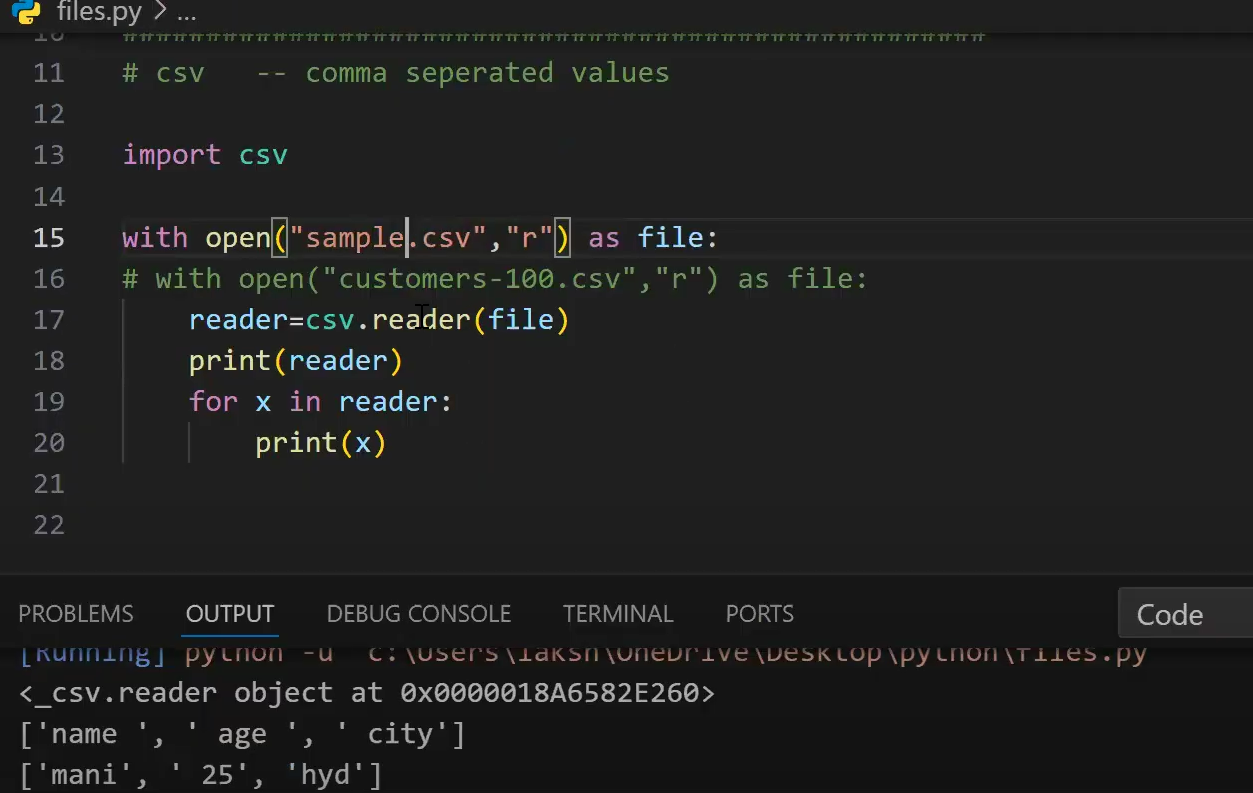
**CSV Files**

* A simple way to store big data sets is to use CSV files (comma separated files).
* CSV files contains plain text and is a well know format that can be read by everyone including Pandas.
* In Python, We can work with CSV (Comma Separated Values) files using the built-in csv module or the popular Pandas library. The csv module provides tools for reading and writing CSV files, while Pandas offers a more user-friendly approach with its read\_csv() and to\_csv() functions.



**1. Importing the csv Module**

Before working with CSV files, you need to import Python's built-in csv module.

**import csv**

**2. Reading a CSV File**

You can read a CSV file using the csv.reader() method.

**Example: Reading a CSV File**

Assume **data.csv** contains:

Name,Age,City

Alice,25,New York

Bob,30,London

Charlie,22,Sydney

**Code:**

import csv

with open("data.csv", "r") as file:

reader = csv.reader(file) # Read file

for row in reader:

print(row) # Each row is a list

✔ **Output:**

['Name', 'Age', 'City']

['Alice', '25', 'New York']

['Bob', '30', 'London']

['Charlie', '22', 'Sydney']

**Skipping Headers while Reading**

with open("data.csv", "r") as file:

reader = csv.reader(file)

**next(reader) # Skip header**

for row in reader:

print(row)

✔ **Output (Without Headers):**

['Alice', '25', 'New York']

['Bob', '30', 'London']

['Charlie', '22', 'Sydney']

**3. Writing to a CSV File**

You can write data to a CSV file using csv.writer().

**Example: Writing Data to a CSV File**

import csv

data = [

["Name", "Age", "City"],

["David", 28, "Berlin"],

["Emma", 24, "Paris"]

]

with open("output.csv", "w", newline="") as file:

writer = csv.writer(file) # Create writer object

writer.writerows(data) # Write all rows at once

✔ **Creates output.csv with:**

Name,Age,City

David,28,Berlin

Emma,24,Paris

**Writing One Row at a Time**

with open("output.csv", "w", newline="") as file:

writer = csv.writer(file)

writer.writerow(["Name", "Age", "City"]) # Write header

writer.writerow(["John", 29, "New York"]) # Write single row

**4. Appending to a CSV File**

Use "a" mode to append data **without overwriting**.

**Example: Appending Data**

with open("output.csv", "a", newline="") as file:

writer = csv.writer(file)

writer.writerow(["Sophia", 27, "Toronto"])

✔ **Adds new row to output.csv:**

Name,Age,City

John,35,Chicago

Anna,28,Los Angeles

Sophia,27,Toronto

**5. Reading CSV as Dictionary**

Instead of lists, you can read CSV files as **dictionaries** using csv.DictReader().

**Example: Using DictReader**

import csv

with open("data.csv", "r") as file:

reader = csv.DictReader(file)

for row in reader:

print(row) # Each row is an OrderedDict

✔ **Output:**

{'Name': 'Alice', 'Age': '25', 'City': 'New York'}

{'Name': 'Bob', 'Age': '30', 'City': 'London'}

{'Name': 'Charlie', 'Age': '22', 'City': 'Sydney'}

**6. Writing CSV as Dictionary**

You can write a dictionary into a CSV file using csv.DictWriter().

**Example: Using DictWriter**

import csv

data = [

{"Name": "John", "Age": 35, "City": "Chicago"},

{"Name": "Anna", "Age": 28, "City": "Los Angeles"}

]

with open("output.csv", "w", newline="") as file:

fieldnames = ["Name", "Age", "City"]

writer = csv.DictWriter(file, fieldnames=fieldnames)

writer.writeheader() # Write header

writer.writerows(data) # Write multiple rows

✔ **Creates output.csv with:**

Name,Age,City

John,35,Chicago

Anna,28,Los Angeles

**7. Handling Different Delimiters (e.g., ;, |)**

By default, CSV files use a **comma (,)** as a separator. You can change this using the delimiter argument.

**Example: Using ; as a Separator**

with open("data.csv", "r") as file:

reader = csv.reader(file, delimiter=";") # Change delimiter

for row in reader:

print(row)

✔ **For a file containing:**

Name;Age;City

Alice;25;New York

✔ **Output:**

['Name', 'Age', 'City']

['Alice', '25', 'New York']

**Example: writing a new delimeter**

import csv

data = [

    ["Name", "Age", "City"],

    ["Sophia", 27, "Toronto"],

    ["John", 30, "New York"]

]

with open("data2.csv", "w", newline="") as file:

    writer = csv.writer(file, delimiter="|")  # Use '|' as delimiter

    writer.writerows(data)

**8. Handling Quotes in CSV Files**

Sometimes, CSV fields contain **commas or quotes**. You can handle them using the quotechar argument.

**Example: Handling Quotes**

with open("data.csv", "r") as file:

reader = csv.reader(file, quotechar=' " ')

for row in reader:

print(row)

**9. Checking if a CSV File Exists Before Writing**

import os

import csv

filename = "output.csv"

if not os.path.exists(filename):

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

writer = csv.writer(file)

writer.writerow(["Name", "Age", "City"]) # Write header

This **prevents overwriting existing files**.

**10. Using pandas for CSV Files (Alternative)**

The pandas library provides a simpler way to work with CSV files.

**Reading a CSV File**

import pandas as pd

df = pd.read\_csv("data.csv")

print(df)

**Writing a CSV File**

df.to\_csv("output.csv", index=False)

✔ **This automatically handles headers, delimiters, and formatting!** 🚀

**Conclusion**

✅ csv.reader() → Reads CSV file as a **list of lists**  
✅ csv.writer() → Writes data to a CSV file  
✅ csv.DictReader() → Reads CSV file as **dictionary**  
✅ csv.DictWriter() → Writes data to a CSV file as **dictionary**  
✅ pandas → Easier CSV handling with read\_csv() and to\_csv()

Python’s csv module provides a **powerful yet simple way** to handle structured data!