A black background with white text

AI-generated content may be incorrect.

Name: Shoaib Ahmed

Intern ID: TN/IN02/PY/027

Task no: week 5 task

Internship domain: python language

Date: 26 August 2025

**Task 1 Threads:**

Download 5 URLs with threads and Measure total time they take to save in file

**Code:**

import threading

import requests

import time

*# List of URLs*

urls = [

    "https://www.example.com",

    "https://www.python.org",

    "https://www.wikipedia.org",

    "https://www.github.com",

    "https://www.stackoverflow.com"

]

*# File to save data*

output\_file = "downloaded\_pages.txt"

*# Function to download and save*

*def* download\_url(*url*, *file\_handle*):

    try:

        response = requests.get(url)

        file\_handle.write(*f*"URL: {url}\n")

        file\_handle.write(response.text[:500])  *# first 500 chars only for demo*

        file\_handle.write("\n\n" + "="\*50 + "\n\n")

    except *Exception* as e:

        file\_handle.write(*f*"Failed to download {url}: {*str*(e)}\n")

*# Main*

start\_time = time.time()

*# Open file in write mode*

with open(output\_file, "w", *encoding*="utf-8") as f:

    threads = []

    for url in urls:

        t = threading.Thread(*target*=download\_url, *args*=(url, f))

        threads.append(t)

        t.start()

*# Wait for all threads*

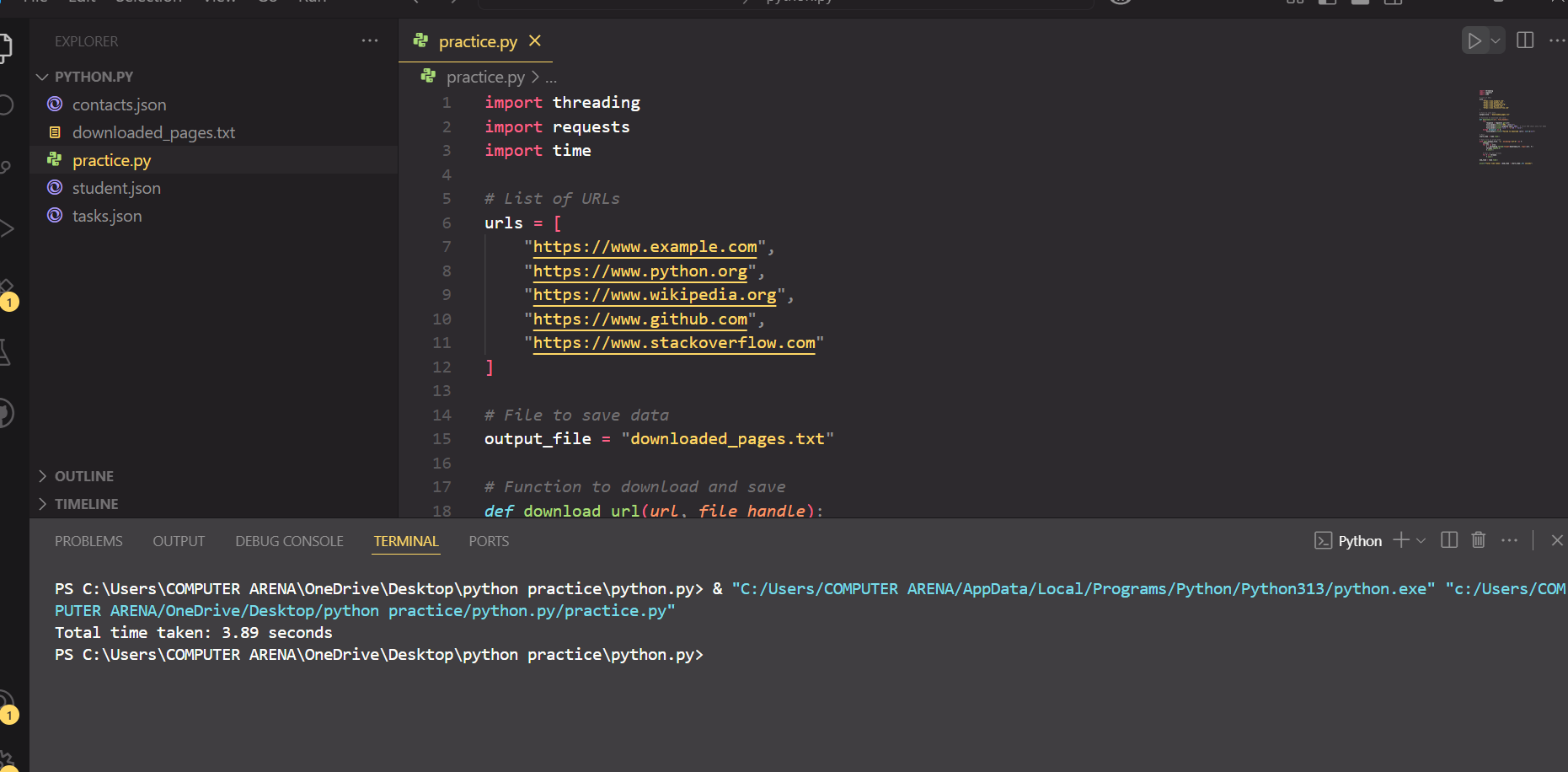
    for t in threads:

        t.join()

end\_time = time.time()

print(*f*"Total time taken: {end\_time - start\_time*:.2f*} seconds")

**Output:**

****

**Task 2: processes:**

1. Square large list with Pool and multiprocesses.

**Code:**

import os

import time

from multiprocessing import Pool, cpu\_count, get\_start\_method

*# --- Pure function for workers ---*

*def* square(*x*: *int*) -> *int*:

    return x \* x

*def* main():

*# Large input list*

    N = 5\_000\_000         *# 5 million items (adjust if RAM low)*

    data = *list*(range(N)) *# e.g., [0,1,2,...]*

*# Choose number of worker processes (all cores by default)*

    processes = cpu\_count()   *# or set manually, e.g., processes = 4*

    print(*f*"Using processes: {processes} (start method: {get\_start\_method()})")

*# Chunk size helps performance on big lists*

*# Rule of thumb: len(data) // (processes \* 8) (at least 1)*

    chunksize = max(1, len(data) // (processes \* 8))

    t0 = time.perf\_counter()

    with Pool(*processes*=processes) as pool:

*# Option A: preserve order (map)*

        result = pool.map(square, data, *chunksize*=chunksize)

*# Option B (faster sometimes): unordered*

*# result = list(pool.imap\_unordered(square, data, chunksize=chunksize))*

    t1 = time.perf\_counter()

    print(*f*"Computed {len(result)} squares in {t1 - t0*:.2f*} seconds")

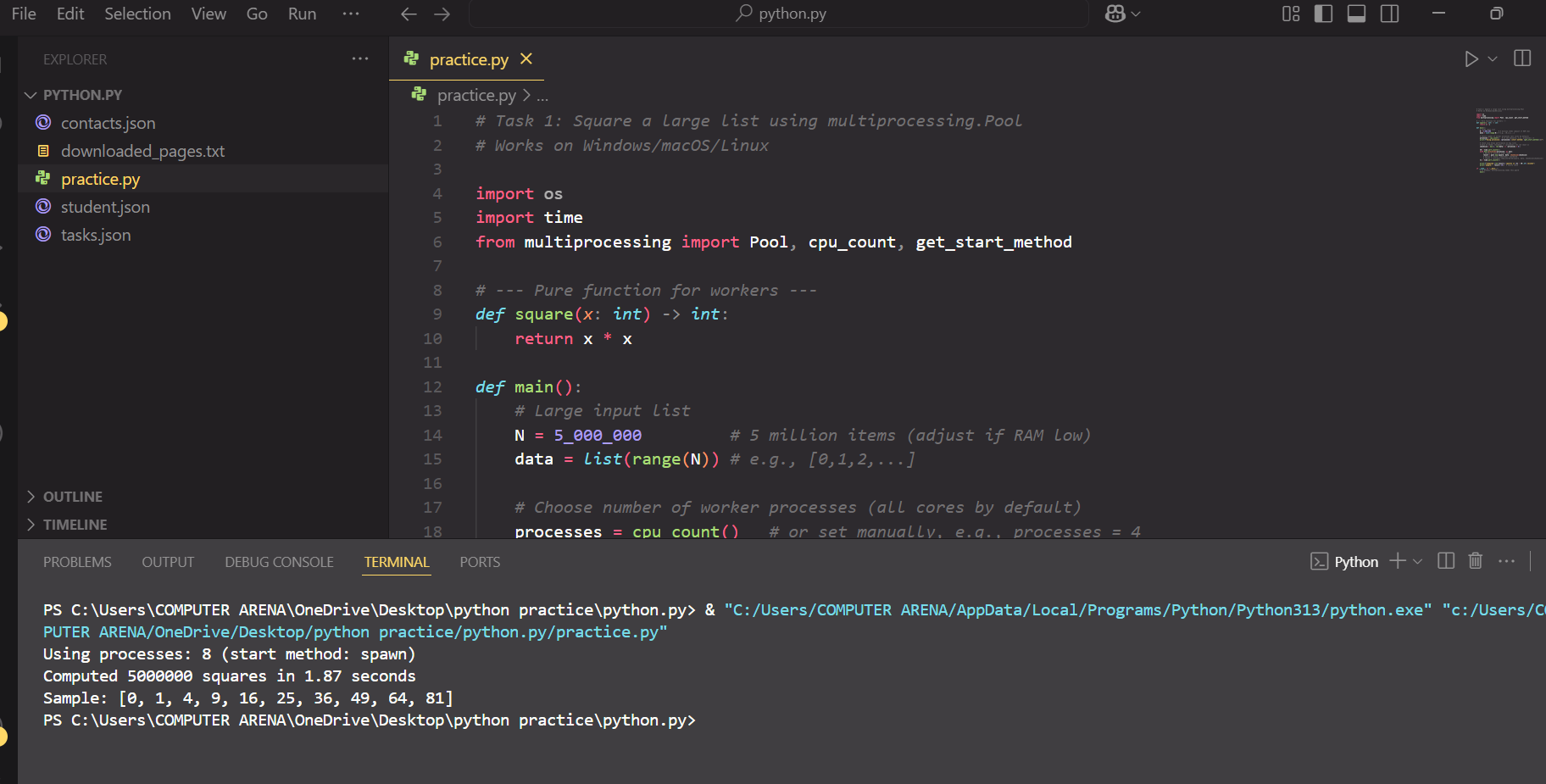
    print("Sample:", result[:10])  *# sanity check*

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

*# On Windows, multiprocessing needs this guard*

    main()

**Output:**

****

**Task 3: datetime:**

Compute days until your birthday.

**Code:**

def days\_until\_birthday(birthday\_month: int, birthday\_day: int) -> int:

    today = date.today()

    current\_year = today.year

    # This year's birthday

    next\_birthday = date(current\_year, birthday\_month, birthday\_day)

    # Agar birthday is saal guzar gaya hai, to next year ka le lo

    if next\_birthday < today:

        next\_birthday = date(current\_year + 1, birthday\_month, birthday\_day)

    delta = next\_birthday - today

    return delta.days

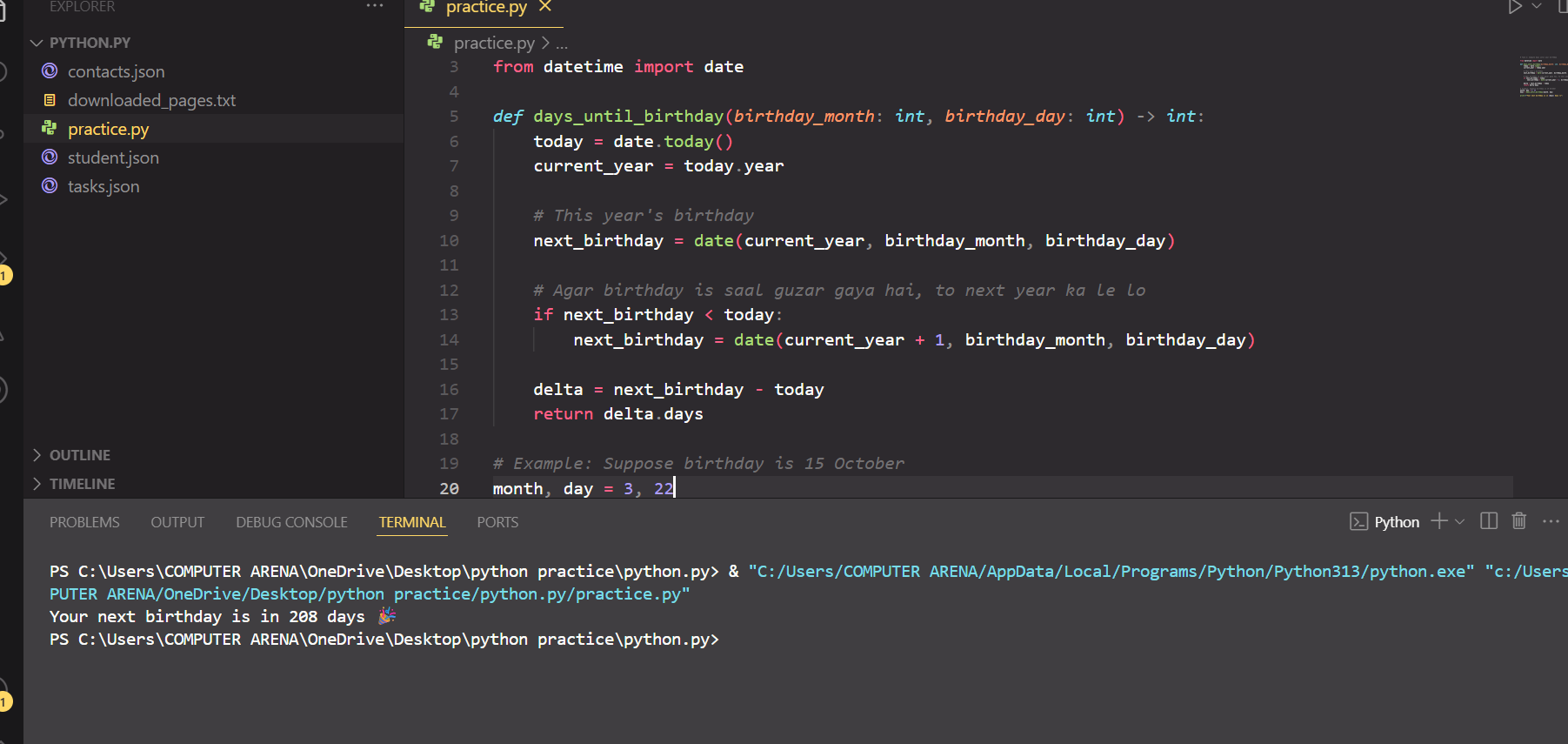
# Example: Suppose birthday is 15 October

month, day = 3,

days = days\_until\_birthday(month, day)

print(f"Your next birthday is in {days} days 🎉")

**Output:**

****

**Task 4: flask basics:**

1. Add /about route in flask app to return Hello About in page.

**Code:**

app = Flask(\_\_name\_\_)

@app.route("/")

*def* home():

    return "Hello Home"

@app.route("/about")

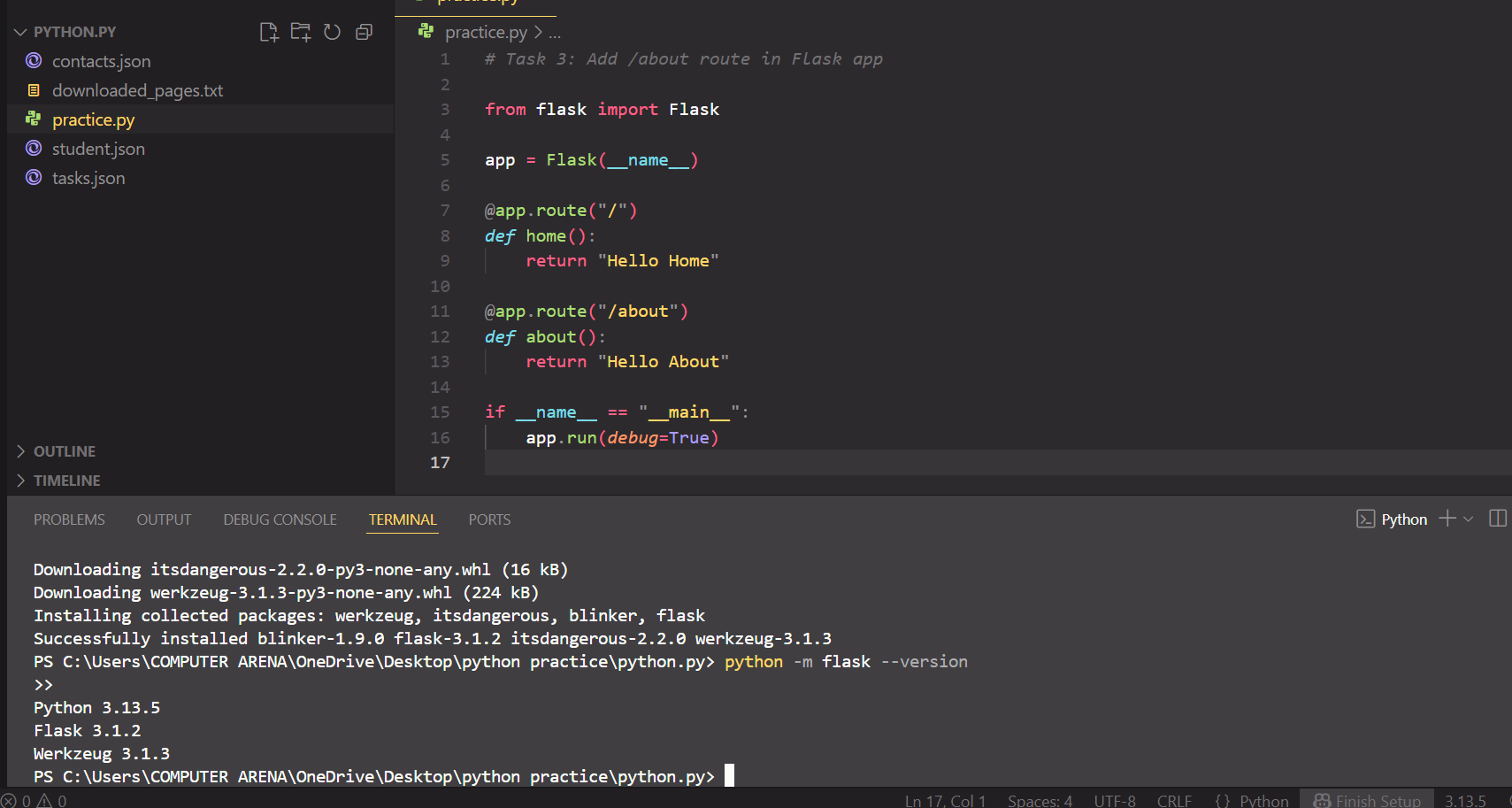
*def* about():

    return "Hello About"

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

    app.run(*debug*=True)

**Output:**

****

Here I install the new version of flask :

**Task 5 Mongo DB:**

**1. Insert 3 users in Data Base and fetch them to print in screen**

**Code:**

import sqlite3

# Step 1: Connect to database (file 'users.db' banega)

conn = sqlite3.connect("users.db")

cursor = conn.cursor()

# Step 2: Create table if not exists

cursor.execute("""

CREATE TABLE IF NOT EXISTS users (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT,

email TEXT

)

""")

# Step 3: Insert 3 users

users = [

("Ali", "ali@example.com"),

("khan", "khan@example.com"),

("abdullah", "abdullah@example.com")

]

cursor.executemany("INSERT INTO users (name, email) VALUES (?, ?)", users)

conn.commit()

# Step 4: Fetch all users

cursor.execute("SELECT \* FROM users")

rows = cursor.fetchall()

print("Users in Database:")

for row in rows:

print(row)

conn.close()

**Output:**

