Chapter 8 Reading & Writing CSV Files

Data used in a program is **temporary**; it is lost when the program terminates. To **permanently** store the data created in a program, you need to store it in a file.

Comma-separated values (CSV) is a text file format that uses commas to separate values, and newlines to separate records. A CSV file stores tabular data (numbers and text) in plain text.

CSV files are commonly used for storing tabular data. They are compatible with many tools like Excel and Google Sheets.

1. File Opening Mode

To read or write a file, we need to open it and specify the file opening mode. There are different modes to open a file:

Mode	Description
w	Write mode: Open a file for writing. If the file that does not exist, a new file will be created. If a file already exists, its content will be erased.
r	Read mode: Open a file for reading. (default). If the file does not exist, it will result in error.
a	Open a file for appending at the end of the file. Creates a new file if it does not exist. The data being written will be inserted at the end, after the existing data.

2. Writing Data to Files

When we want to write data to a file, we need to open it in write mode, "w". After we open a file, we need to create a writer object to call a writer method to write its contents. When we are done, it needs to be closed.

Example 01: Open a file to write data to it.

```
import csv     # import module

# Open a file in write mode
file = open("Employees.csv", "w", newline = "")

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

# Write contents to the file
writer.writerow(["ID", "Name", "Department"])
writer.writerow([1, "John", "DSE"])
writer.writerow([2, "Jack", "ITE"])

# Close the file
file.close() # It has no effect if the file is already closed.
```

Output

Students.csv

```
ID    Name   Department
1     John   DSE
2     Jack   ITE
```

Note that the open() function does not close the file, so you have to close the file with the close() method.

In Python, we can use the with statement to automatically close the file. The with statement ensures that the file is properly closed after the block of code is executed, even if an error occurs.

Example 02: Use with statement.

```python
import csv     # import module

# Open a file in write mode
with open("Students.csv", "w", newline = "") as file:
    # Create a writer object
    writer = csv.writer(file)

    # Write contents to the file
    writer.writerow(["ID", "Name", "Department"])
    writer.writerow([1, "John", "DSE"])
    writer.writerow([2, "Jack", "ITE"])
```

Output

Students.csv

```
ID    Name   Department
1     John   DSE
2     Jack   ITE
```

Example 03: You can open multiple files using with statement.

```python
import csv

# Open file1 and file2 in write mode
with open("File1.csv", "w", newline = "") as file1, \
     open("File2.csv", "w", newline = "") as file2:
    # Create two writer objects
    writer1 = csv.writer(file1)
    writer2 = csv.writer(file2)

    # Write contents to the files
    writer1.writerow([1, "John", "DSE"])
    writer2.writerow([2, "Jack", "ITE"])
```

Output

File1.csv

```
1    John    DSE
```

File2.csv

```
2    Jack    ITE
```

Example 04: Write data from a 2D list to a file.

```
import csv

data = [
    ["ID", "Name", "Department"],
    [1, "John", "DSE"],
    [2, "Jack", "ITE"]
]

# Open a file in write mode
with open("Students.csv", "w", newline = "") as file:

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

    # Write contents to the file
    writer.writerows(data)  # Write multiple rows
```

Output

Students.csv

```
ID      Name     Department
1       John     DSE
2       Jack     ITE
```

3. Reading Data from Files

When we want to read data from a file, we also need to open it in read mode, "r". After we open a file, we need to create a reader. This reader object can be looped through row by row where each row is a list of strings. When we are done, it needs to be closed.

Example 05: Read a file.

```
import csv

# Open a file in read mode
with open("Students.csv", "r") as file:

    # Creates a reader object
    reader = csv.reader(file)

    # Loop through each row and display it
    for row in reader:
        print(row)
```

Output

```
['ID', 'Name', 'Department']
['1', 'John', 'DSE']
['2', 'Jack', 'ITE']
```

Example 06: Skip the header

```
import csv

# Open a file in read mode
with open("Students.csv", "r") as file:

    # Creates a reader object
    reader = csv.reader(file)

    next(reader)  # Skip the header row

    # Loop through each row and display it
    for row in reader:
        print(row)
```

Output

```
['1', 'John', 'DSE']
['2', 'Jack', 'ITE']
```

Example 07: Read data in a CSV file into a list:

```python
import csv

# Open a file in read mode
with open("Students.csv", "r") as file:

     # Creates a reader object
     reader = csv.reader(file)

     next(reader)  # Skip the header row

     data = list(reader)   # convert the data into list

     print(data)
```

Output

```
[['1', 'John', 'DSE'], ['2', 'Jack', 'ITE']]
```

4. Appending to Files

When we want to append data to a file, we need to open it in append mode, "a". After we open the file, When we are done writing its content, it needs to be closed.

For example, we have a file below:

Students.csv

```
ID     Name    Department
1      John    DSE
2      Jack    ITE
```

Example 08: Append a row to a file.

```python
import csv

# Data to append
new_row = [3, "David", "FTE"]

# Open the file in append mode
with open('Students.csv', 'a', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(new_row)  # Append the new row
```

Output

Students.csv

```
ID     Name    Department
1      John    DSE
2      Jack    ITE
3      David   FTE
```

Example 08: Append multiple rows to a file.

```
import csv

# Data to append
new_rows = [
     [4, "Lucy", "DEE"],
     [5, "Sana", "DES"]
]

# Open the file in append mode
with open('Students.csv', 'a', newline='') as file:
    writer = csv.writer(file)
    writer.writerows(new_rows)  # Append the new row
```

Output

Students.csv

```
ID     Name    Department
1      John    DSE
2      Jack    ITE
3      David   FTE
4      Lucy    DEE
5      Sana    DSE
```

Example 09: Write, append and read a file.

```
import csv

# write contents to a file
with open("Students.csv", "w", newline = "") as file:
    writer = csv.writer(file)
    writer.writerow(["ID", "Name", "Department"])

# append contents to the file
with open("Students.csv", "a", newline = "") as file:
    writer = csv.writer(file)
    writer.writerow([1, "John", "DSE"])
    writer.writerow([2, "Jack", "ITE"])

# read contents from the file
with open("Students.csv", "r") as file:
    # Creates a reader object
     reader = csv.reader(file)

     # Loop through each row and display it
     for row in reader:
          print(row)
```

Output
```
['ID', 'Name', 'Department']
['1', 'John', 'DSE']
['2', 'Jack', 'ITE']
```

5. Modifying a File

We can modify the content of a file by rewriting it.

Steps:
- Read the data from the file into a list.
- Update the specific rows or fields you want.
- Write the updated data back to the file.

For example, we have a file below:

Students.csv
```
ID     Name    Department
1      John    DSE
2      Jack    ITE
```

Example 10: Modify the content of a file.
```
import csv

data = []

# Read the content of a file into a list
with open("Students.csv", "r") as file:
     # Read the data from a file
     reader = csv.reader(file)

     data = list(reader)

# Update the list, update where id = 2
for row in data:
     if row[0] == "1":
          row[1] = "Mark"
          row[2] = "CS"
          break

# Write the updated list back to the file
with open("Students.csv", "w", newline = "") as file:
    writer = csv.writer(file)
    writer.writerows(data)

print("CSV file updated successfully.")
```

Output
```
CSV file updated successfully.
```

Students.csv

```
ID     Name    Department
1      Mark    CS
2      Jack    ITE
```

5. Checking if Files Exist

You can use the **exists()** method to check if a file exists or not.

Example 11: Check if a file exists.

```
import os.path  # to use the exsits() function

file_exists1 = os.path.exists("Students.csv")
file_exists2 = os.path.exists("data/Employees.csv")
file_exists3 = os.path.exists("readme.txt")

print(file_exists1)
print(file_exists2)
print(file_exists3)
```

Output

```
True
False
False
```

Exercises

1. A file named Scores.csv contains id, name and score. Write a program to read this file, calculate grades based on the following criteria:

   If score > 85, grade = A
   If 70 <= score <= 85, grade = B
   If 50 <= score < 70, grade = C
   If score < 50, grade = F

   Save the updated data (id, name, score, and grade) into a new file named Scores_with_grades.csv.

2. A file named Products.csv contains the following columns: ProductID, ProductName, Category, and Price. Write a program to:
   - Read the data from Products.csv.
   - Filter all products with a price greater than $50 and save them into a new file named Expensive_products.csv.

3. Write an ATM machine program. Create four functions below:
    - `login()` that accepts an `account_no` and a `password` to login, and check in a file named Accounts.csv that contains `account_no`, `name`, `balance` and `password`. The function returns True if the login is successful, otherwise, returns False.
    - `display_balance()` that displays the user's balance.
    - `widthraw()` that asks the user to enter an amount to withdraw then update the Accounts.csv file if the transaction is successful.
    - `deposit()` that asks the user to enter an amount to deposit and updates the Accounts.csv file.

   For the test program, write the data (`account_no`, `name`, `balance` and `password`) of five users to a file called Accounts.csv. When the program starts, ask the user to enter an account_no and a password to login. If login succeeded, display a menu:
    a. Balance
    b. Withdraw
    c. Deposit
    d. Exit the program

   Note:
   - If the login is not successful, repeat it again; do not end the program there.
   - If the balance is not enough for withdrawing, ask the user if they want to enter another amount.

4. Create four functions below:
    - `add()` that asks the user to enter `id`, `name`, `gender` and `salary` of an employee, then save the employee data into a Employees.csv file.
    - `delete()` that removes the employee data from Employees.csv file. Hint: read the content of the store it into a variable. Remove the employee and store the content back into the file.
    - `search()` that accepts an id, searches for the student in the map. Displays the student information if found, otherwise, displays `"Search Not Found"`.
    - `display()` that displays all the employee. in a tabular format, for example:

```
---------------------------------------------------------------------
ID          Name                  Gender          Salary
---------------------------------------------------------------------
1           Lucy                  F               300
2           John                  M               400
3           Alex                  M               500
---------------------------------------------------------------------
```

   For the test program, check if the Employees.csv file exists or not. If the file does not exist, create it by adding a header that consists of ID, Name, Gender and Salary. If the file already exists, display a menu:
    a. Add a new employee
    b. Delete employee by id
    c. Search employee by id
    d. Display all employee
    e. Exit the program