = []
rows = []

# reading csv file
with open(filename, 'r') as csvfile:
    # creating a csv reader object
    csvreader = csv.reader(csvfile)

    # extracting field names through first row
    fields = csvreader.next()

    # extracting each data row one by one
    for row in csvreader:
        rows.append(row)

    # get total number of rows
    print("Total no. of rows: %d"%(csvreader.line_num))

# printing the field names
print('Field names are:' + ', '.join(field for field in fields))

#  printing first 5 rows
```

```
print('\nFirst 5 rows are:\n')
for row in rows[:5]:
    # parsing each column of a row
    for col in row:
        print("%10s"%col),
    print('\n')
```

Run on IDE

The output of above program looks like this:



```
Total no. of rows: 252
Field names are:Date, Open, High, Low, Close, Volume

First 5 rows are:

   7-Dec-16     109.26     111.19     109.16     111.03   29998719

   6-Dec-16     109.50     110.36     109.19     109.95   26195462

   5-Dec-16     110.00     110.03     108.25     109.11   34324540

   2-Dec-16     109.17     110.09     108.85     109.90   26527997

   1-Dec-16     110.36     110.94     109.03     109.49   37086862
```

The above example uses a CSV file aapl.csv which can be downloaded from here. Run this program with the aapl.csv file in same directory.

Let us try to understand this piece of code.

-
  ```
  with open(filename, 'r') as csvfile:
      csvreader = csv.reader(csvfile)
  ```

  Here, we first open the CSV file in READ mode. The file object is named as **csvfile**. The file object is converted to csv.reader object. We save the csv.reader object as **csvreader**.

-
  ```
  fields = csvreader.next()
  ```

  csvreader is an iterable object. Hence, .next() method returns the current row and advances the iterator to the next row. Since the first row of our csv file contains the headers (or field names), we save them in a list called **fields**.

-
  ```
  for row in csvreader:
          rows.append(row)
  ```

  Now, we iterate through remaining rows using a for loop. Each row is appended to a list called **rows**. If you try to print each row, one can find that row is nothing but a list containing all the field values.

-
  ```
  print("Total no. of rows: %d"%(csvreader.line_num))
  ```

  csvreader.line_num is nothing but a counter which returns the number of rows which have been iterated.

Writing to a CSV file

```
# importing the csv module
import csv

# field names
fields = ['Name', 'Branch', 'Year', 'CGPA']

# data rows of csv file
rows = [ ['Nikhil', 'COE', '2', '9.0'],
         ['Sanchit', 'COE', '2', '9.1'],
         ['Aditya', 'IT', '2', '9.3'],
         ['Sagar', 'SE', '1', '9.5'],
         ['Prateek', 'MCE', '3', '7.8'],
         ['Sahil', 'EP', '2', '9.1']]

# name of csv file
filename = "university_records.csv"

# writing to csv file
with open(filename, 'w') as csvfile:
    # creating a csv writer object
    csvwriter = csv.writer(csvfile)

    # writing the fields
    csvwriter.writerow(fields)

    # writing the data rows
    csvwriter.writerows(rows)
```

Run on IDE

Let us try to understand the above code in pieces.

- **fields** and **rows** have been already defined. fields is a list containing all the field names. **rows** is a list of lists. Each row is a list containing the field values of that row.

-
  ```
  with open(filename, 'w') as csvfile:
      csvwriter = csv.writer(csvfile)
  ```

  Here, we first open the CSV file in WRITE mode. The file object is nam **csvfile**. The file object is converted to csv.writer object. We save the csv.wr object as **csvwriter**.

-
  ```
  csvwriter.writerow(fields)
  ```

  Now we use **writerow** method to write the first row which is nothing but the field names.

-
  ```
  csvwriter.writerows(rows)
  ```

  We use **writerows** method to write multiple rows at once.

Writing a dictionary to a CSV file

```
# importing the csv module
import csv

# my data rows as dictionary objects
mydict =[{'branch': 'COE', 'cgpa': '9.0', 'name': 'Nikhil', 'year': '2'},
         {'branch': 'COE', 'cgpa': '9.1', 'name': 'Sanchit', 'year': '2'}
         {'branch': 'IT', 'cgpa': '9.3', 'name': 'Aditya', 'year': '2'},
         {'branch': 'SE', 'cgpa': '9.5', 'name': 'Sagar', 'year': '1'},
         {'branch': 'MCE', 'cgpa': '7.8', 'name': 'Prateek', 'year': '3'}
         {'branch': 'EP', 'cgpa': '9.1', 'name': 'Sahil', 'year': '2'}]

# field names
fields = ['name', 'branch', 'year', 'cgpa']

# name of csv file
filename = "university_records.csv"

# writing to csv file
with open(filename, 'w') as csvfile:
    # creating a csv dict writer object
    writer = csv.DictWriter(csvfile, fieldnames = fields)

    # writing headers (field names)
    writer.writeheader()

    # writing data rows
    writer.writerows(mydict)
```

Run on IDE

In this example, we write a dictionary **mydict** to a CSV file.

-
  ```
  with open(filename, 'w') as csvfile:
      writer = csv.DictWriter(csvfile, fieldnames = fields)
  ```

  Here, the file object (**csvfile**) is converted to a DictWriter object.

  Here, we specify the **fieldnames** as an argument.

-
  ```
  writer.writeheader()
  ```

writeheader method simply writes the first row of your csv file using the pre-specified fieldnames.

-
  ```
  writer.writerows(mydict)
  ```

  writerows method simply writes all the rows but in each row, it writes only the values(not keys).

So, in the end, our CSV file looks like this:

name	branch	year	cgpa
Nikhil	COE	2	9.0
Sanchit	COE	2	9.1
Aditya	IT	2	9.3
Sagar	SE	1	9.5
Prateek	MCE	3	7.8
Sahil	EP	2	9.1

Important Points:

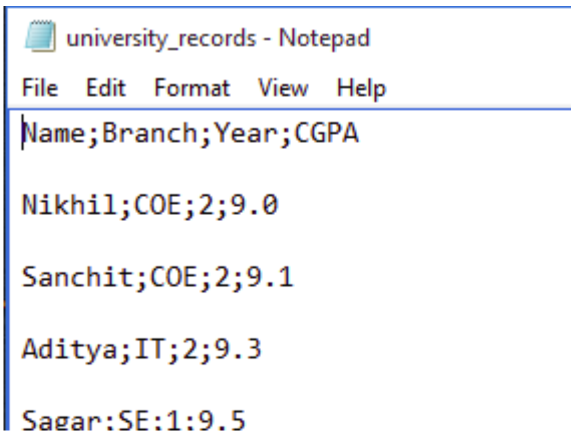
- In csv modules, an optional *dialect* parameter can be given which is used to define a set of parameters specific to a particular *CSV format*. By default, csv module uses *excel* dialect which makes them compatible with excel spreadsheets. You can define your own dialect using **register_dialect** method.
  Here is an example:

```
 csv.register_dialect(
'mydialect',
delimiter = ',',
quotechar = '"',
doublequote = True,
skipinitialspace = True,
lineterminator = '\r\n',
quoting = csv.QUOTE_MINIMAL)
```

Now, while defining a csv.reader or csv.writer object, we can specify the dialect like this:

```
csvreader = csv.reader(csvfile, dialect='mydialect')
```

- Now, consider that a CSV file looks like this in plain-text:



```
Name;Branch;Year;CGPA

Nikhil;COE;2;9.0

Sanchit;COE;2;9.1

Aditya;IT;2;9.3

Sagar;SE;1;9.5
```

We notice that the delimiter is not a comma but a semi-colon. Also, the rows are separated by two newlines instead of one. In such cases, we can specify the delimiter and line terminator as follows:

```
csvreader = csv.reader(csvfile, delimiter = ';', lineterminator = '\n\n'
```

So, this was a brief, yet concise discussion on how to load and parse CSV files in a python program.

This blog is contributed by **Nikhil Kumar**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GATE CS Corner **Company Wise Coding Practice**

GBlog Python Login to Improve this Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Recommended Posts:

Downloading files from web using Python
Graph Plotting in Python | Set 1
GET and POST requests using Python
Whatsapp using Python!
Data analysis and Visualization with Python

▲

Fetch top 10 starred repositories of user on GitHub | Python
Natural Language Programming — Teaching Kids
Activation functions in Neural Networks
Natural Language Programming
Cloud Based Services

(Login to Rate)

3.5 Average Difficulty : **3.5/5.0**
Based on **2** vote(s)

Basic Easy Medium Hard Expert

Add to TODO List
Mark as DONE

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments Share this post!

▲