🐍 Python Day 4 Notes - Advanced File Handling, Requests, OOPs & Database

Theme: Making Python Work with Files, Web, and Real-World Applications

---

📚 Table of Contents

---

1. Advanced File Handling

🔍 What are File Modes?

Different ways to open and work with files.

File Modes Table:

Mode	What It Does	Example Use
`r`	Read only	Reading a book
`w`	Write (overwrite)	Writing a new letter
`x`	Create new file	Making a new diary
`a`	Append (add to end)	Adding to a diary
`r+`	Read + Write	Edit existing document
`w+`	Write + Read	Create and edit
`a+`	Append + Read	Add and check
`b`	Binary mode	For images, videos
`t`	Text mode (default)	For text files

Example 1: Write and Overwrite

python

```python
# Creates new file or overwrites existing
f = open('day_4.txt', mode='w')
f.write('This is Day 4 of Python training!')
f.close()
print('File created successfully!')
```

Example 2: Create New File Only

python

```python
# Creates file only if it doesn't exist
try:
    file = open('new_file.txt', mode='x')
    file.write('New file created!')
    file.close()
    print('New file created!')
except FileExistsError:
    print('File already exists!')
```

Example 3: Append to File

python

```python
# Adds content to the end of file
f = open('day_4.txt', mode='a')
f.write('\nNew line added!')
f.close()
print('Content added!')
```

Example 4: Read and Write Together

python

```python
# Can read existing content and add new content
f = open('day_4.txt', 'r+')
print("Current content:", f.read())
f.write('\nAdding more lines!')
f.close()
```

Example 5: Copy a Text File

python

```python
# Read from one file and write to another
f1 = open('day_4.txt', 'r')
data = f1.read()
f1.close()

f2 = open('copy_day_4.txt', 'w')
f2.write(data)
f2.close()
print('File copied successfully!')
```

Binary Files (Images, Videos, Audio)

python

```python
# Copy an image file
with open('photo.jpg', 'rb') as original:
    data = original.read()

with open('photo_copy.jpg', 'wb') as copy:
    copy.write(data)
print('Image copied!')
```

🔑 Key Points:

- **Text mode:** For `.txt`, `.py`, `.csv` files
- **Binary mode:** For `.jpg`, `.mp3`, `.pdf` files
- Always close files or use `with` statement

---

2. Requests Module

🔍 What is Requests Module?

A powerful tool to download data from the internet.

Installation:

bash

```bash
pip install requests
```

Example 1: Download an Image

python

```
import requests

url = 'https://picsum.photos/300/200'
response = requests.get(url)

with open('random_image.jpg', 'wb') as f:
    f.write(response.content)
print('Image downloaded!')
```

Example 2: Get Website Data

python

```
import requests

url = 'https://api.github.com/users/octocat'
response = requests.get(url)

if response.status_code == 200:
    data = response.json()
    print(f"User: {data['name']}")
    print(f"Followers: {data['followers']}")
else:
    print("Failed to get data")
```

Example 3: Save Search Results

python

```
import requests
from googlesearch import search

# Search for something
topic = 'Python programming tutorial'
results = search(topic, num_results=10)

# Save results to file
with open(f'{topic}_results.txt', 'w') as file:
    for i, link in enumerate(results, 1):
        file.write(f"{i}. {link}\n")
print('Search results saved!')
```

💡 Real-Life Applications:

- Download files from internet
- Get weather data

- Fetch news articles
- API integration

---

3. Object-Oriented Programming (OOPs)

🔍 What is OOPs?

A way to organize code using classes and objects, like real-world entities.

4 Pillars of OOPs:

Pillar	Meaning	Example
Encapsulation	Bundle data and methods together	Car has engine + drive method
Inheritance	Child class inherits from parent	Student inherits from Person
Polymorphism	Same method, different behavior	`+` works for numbers and strings
Abstraction	Hide complex details	TV remote hides internal circuits

Example 1: Simple Class

python

```python
class Student:
    def __init__(self, name, age):
        self.name = name  # Attribute
        self.age = age    # Attribute

    def introduce(self):  # Method
        print(f"Hi, I'm {self.name} and I'm {self.age} years old")

    def study(self, subject):
        print(f"{self.name} is studying {subject}")

# Create objects
student1 = Student('Rahul', 20)
student2 = Student('Priya', 19)

student1.introduce()
student1.study('Python')
```

Example 2: Inheritance

python

```python
class Person:
    def __init__(self, name):
        self.name = name

    def speak(self):
        print(f"{self.name} is speaking")

class Teacher(Person):  # Teacher inherits from Person
    def teach(self, subject):
        print(f"{self.name} is teaching {subject}")

class Student(Person):  # Student inherits from Person
    def study(self, subject):
        print(f"{self.name} is studying {subject}")

# Create objects
teacher = Teacher('Ms. Sharma')
student = Student('Raj')

teacher.speak()  # Inherited method
teacher.teach('Mathematics')

student.speak()  # Inherited method
student.study('Science')
```

Example 3: Polymorphism

python

```
# Same function name, different behavior
print(len('Hello'))     # Length of string = 5
print(len([1, 2, 3]))   # Length of list = 3
print(len({'a': 1}))    # Length of dictionary = 1

# Same method name in different classes
class Dog:
    def make_sound(self):
        print("Woof!")

class Cat:
    def make_sound(self):
        print("Meow!")

dog = Dog()
cat = Cat()

dog.make_sound()  # Woof!
cat.make_sound()  # Meow!
```

🔑 Key Points:

- **Class:** Blueprint for creating objects
- **Object:** Instance of a class
- **Method:** Function inside a class
- **Attribute:** Variable inside a class

---

4. Database Connectivity

🔍 What is Database Connectivity?

Connecting Python to databases to store and retrieve data permanently.

Popular Databases:

- **SQLite:** Built-in, good for learning
- **MySQL:** Popular for web applications
- **PostgreSQL:** Advanced features
- **MongoDB:** For document storage

SQLite Example (Built-in):

python

```python
import sqlite3

# Connect to database (creates if doesn't exist)
conn = sqlite3.connect('students.db')
cursor = conn.cursor()

# Create table
cursor.execute('''
    CREATE TABLE IF NOT EXISTS students (
        id INTEGER PRIMARY KEY,
        name TEXT NOT NULL,
        age INTEGER,
        grade TEXT
    )
''')

# Insert data
cursor.execute("INSERT INTO students (name, age, grade) VALUES (?, ?, ?)",
               ("Rahul", 20, "A"))
cursor.execute("INSERT INTO students (name, age, grade) VALUES (?, ?, ?)",
               ("Priya", 19, "B"))

# Save changes
conn.commit()

# Read data
cursor.execute("SELECT * FROM students")
rows = cursor.fetchall()

print("Students in database:")
for row in rows:
    print(f"ID: {row[0]}, Name: {row[1]}, Age: {row[2]}, Grade: {row[3]}")

# Close connection
conn.close()
```

MySQL Example:

python

```python
import mysql.connector

try:
    # Connect to MySQL
    conn = mysql.connector.connect(
        host='localhost',
        user='root',
        password='your_password',
        database='school'
    )

    cursor = conn.cursor()

    # Execute query
    cursor.execute("SELECT * FROM students")

    # Fetch results
    for row in cursor.fetchall():
        print(row)

except mysql.connector.Error as err:
    print(f"Error: {err}")

finally:
    if conn.is_connected():
        cursor.close()
        conn.close()
```

💡 Real-Life Applications:

- User login systems
- E-commerce product storage
- Banking transactions
- Social media posts

---

5. Mini Assignments

📝 Practice Problems:

1. **Government Jobs Scraper**
    - Search for "Government Jobs 2025" on Google
    - Save top 10 results in a text file
    - Use requests and file handling

2. **Image Downloader**
    - Download 5 random images from https://picsum.photos
    - Save them as image_1.jpg to image_5.jpg
    - Use a loop and requests
3. **Student Management System**
    - Create a Student class with name, roll_no, marks
    - Add methods to calculate grade
    - Save student data to a file
4. **File Backup System**
    - Create a program to backup all .py files
    - Copy them to a 'backup' folder
    - Use file handling and os module
5. **Simple Database App**
    - Create a SQLite database for books
    - Add, view, and search books
    - Include title, author, and year

---

6. Key Takeaways

🎯 Important Points:

1. **File Modes Give You Power**
    - `w` for new files (overwrites)
    - `a` for adding to existing files
    - `r+` for reading and writing
    - `x` for creating new files only
2. **Requests Opens the Internet**
    - Download files from web
    - Get data from APIs
    - Automate web tasks
3. **OOPs Organizes Your Code**
    - Classes are blueprints
    - Objects are real instances
    - Inheritance saves code repetition
    - Polymorphism allows flexibility

4. **Databases Store Data Forever**
    - SQLite is great for learning
    - MySQL for web applications
    - Always close connections

🔧 Best Practices:

- Always close files and database connections
- Use `with` statement for automatic cleanup
- Handle exceptions for file and network operations
- Keep classes simple and focused
- Use meaningful names for classes and methods

💡 Real-World Projects:

- **File Manager:** Copy, move, backup files
- **Web Scraper:** Download images and data
- **Library System:** Manage books with database
- **Student Portal:** Track grades and attendance

📖 Study Plan

Today's Focus:

- Practice different file modes
- Download files using requests
- Create simple classes
- Connect to SQLite database

Tomorrow's Preview:

- Advanced OOPs concepts
- GUI programming with Tkinter
- Web scraping techniques

🎯 Quick Review Questions:

1. What's the difference between `w` and `a` file modes?
2. How do you download an image using requests?
3. What are the 4 pillars of OOPs?

4. Why do we need database connectivity?
5. What's the difference between a class and an object?

Remember: The best way to learn is by building real projects! 🚀

Start small, think big, and keep coding! 💻