**Case Study 1: Reading a Large File Efficiently**

**Scenario**: A data scientist needs to analyze a massive log file but can't load the entire file into memory.

**Solution**: Use **lazy loading** with with open() and **iterating line by line**.

with open("large\_log.txt", "r") as file:

for line in file:

process(line) # Function to analyze data

**Case Study 2: Writing a CSV File**

**Scenario**: A company wants to export customer data into a CSV file.

**Solution**: Use the csv module.

import csv

data = [("Name", "Age", "Email"), ("John", 25, "john@example.com"), ("Alice", 30, "alice@example.com")]

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

writer = csv.writer(file)

writer.writerows(data)

**Case Study 3: Appending Data to a Log File**

**Scenario**: A logging system needs to append error messages to a log file.

**Solution**: Use append mode ('a').

with open("error\_log.txt", "a") as file:

file.write("Error: Database connection failed at 10:45 AM\n")

**Case Study 4: Reading a JSON File**

**Scenario**: A web app retrieves user settings stored in a JSON file.

**Solution**: Use the json module.

import json

with open("settings.json", "r") as file:

settings = json.load(file)

print(settings)

**Case Study 5: Writing a JSON File**

**Scenario**: A Python script stores user preferences in a JSON file.

**Solution**:

settings = {"theme": "dark", "fontSize": 12}

with open("settings.json", "w") as file:

json.dump(settings, file, indent=4)

**Case Study 6: Handling File Not Found Errors**

**Scenario**: A program should gracefully handle missing files.

**Solution**: Use try-except.

try:

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

content = file.read()

except FileNotFoundError:

print("File not found. Please check the filename.")

**Case Study 7: Counting Words in a Text File**

**Scenario**: A text analysis tool needs to count words in a document.

**Solution**:

with open("document.txt", "r") as file:

words = file.read().split()

print("Word count:", len(words))

**Case Study 8: Checking if a File Exists**

**Scenario**: A program needs to verify if a file exists before reading.

**Solution**: Use os.path.exists().

import os

if os.path.exists("data.txt"):

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

print(file.read())

else:

print("File does not exist.")

**Case Study 9: Renaming a File**

**Scenario**: A script renames old reports.

**Solution**: Use os.rename().

import os

os.rename("old\_report.txt", "new\_report.txt")

**Case Study 10: Deleting a File**

**Scenario**: A cleanup script removes temporary files.

**Solution**: Use os.remove().

import os

if os.path.exists("temp.txt"):

os.remove("temp.txt")

**Case Study 11: Copying a File**

**Scenario**: A backup system copies important files.

**Solution**: Use shutil.copy().

import shutil

shutil.copy("data.txt", "backup\_data.txt")

**Case Study 12: Reading a File Line by Line**

**Scenario**: A script processes each line of a file separately.

**Solution**:

with open("log.txt", "r") as file:

for line in file:

print(line.strip())

**Case Study 13: Writing Binary Data to a File**

**Scenario**: A program stores an image in binary format.

**Solution**:

with open("image.jpg", "rb") as file:

binary\_data = file.read()

with open("copy.jpg", "wb") as file:

file.write(binary\_data)

**Case Study 14: Reading a CSV File**

**Scenario**: A script processes sales data from a CSV file.

**Solution**:

import csv

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

reader = csv.reader(file)

for row in reader:

print(row)

**Case Study 15: Using Context Manager for File Handling**

**Scenario**: Ensure files are closed properly after reading/writing.

**Solution**:

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

print(file.read()) # File closes automatically after execution

**Case Study 16: Reading a Compressed File**

**Scenario**: A script reads a .gz compressed log file.

**Solution**:

import gzip

with gzip.open("log.gz", "rt") as file:

print(file.read())

**Case Study 17: Counting Lines in a File**

**Scenario**: A program counts the number of lines in a document.

**Solution**:

with open("document.txt", "r") as file:

line\_count = sum(1 for line in file)

print("Total lines:", line\_count)

**Case Study 18: Storing Python Objects in a File (Pickle)**

**Scenario**: A script saves and loads Python objects.

**Solution**: Use pickle.

import pickle

data = {"name": "Alice", "age": 25}

with open("data.pkl", "wb") as file:

pickle.dump(data, file)

with open("data.pkl", "rb") as file:

loaded\_data = pickle.load(file)

print(loaded\_data)

**Case Study 19: Reading Specific Columns from a CSV**

**Scenario**: A script extracts only required data from a CSV file.

**Solution**:

import csv

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

reader = csv.DictReader(file)

for row in reader:

print(row["Name"], row["Department"])

**Case Study 20: Monitoring a File for Changes**

**Scenario**: A script watches a log file for updates.

**Solution**:

import time

filename = "server.log"

with open(filename, "r") as file:

file.seek(0, 2) # Move to end of file

while True:

line = file.readline()

if line:

print(line.strip())

time.sleep(1)