CSV File Handling

Reading data from CSV(comma separated values) is a fundamental necessity in Data Science. Often, we get data from various sources which can get exported to CSV format so that they can be used by other systems. The Pandas library provides features using which we can read the CSV file in full as well as in parts for only a selected group of columns and rows.

The structure of a CSV file is given away by its name. Normally, CSV files use a comma to separate each specific data value. Here’s what that structure looks like:

column 1 name,column 2 name, column 3 name

first row data 1,first row data 2,first row data 3

second row data 1,second row data 2,second row data 3

In general, the separator character is called a delimiter, and the comma is not the only one used. Other popular delimiters include the tab (\t), colon (:) and semi-colon (;) characters. Properly parsing a CSV file requires us to know which delimiter is being used.

CSV files are normally created by programs that handle large amounts of data. They are a convenient way to export data from spreadsheets and databases as well as import or use it in other programs.

## Parsing CSV Files With Python’s Built-in CSV Library

The csv library provides functionality to both read from and write to CSV files. Designed to work out of the box with Excel-generated CSV files, it is easily adapted to work with a variety of CSV formats. The csv library contains objects and other code to read, write, and process data from and to CSV files.

**Reading CSV Files With csv:**

Reading from a CSV file is done using the reader object. The CSV file is opened as a text file with Python’s built-in open() function, which returns a file object. This is then passed to the reader.

**Object.reader(fileobj,delimiter)**

**Ex:**

**import csv**

**f1=open("emp.csv","r")**

**#reading a csv file: using reader()**

**csv\_reader=csv.reader(f1,delimiter=',')**

**for i in csv\_reader:**

**print(i)**

**f1.close()**

**o/p:**

**['name', 'dept', 'month']**

**['smith', 'sales', 'january']**

**['siri', 'IT', 'june']**

**['smk', 'IT', 'may']**

**(or)**

**import csv**

**f1=open("emp.csv","r")**

**#reading a csv file: using reader()**

**csv\_reader=csv.reader(f1,delimiter=',')**

**list1=[]**

**for i in csv\_reader:**

**list1.append(i)**

**print(list1)**

**f1.close()**

**o/p:**

**[['name', 'dept', 'month'], ['smith', 'sales', 'january'], ['siri', 'IT', 'june'], ['smk', 'IT', 'may']]**

**next():**

**The .next() method returns the current row and moves to the next row.**

**The first time you run next() it returns the header and the next time you run it returns the first record and so on.**

**Ex:**

**import csv**

**f1=open("emp.csv","r")**

**#reading a csv file: using reader()**

**csv\_reader=csv.reader(f1,delimiter=',')**

**titles=next(csv\_reader)**

**print(titles)**

**o/p:**

**['name', 'dept', 'month']**

**To extract the next records,**

**import csv**

**f1=open("emp.csv","r")**

**#reading a csv file: using reader()**

**csv\_reader=csv.reader(f1,delimiter=',')**

**titles=next(csv\_reader)**

**data=[]**

**for i in csv\_reader:**

**data.append(i)**

**print(data)**

**f1.close()**

## Implementing the above code using with() statement:

**import csv**

**with open("emp.csv","r") as f1:**

**csv\_reader=csv.reader(f1,delimiter=',')**

**titles=next(csv\_reader)**

**data=[]**

**for i in csv\_reader:**

**data.append(i)**

**print(titles)**

**print(data)**

**Modes**

**Modes:**

**‘r’ – to read an existing file,**

**‘w’ – to create a new file if the given file doesn’t exist and write to it,**

**‘a’ – to append to existing file content,**

**‘+’ – to create a new file for reading and writing**

**Reading all the lines of a file: readlines()**

**No ‘csv’ module needed**

**list1=[]**

**with open("emp.csv","r") as f1:**

**list1=f1.readlines()**

**print("titles----")**

**print(list1[0])**

**print("data-----")**

**print(list1[1:])**

**o/p:**

**titles----**

**name,dept,month**

**data-----**

**['smith,sales,january\n', 'siri,IT,june\n', 'smk,IT,may']**

**Using pandas**

**Load CSV files to pandas using read\_csv():**

**Ex:**

**import pandas as pd**

**data=pd.read\_csv("emp.csv")**

**print(data)**

**o/p:**

**name dept month**

**0 smith sales january**

**1 siri IT june**

**2 smk IT may**

**To Extract the field names**

**.columns is used to obtain the header/field names.**

**Ex:**

**import pandas as pd**

**data=pd.read\_csv("emp.csv")**

**print(data.columns)**

**o/p:**

**Index(['name', 'dept', 'month'], dtype='object')**

**To get specified columns,**

**data.fieldname**

**Ex:**

**import pandas as pd**

**data=pd.read\_csv("emp.csv")**

**print(data.name)**

**o/p:**

**0 smith**

**1 siri**

**2 smk**

**Name: name, dtype: object**

Writing to a CSV file

**We can write to a CSV file in multiple ways.**

1. **Using csv.writer**
2. **Using obj.writelines()**
3. **Using pandas**

**Using csv.writer:**

**1. Import csv library**

**import csv**

**2. Define a filename and Open the file using open()**

**3. Create a csvwriter object using csv.writer()**

**4. Write the header**

**5. Write the rest of the data**

**Using .writelines():**

header = ['Name', 'M1 Score', 'M2 Score']

data = [['Alex', 62, 80], ['Brad', 45, 56], ['Joey', 85, 98]]

filename = 'Student\_scores.csv'

with open(filename, 'w') as file:

for header in header:

file.write(str(header)+', ')

file.write('\n')

for row in data:

for x in row:

file.write(str(x)+', ')

file.write('n')

**o/p:**

**Name, M1 Score, M2 Score, nAlex, 62, 80, nBrad, 45, 56, nJoey, 85, 98, n**

**Using Pandas:**

**Table of Contents:**

**What is a CSV?**

**Reading a CSV**

**Writing to a CSV**

**1. What is a CSV?**

**CSV stands for “Comma Separated Values.” It is the simplest form of storing data in tabular form as plain text. It is important to know to work with CSV because we mostly rely on CSV data in our day-to-day lives as data scientists.**

**Structure of CSV:**

**We have a file named “Salary\_Data.csv.” The first line of a CSV file is the header and contains the names of the fields/features.**

**After the header, each line of the file is an observation/a record. The values of a record are separated by “comma.”**

**2. Reading a CSV**

**CSV files can be handled in multiple ways in Python.**

**2.1 Using csv.reader**

**Reading a CSV using Python’s inbuilt module called csv using csv.reader object.**

**Steps to read a CSV file:**

**1. Import the csv library**

**import csv**

**2. Open the CSV file**

**The .open() method in python is used to open files and return a file object.**

**file = open('Salary\_Data.csv')**

**type(file)**

**The type of file is “\_io.TextIOWrapper” which is a file object that is returned by the open() method.**

**3. Use the csv.reader object to read the CSV file**

**csvreader = csv.reader(file)**

**4. Extract the field names**

**Create an empty list called header. Use the next() method to obtain the header.**

**The .next() method returns the current row and moves to the next row.**

**The first time you run next() it returns the header and the next time you run it returns the first record and so on.**

**header = []**

**header = next(csvreader)**

**header**

**5. Extract the rows/records**

**Create an empty list called rows and iterate through the csvreader object and append each row to the rows list.**

**rows = []**

**for row in csvreader:**

**rows.append(row)**

**rows**

**6. Close the file**

**.close() method is used to close the opened file. Once it is closed, we cannot perform any operations on it.**

**file.close()**

**Complete Code:**

**import csv**

**file = open("Salary\_Data.csv")**

**csvreader = csv.reader(file)**

**header = next(csvreader)**

**print(header)**

**rows = []**

**for row in csvreader:**

**rows.append(row)**

**print(rows)**

**file.close()**

**Naturally, we might forget to close an open file. To avoid that we can use the with() statement to automatically release the resources. In simple terms, there is no need to call the .close() method if we are using with() statement.**

**Implementing the above code using with() statement:**

**Syntax: with open(filename, mode) as alias\_filename:**

**Modes:**

**‘r’ – to read an existing file,**

**‘w’ – to create a new file if the given file doesn’t exist and write to it,**

**‘a’ – to append to existing file content,**

**‘+’ – to create a new file for reading and writing**

**import csv**

**rows = []**

**with open("Salary\_Data.csv", 'r) as file:**

**csvreader = csv.reader(file)**

**header = next(csvreader)**

**for row in csvreader:**

**rows.append(row)**

**print(header)**

**print(rows)**

**2.2 Using .readlines()**

**Now the question is – “Is it possible to fetch the header, rows using only open() and with() statements and without the csv library?” Let’s see…**

**.readlines() method is the answer. It returns all the lines in a file as a list. Each item of the list is a row of our CSV file.**

**The first row of the file.readlines() is the header and the rest of them are the records.**

**with open('Salary\_Data.csv') as file:**

**content = file.readlines()**

**header = content[:1]**

**rows = content[1:]**

**print(header)**

**print(rows)**

**\*\*The ‘n’ from the output can be removed using .strip() method.**

**What if we have a huge dataset with hundreds of features and thousands of records. Would it be possible to handle lists??**

**Here comes the pandas library into the picture.**

**2.3 Using pandas**

**Steps of reading CSV files using pandas**

**1. Import pandas library**

**import pandas as pd**

**2. Load CSV files to pandas using read\_csv()**

**Basic Syntax: pandas.read\_csv(filename, delimiter=’,’)**

**data= pd.read\_csv("Salary\_Data.csv")**

**data**

**3. Extract the field names**

**.columns is used to obtain the header/field names.**

**data.columns**

**4. Extract the rows**

**All the data of a data frame can be accessed using the field names.**

**data.Salary**

**Writing to a CSV file**

**We can write to a CSV file in multiple ways.**

**Using csv.writer**

**Let’s assume we are recording 3 Students data(Name, M1 Score, M2 Score)**

**header = ['Name', 'M1 Score', 'M2 Score']**

**data = [['Alex', 62, 80], ['Brad', 45, 56], ['Joey', 85, 98]]**

**Steps of writing to a CSV file:**

**1. Import csv library**

**import csv**

**2. Define a filename and Open the file using open()**

**3. Create a csvwriter object using csv.writer()**

**4. Write the header**

**5. Write the rest of the data**

**code for steps 2-5**

**filename = 'Students\_Data.csv'**

**with open(filename, 'w', newline="") as file:**

**csvwriter = csv.writer(file) # 2. create a csvwriter object**

**csvwriter.writerow(header) # 4. write the header**

**csvwriter.writerows(data) # 5. write the rest of the data**

**o/p:**

**Below is how our CSV file looks.**

**Name,M1 Score,M2 Score**

**Alex,62,80**

**Brad,45,56**

**Joey,85,98**

**Using .writelines()**

**Iterate through each list and convert the list elements to a string and write to the csv file.**

**header = ['Name', 'M1 Score', 'M2 Score']**

**data = [['Alex', 62, 80], ['Brad', 45, 56], ['Joey', 85, 98]]**

**filename = 'Student\_scores.csv'**

**with open(filename, 'w') as file:**

**for header in header:**

**file.write(str(header)+', ')**

**file.write('\n')**

**for row in data:**

**for x in row:**

**file.write(str(x)+', ')**

**file.write('n')**

**Using pandas**

**Steps to writing to a CSV using pandas**

**1. Import pandas library**

**import pandas as pd**

**2. Create a pandas dataframe using pd.DataFrame**

**Syntax: pd.DataFrame(data, columns)**

**The data parameter takes the records/observations and the columns parameter takes the columns/field names.**

**header = ['Name', 'M1 Score', 'M2 Score']**

**data = [['Alex', 62, 80], ['Brad', 45, 56], ['Joey', 85, 98]]**

**data = pd.DataFrame(data, columns=header)**

**Ex:**

**import pandas as pd**

**titles=['deptid','dname','loc']**

**data=[**

**[101,'purchases','LA'],**

**[102,'sales','BPL']**

**]**

**df=pd.DataFrame(data,columns=titles)**

**print(df)**

**o/p:**

**deptid dname loc**

**0 101 purchases LA**

**1 102 sales BPL**

**3. Write to a CSV file using to\_csv()**

**Syntax: DataFrame.to\_csv(filename, sep=’,’, index=False)**

**\*\*separator is ‘,’ by default.**

**index=False to remove the index numbers.**

**data.to\_csv('Stu\_data.csv', index=False)**

**o/p:**

**Name,M1 Score,M2 Score**

**Alex,62,80**

**Brad,45,56**

**Joey,85,98**

[**https://www.analyticsvidhya.com/blog/2021/08/python-tutorial-working-with-csv-file-for-data-science/**](https://www.analyticsvidhya.com/blog/2021/08/python-tutorial-working-with-csv-file-for-data-science/)

[**https://realpython.com/python-csv/**](https://realpython.com/python-csv/)

[**https://www.tutorialspoint.com/python\_data\_science/python\_processing\_csv\_data.htm**](https://www.tutorialspoint.com/python_data_science/python_processing_csv_data.htm)