**Python read csv file**

**CSV File**

A **csv** stands for "comma separated values", which is defined as a simple file format that uses specific structuring to arrange tabular data. It stores tabular data such as spreadsheet or database in plain text and has a common format for data interchange. A **csv** file opens into the excel sheet, and the rows and columns data define the standard format.

**Python CSV Module Functions**

The CSV module work is used to handle the CSV files to read/write and get data from specified columns. There are different types of CSV functions, which are as follows:

* **csv.field\_size\_limit -** It returns the current maximum field size allowed by the parser.
* **csv.get\_dialect -** It returns the dialect associated with a name.
* **csv.list\_dialects -** It returns the names of all registered dialects.
* **csv.reader -** It read the data from a csv file
* **csv.register\_dialect -** It associates dialect with a name. The name must be a string or a Unicode object.
* **csv.writer -** It writes the data to a csv file
* **o csv.unregister\_dialect -** It deletes the dialect which is associated with the name from the dialect registry. If a name is not a registered dialect name, then an error is being raised.
* **csv.QUOTE\_ALL -** It instructs the writer objects to quote all fields. csv.QUOTE\_MINIMAL - It instructs the writer objects to quote only those fields which contain special characters such as quotechar, delimiter, etc.
* **csv.QUOTE\_NONNUMERIC -** It instructs the writer objects to quote all the non-numeric fields.
* **csv.QUOTE\_NONE -** It instructs the writer object never to quote the fields.

**Reading CSV files**

Python provides various functions to read csv file. We are describing few method of reading function.

* **Using csv.reader() function**

In Python, the **csv.reader()** module is used to read the csv file. It takes each row of the file and makes a list of all the columns.

x

We have taken a txt file named as python.txt that have default delimiter **comma(,)** with the following data:

1. name,department,birthday month
2. Parker,Accounting,November
3. Smith,IT,October

**Example**

1. import csv
2. with open('python.csv') as csv\_file:
3. csv\_reader = csv.reader(csv\_file, delimiter=',')
4. line\_count = 0
5. for row in csv\_reader:
6. if line\_count == 0:
7. print(f'Column names are {", ".join(row)}')
8. line\_count += 1

**Output:**

Column names are name, department, birthday month

Parker works in the Accounting department, and was born in November.

Smith works in the IT department, and was born in October.

Processed 3 lines.

In the above code, we have opened 'python.csv' using the **open()** function. We used **csv.reader()** function to read the file, that returns an iterable reader object. The **reader** object have consisted the data and we iterated using **for** loop to print the content of each row

**Read a CSV into a Dictionar**

We can also use **DictReader()** function to read the csv file directly into a dictionary rather than deal with a list of individual string elements.

Again, our input file, python.txt is as follows:

1. name,department,birthday month
2. Parker,Accounting,November
3. Smith,IT,October

**Example**

1. import csv
2. with open('python.txt', mode='r') as csv\_file:
3. csv\_reader = csv.DictReader(csv\_file)
4. line\_count = 0
5. for row in csv\_reader:
6. if line\_count == 0:
7. print(f'The Column names are as follows {", ".join(row)}')
8. line\_count += 1
9. print(f'\t{row["name"]} works in the {row["department"]} department, and was born in {row["birthday month"]}.')
10. line\_count += 1
11. print(f'Processed {line\_count} lines.')

**Output:**

The Column names are as follows name, department, birthday month

Parker works in the Accounting department, and was born in November.

Smith works in the IT department, and was born in October.

Processed 3 lines.

**Reading csv files with Pandas**

The Pandas is defined as an open-source library which is built on the top of the NumPy library. It provides fast analysis, data cleaning, and preparation of the data for the user.

Reading the csv file into a pandas **DataFrame** is quick and straight forward. We don't need to write enough lines of code to open, analyze, and read the csv file in pandas and it stores the data in **DataFrame**.

Here, we are taking a slightly more complicated file to read, called hrdata.csv, which contains data of company employees.

1. Name,Hire Date,Salary,Leaves Remaining
2. John Idle,08/15/14,50000.00,10
3. Smith Gilliam,04/07/15,65000.00,8
4. Parker Chapman,02/21/14,45000.00,10
5. Jones Palin,10/14/13,70000.00,3
6. Terry Gilliam,07/22/14,48000.00,7
7. Michael Palin,06/28/13,66000.00,8

**Example**

1. import pandas
2. df = pandas.read\_csv('hrdata.csv')
3. print(df)

In the above code, the three lines are enough to read the file, and only one of them is doing the actual work, i.e., pandas.read\_csv()

# Python Write CSV File

## CSV File

A CSV stands for "comma-separated values", which is defined as a simple file format that uses specific structuring to arrange tabular data. It stores tabular data such as spreadsheet or database in plain text and has a standard format for data interchange. The CSV file opens into the excel sheet, and the rows and columns data define the standard format.

## Python CSV Module Functions

The CSV module work is to handle the CSV files to read/write and get data from specified columns. There are different types of CSV functions, which are as follows:

* **csv.field\_size\_limit -** It returns the current maximum field size allowed by the parser.
* **csv.get\_dialect -** Returns the dialect associated with a name.
* **csv.list\_dialects -** Returns the names of all registered dialects.
* **csv.reader -** Read the data from a CSV file
* **csv.register\_dialect -** It associates dialect with a name, and name must be a string or a Unicode object.
* **csv.writer -** Write the data to a CSV file
* **csv.unregister\_dialect -** It deletes the dialect, which is associated with the name from the dialect registry. If a name is not a registered dialect name, then an error is being raised.
* **csv.QUOTE\_ALL -** It instructs the writer objects to quote all fields.
* **csv.QUOTE\_MINIMAL -** It instructs the writer objects to quote only those fields which contain special characters such as quotechar, delimiter, etc.
* **csv.QUOTE\_NONNUMERIC -** It instructs the writer objects to quote all the non-numeric fields.
* **csv.QUOTE\_NONE -** It instructs the writer object never to quote the fields.

## Writing CSV Files

We can also write any new and existing CSV files in Python by using the csv.writer() module. It is similar to the csv.reader() module and also has two methods, i.e., **writer** function or the **Dict Writer** class.

It presents two functions, i.e., **writerow()** and **writerows()**. The **writerow()** function only write one row, and the **writerows()** function write more than one row.  
**Dialects**

It is defined as a construct that allows you to create, store, and re-use various formatting parameters. It supports several attributes; the most frequently used are:

* **Dialect.delimiter:** This attribute is used as the separating character between the fields. The default value is a comma (,).
* **Dialect.quotechar:** This attribute is used to quote fields that contain special characters.
* **Dialect.lineterminator:** It is used to create new lines, and the default value is '\r\n'.

Let's write the following data to a CSV File.

1. data = [{'Rank': 'B', 'first\_name': 'Parker', 'last\_name': 'Brian'},
2. {'Rank': 'A', 'first\_name': 'Smith', 'last\_name': 'Rodriguez'},
3. {'Rank': 'C', 'first\_name': 'Tom', 'last\_name': 'smith'},
4. {'Rank': 'B', 'first\_name': 'Jane', 'last\_name': 'Oscar'},
5. {'Rank': 'A', 'first\_name': 'Alex', 'last\_name': 'Tim'}]

### Example -

1. import csv
3. with open('Python.csv', 'w') as csvfile:
4. fieldnames = ['first\_name', 'last\_name', 'Rank']
5. writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
7. writer.writeheader()
8. writer.writerow({'Rank': 'B', 'first\_name': 'Parker', 'last\_name': 'Brian'})
9. writer.writerow({'Rank': 'A', 'first\_name': 'Smith',
10. 'last\_name': 'Rodriguez'})
11. writer.writerow({'Rank': 'B', 'first\_name': 'Jane', 'last\_name': 'Oscar'})
12. writer.writerow({'Rank': 'B', 'first\_name': 'Jane', 'last\_name': 'Loive'})
14. print("Writing complete")

**Output:**

Writing complete

It returns the file named as 'Python.csv' that contains the following data:

1. first\_name,last\_name,Rank
2. Parker,Brian,B
3. Smith,Rodriguez,A
4. Jane,Oscar,B
5. Jane,Loive,B

## Write a CSV into a Dictionary

We can also use the class **DictWriter** to write the CSV file directly into a dictionary.

A file named as python.csv contains the following data:

Parker, Accounting, November

Smith, IT, October

### Example -

1. import csv
2. with open('python.csv', mode='w') as csv\_file:
3. fieldnames = ['emp\_name', 'dept', 'birth\_month']
4. writer = csv.DictWriter(csv\_file, fieldnames=fieldnames)
5. writer.writeheader()
6. writer.writerow({'emp\_name': 'Parker', 'dept': 'Accounting', 'birth\_month': 'November'})
7. writer.writerow({'emp\_name': 'Smith', 'dept': 'IT', 'birth\_month': 'October'})

**Output:**

emp\_name,dept,birth\_month

Parker,Accounting,November

Smith,IT,October

## Writing CSV Files Using Pandas

Pandas is defined as an open source library which is built on the top of Numpy library. It provides fast analysis, data cleaning and preparation of the data for the user.

It is as easy as reading the CSV file using pandas. You need to create the DataFrame, which is a two-dimensional, heterogeneous tabular data structure and consists of three main components- data, columns, and rows. Here, we take a slightly more complicated file to read, called hrdata.csv, which contains data of company employees.

1. Name,Hire Date,Salary,Leaves Remaining
2. John Idle,08/15/14,50000.00,10
3. Smith Gilliam,04/07/15,65000.00,8
4. Parker Chapman,02/21/14,45000.00,10
5. Jones Palin,10/14/13,70000.00,3
6. Terry Gilliam,07/22/14,48000.00,7
7. Michael Palin,06/28/13,66000.00,8

### Example -

1. import pandas
2. df = pandas.read\_csv('hrdata.csv',
3. index\_col='Employee',
4. parse\_dates=['Hired'],
5. header=0,
6. names=['Employee', 'Hired', 'Salary', 'Sick Days'])
7. df.to\_csv('hrdata\_modified.csv')

**Output:**

Employee, Hired, Salary, Sick Days

John Idle, 2014-03-15, 50000.0,10

Smith Gilliam, 2015-06-01, 65000.0,8

Parker Chapman, 2014-05-12, 45000.0,10

Jones Palin, 2013-11-01, 70000.0,3

Terry Gilliam, 2014-08-12 , 48000.0,7

Michael Palin, 2013-05-23, 66000.0,8