File I/O

To start building the \*\*Text-Based Data Cleaning Assistant\*\*, let's break down \*\*Step 1: File Input/Output\*\*.

### \*\*Objective of Step 1:\*\*

- \*\*Read from a file\*\*: Accept a file input from the user via the command line.

- \*\*Process the file\*\*: Ensure we can load and inspect the file's contents.

- \*\*Write to a file\*\*: Output the cleaned/modified data to a new file.

We’ll first implement this functionality in a basic way for \*\*CSV\*\* files (since it’s a common format), and then we can expand it for other formats later.

---

### \*\*1. Basic File I/O for CSV Files\*\*

We’ll use Python’s built-in `csv` module for handling CSV files. This allows us to read, write, and process CSV data easily.

#### \*\*Reading a CSV File\*\*

Here’s how you can start by reading a CSV file and printing its contents:

```python

import csv

def read\_csv(file\_path):

try:

with open(file\_path, mode='r', newline='') as file:

reader = csv.reader(file)

for row in reader:

print(row) # You can process the row here later

except FileNotFoundError:

print(f"The file at {file\_path} was not found.")

except Exception as e:

print(f"An error occurred: {e}")

if \_\_name\_\_ == "\_\_main\_\_":

# Test by providing the path to a CSV file

file\_path = input("Enter the path to your CSV file: ")

read\_csv(file\_path)

```

- \*\*Explanation\*\*:

- The `csv.reader()` function reads the CSV file row by row.

- It handles cases like commas inside quotes automatically, making it easier to work with.

- We’ve added a simple `try-except` block to handle file errors (like the file not being found).

#### \*\*Writing to a CSV File\*\*

Once we’ve read and processed the data, we’ll want to write it back to a new file. Here’s how to write a CSV file:

```python

def write\_csv(file\_path, rows):

try:

with open(file\_path, mode='w', newline='') as file:

writer = csv.writer(file)

writer.writerows(rows)

print(f"Data successfully written to {file\_path}")

except Exception as e:

print(f"An error occurred: {e}")

if \_\_name\_\_ == "\_\_main\_\_":

# Example data to write (you'll replace this with processed data)

data = [

["Name", "Email", "Phone"],

["John Doe", "john@example.com", "123-456-7890"],

["Jane Smith", "jane@example.com", "098-765-4321"]

]

file\_path = input("Enter the path to save your CSV file: ")

write\_csv(file\_path, data)

```

- \*\*Explanation\*\*:

- `csv.writer()` writes the data to a file, row by row.

- `writerows()` writes multiple rows at once.

---

### \*\*2. Putting it Together: Read, Process, and Write\*\*

Let’s now combine these steps so that you can read a CSV file, process it, and write the cleaned data back out.

```python

import csv

def read\_csv(file\_path):

"""Reads the CSV file and returns a list of rows."""

rows = []

try:

with open(file\_path, mode='r', newline='') as file:

reader = csv.reader(file)

rows = list(reader) # Load all rows into memory

except FileNotFoundError:

print(f"The file at {file\_path} was not found.")

except Exception as e:

print(f"An error occurred: {e}")

return rows

def write\_csv(file\_path, rows):

"""Writes the cleaned rows to a new CSV file."""

try:

with open(file\_path, mode='w', newline='') as file:

writer = csv.writer(file)

writer.writerows(rows)

print(f"Data successfully written to {file\_path}")

except Exception as e:

print(f"An error occurred: {e}")

if \_\_name\_\_ == "\_\_main\_\_":

# Step 1: Read the input CSV

input\_path = input("Enter the path to your input CSV file: ")

data = read\_csv(input\_path)

if data:

# Step 2: Process the data (for now, we'll just print it)

print("Original Data:")

for row in data:

print(row)

# Step 3: Write the processed data to a new CSV

output\_path = input("Enter the path to save the cleaned CSV file: ")

write\_csv(output\_path, data) # Currently writing unmodified data

```

---

### \*\*Next Steps\*\*

1. \*\*Testing\*\*: You can now test the file reading and writing. Run the script and provide it with a CSV file. It should read the file and output a duplicate of it.

2. \*\*Processing Data\*\*: In the next step, we’ll start adding data processing features like removing duplicates, trimming whitespace, and more.

Would you like to move forward with implementing basic cleaning functions, or do you need help testing this first part?