**CSV**

## CSV

CSV (Comma Separated Values) is a very popular import and export data format used in spreadsheets and databases. Each line in a CSV file is a data record. Each record consists of one or more fields, separated by commas. While CSV is a very simple data format, there can be many differences, such as different delimiters, new lines, or quoting characters.

## Python csv module

The csv module implements classes to read and write tabular data in CSV format. The csv module's reader and writer objects read and write sequences. Programmers can also read and write data in dictionary form using the DictReader and DictWriter classes.

## Python CSV methods

The following table shows Python csv methods:

| **Method** | **Description** |
| --- | --- |
| csv.reader | returns a reader object which iterates over lines of a CSV file |
| csv.writer | returns a writer object which writes data into CSV file |
| csv.register\_dialect | registers a CSV dialect |
| csv.unregister\_dialect | unregisters a CSV dialect |
| csv.get\_dialect | returns a dialect with the given name |
| csv.list\_dialects | returns all registered dialects |
| csv.field\_size\_limit | returns the current maximum field size allowed by the parser |

## Using Python csv module

import csv

To use Python CSV module, we import csv.

## Python CSV reader

The csv.reader() method returns a reader object which iterates over lines in the given CSV file.

Prepare as CSV file with name “numbers.csv” by using any text editor. The content of this file is just one line as listed below:

16,6,4,12,81,6,71,6

Prepare the following code to test:

|  |
| --- |
| **CSV01.py** |
| **Line** | **Code** |
| **1****2****3****4****5****6****7****8** | **import** csvf = **open**('numbers.csv', 'r')with f: reader = csv.reader(f) **for** row **in** reader: **for** e **in** row: **print**(e) |

With two for loops, we iterate over the data.

**Output:**

16

6

4

12

81

6

71

6

**Python CSV reader with different delimiter**

The csv.reader() method allows to use a different delimiter with its delimiter attribute.

Prepare as CSV file with name “members.csv” by using any text editor. The content of this file is below:

ID;Name;Age;EMail
100;Ali;21;ali@gmail.com
102;Abu;25;abu@gmail.com
105;Ahmad;23;ahmad@abc.com

The items.csv contains values separated with ';' character.

Prepare the following python code to test:

|  |
| --- |
| **CSV02.py** |
| **Line** | **Code** |
| **1****2****3****4****5****6****7****8** | **import** csvf = **open**('members.csv', 'r')with f: reader = csv.reader(f, delimiter=";") **for** row **in** reader: **for** e **in** row: **print**(e,end="\t") **print**(); |

**Output:**

ID Name Age EMail

100 Ali 21 ali@gmail.com

102 Abu 25 abu@gmail.com

105 Ahmad 23 ahmad@abc.com

## Python CSV DictReader

The csv.DictReader class operates like a regular reader but maps the information read into a dictionary. The keys for the dictionary can be passed in with the fieldnames parameter or inferred from the first row of the CSV file.

Prepare as CSV file with name “valuess.csv” by using any text editor. The content of this file is below:

min,avg,max

1, 5.5, 10

2, 3.5, 5

The first line of the file consists of dictionary keys.

Prepare the following python code to test:

|  |
| --- |
| **CSV03.py** |
| **Line** | **Code** |
| **1****2****3****4****5****6****7** | **import** csvf = **open**('values.csv', 'r')with f: reader = csv.DictReader(f) **for** row **in** reader: **print**(row['min'], row['avg'], row['max']) |

**Output:**

1 5.5 10

2 3.5 5

The example reads the values from the values.csv file using the csv.DictReader.

for row in reader:

 print(row['min'], row['avg'], row['max'] )

The row is a Python dictionary and we reference the data with the keys.

## Python CSV writer

The csv.writer() method returns a writer object which converts the user's data into delimited strings on the given file-like object.

|  |
| --- |
| **CSV04.py** |
| **Line** | **Code** |
| **1****2****3****4****5****6****7****8****9****10****11** | **import** csvnms = [[1, 2, 3, 4, 5, 6], [7, 8, 9, 10, 11, 12]]f = **open**('numbers2.csv', 'w')with f: writer = csv.writer(f) **for** row **in** nms: writer.writerow(row) |

After running the above python code, a file name “numbers2.csv” will be created with the following contents:

1,2,3,4,5,6

7,8,9,10,11,12

It is possible to write all data in one shot. The writerows() method writes all given rows to the CSV file.

Prepare the following python code:

|  |
| --- |
| **CSV05.py** |
| **Line** | **Code** |
| **1****2****3****4****5****6****7****8****9** | **import** csvnms = [[1, 2, 3], [7, 8, 9], [10, 11, 12]]f = **open**('numbers3.csv', 'w')with f: writer = csv.writer(f) writer.writerows(nms) |

The code example writes three rows of numbers into the file using the writerows() method.

1,2,3

7,8,9

10,11,12

## Python CSV DictWriter

The csv.DictWriter class operates like a regular writer but maps Python dictionaries into CSV rows. The fieldnames parameter is a sequence of keys that identify the order in which values in the dictionary passed to the writerow() method are written to the CSV file.

|  |
| --- |
| **CSV06.py** |
| **Line** | **Code** |
| **1****2****3****4****5****6****7****8****9****10****11****12** | **import** csvf = **open**('names.csv', 'w')with f: fnames = ['first\_name', 'last\_name'] writer = csv.DictWriter(f, fieldnames=fnames) writer.writeheader() writer.writerow({'first\_name': 'John', 'last\_name': 'Smith'}) writer.writerow({'first\_name': 'Robert', 'last\_name': 'Brown'}) writer.writerow({'first\_name': 'Julia', 'last\_name': 'Griffin'}) |

he example writes the values from Python dictionaries into the CSV file using the csv.DictWriter.

writer = csv.DictWriter(f, fieldnames=fnames)

New csv.DictWriter is created. The header names are passed to the fieldnames parameter.

writer.writeheader()

The writeheader() method writes the headers to the CSV file.

writer.writerow({'first\_name' : 'John', 'last\_name': 'Smith'})

The Python dictionary is written to a row in a CSV file.

**Output:**

first\_name,last\_name

John,Smith

Robert,Brown

Julia,Griffin

## Python CSV custom dialect

A custom dialect is created with the csv.register\_dialect() method.

|  |
| --- |
| **CSV07.py** |
| **Line** | **Code** |
| **1****2****3****4****5****6****7****8****9****10****11****12** | **import** csvcsv.register\_dialect("hashes", delimiter="#")f = **open**('items3.csv', 'w')with f: writer = csv.writer(f, dialect="hashes") writer.writerow(("pens", 4)) writer.writerow(("plates", 2)) writer.writerow(("bottles", 4)) writer.writerow(("cups", 1)) |

The program uses a (#) character as a delimiter. The dialect is specified with the dialect option in the csv.writer() method.

**Output:**

pens#4

plates#2

bottles#4

cups#1