Python CSV: Read and Write CSV Files

The CSV (Comma Separated Values) format is a common and straightforward way to store tabular data. To represent a CSV file, it should have the `.csv` file extension.

Now, let's proceed with an example of the info `.csv` file and its data.

```
SN, Name,    City
1,  Michael, New Jersey
2,  Jack,    California
```

Working With CSV Files in Python

Python provides a dedicated `csv` module to work with csv files. The module includes various methods to perform different operations.

However, we first need to import the module using:

```
import csv
```

Read CSV Files with Python

The `csv` module provides the `csv.reader()` function to read a CSV file.

Suppose we have a `csv` file named `people.csv` with the following entries.

```
Name,   Age, Profession
Jack,   23,  Doctor
Miller, 22,  Engineer
```

Now, let's read this `csv` file.

```
import csv

with open('people.csv', 'r') as file:
    reader = csv.reader(file)

    for row in reader:
        print(row)
```

Output

```
['Name', 'Age', 'Profession']
['Jack', '23', 'Doctor']
['Miller', '22', 'Engineer']
```

Here, we have opened the **people.csv** file in reading mode using:

```
with open(airtravel.csv', 'r') as file:
```

We then used the `csv.reader()` function to read the file. To learn more about reading csv files, Python Reading CSV Files.

Using `csv.DictReader()` for More Readable Code

The `csv.DictReader()` class can be used to read the CSV file into a dictionary, offering a more user-friendly and accessible method.

Suppose we want to read the following `people.csv` file.

```
Name,   Age, Profession
Jack,   23,  Doctor
Miller, 22,  Engineer
```

Now let's read this file.

```
import csv
with open('people.csv', 'r') as file:
    csv_file = csv.DictReader(file)
    for row in csv_file:
        print(row)
```

Output

```
{'SN': '1', ' Name': ' Michael', ' City': ' New Jersey'}
```

```
{'SN': '2', ' Name': ' Jack', ' City': ' California'}
```

In this example, we have read data from the `people.csv` file and print each row as a dictionary.

Here, we used `csv.DictReader(file)`, which treats the first row of the CSV file as column headers and each subsequent row as a data record.

Write to CSV Files with Python

The `csv` module provides the `csv.writer()` function to write to a CSV file.

Let's look at an example.

```
import csv
with open('protagonist.csv', 'w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(["SN", "Movie", "Protagonist"])
    writer.writerow([1, "Lord of the Rings", "Frodo Baggins"])
    writer.writerow([2, "Harry Potter", "Harry Potter"])
```

When we run the above program, a **protagonist.csv** file is created with the following content:

```
SN,Movie,Protagonist
1,Lord of the Rings,Frodo Baggins
2,Harry Potter,Harry Potter
```

In this example, we have created the CSV file named `protagonist.csv` in the writing mode.

We then used the `csv.writer()` function to write to the file. To learn more about writing to a csv file, Python Writing CSV Files.

Here,

- `writer.writerow(["SN", "Movie", "Protagonist"])` writes the header row with column names to the CSV file.
- `writer.writerow([1, "Lord of the Rings", "Frodo Baggins"])` writes the first data row to the CSV file.
- `writer.writerow([2, "Harry Potter", "Harry Potter"])` writes the second data row to the CSV file.

Writing Dictionaries to CSV Files

We can use the `csv.DictWriter()` class to write dictionary data into a CSV file, which is useful for more structured data. For example,

```
import csv

with open('players.csv', 'w', newline='') as file:
    fieldnames = ['player_name', 'fide_rating']
    writer = csv.DictWriter(file, fieldnames=fieldnames)

    writer.writeheader()
    writer.writerow({'player_name': 'Magnus Carlsen', 'fide_rating': 2870})
    writer.writerow({'player_name': 'Fabiano Caruana', 'fide_rating': 2822})
    writer.writerow({'player_name': 'Ding Liren', 'fide_rating': 2801})
```

The program creates a players.csv file with the following entries:

```
player_name,fide_rating
Magnus Carlsen,2870
Fabiano Caruana,2822
Ding Liren,2801
```

In this example, we have written data to the players.csv file using `csv.DictWriter(file, fieldnames=fieldnames)`.

The `writer.writeheader()` function writes these column headers to the first row of the file.

And each call to `writerow()` adds a new row to the CSV file, where each dictionary represents a record with `player_name` and `fide_rating` as keys corresponding to the columns.

Using Python Pandas to Handle CSV Files

Pandas is a popular data science library in Python for data manipulation and analysis.

If we are working with huge chunks of data, it's better to use pandas to handle CSV files for ease and efficiency.

Note: Before we can use pandas, we need to install and import it. To learn more, visit Install and Import Pandas.

Read CSV Files

To read the CSV file using pandas, we can use the read_csv() function.

```
import pandas as pd
pd.read_csv("people.csv")
```

Here, the program reads `people.csv` from the current directory.

Write to a CSV Files

To write to a CSV file, we need to use the to_csv() function of a DataFrame.

```
import pandas as pd

# creating a data frame
df = pd.DataFrame([['Jack', 24], ['Rose', 22]], columns = ['Name', 'Age'])

# writing data frame to a CSV file
df.to_csv('person.csv')
```

Here, we have created a DataFrame using the `pd.DataFrame()` method. Then, the `to_csv()` function for this object is called, to write into `person.csv`.

Also Read

- Python File Operation

Table of Contents