Name: Saad Bin Haroon:

Intern ID: TN/IN02/PY/026:

Task no: week 5 task:

Internship domain: python language:

Date: 26 August 2025:

Task 1 Threads:

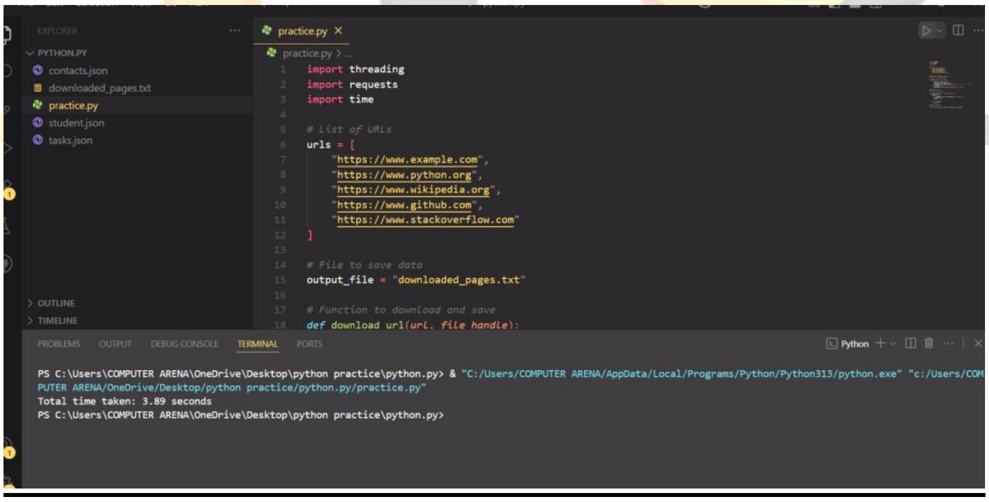
. Download 5 URLs with threads and Measure total time they take to save in file

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)
```

```python
        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:



```python
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):
```

```
PS C:\Users\COMPUTER ARENA\OneDrive\Desktop\python practice\python.py> & "C:/Users/COMPUTER ARENA/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users/COM
PUTER ARENA/OneDrive/Desktop/python practice/python.py/practice.py"
Total time taken: 3.89 seconds
PS C:\Users\COMPUTER ARENA\OneDrive\Desktop\python practice\python.py>
```

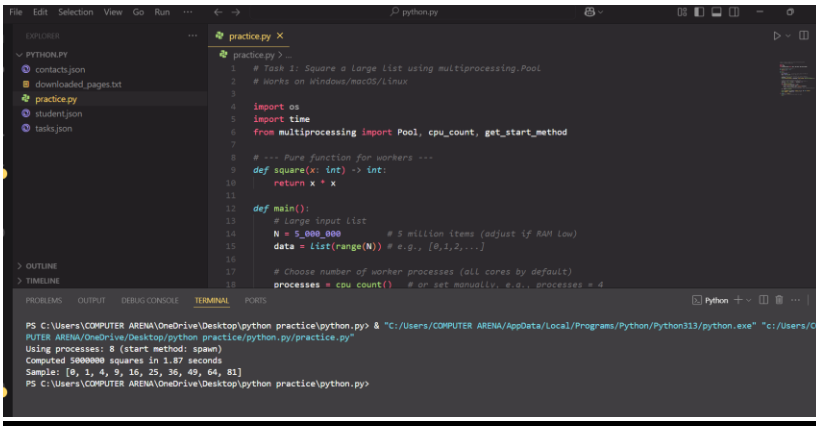
Task 2: processes:

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:



```python
# Task 1: Square a large list using multiprocessing.Pool
# Works on Windows/macOS/Linux

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
```

```
PS C:\Users\COMPUTER ARENA\OneDrive\Desktop\python practice\python.py> & "C:/Users/COMPUTER ARENA/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users/CO
PUTER ARENA/OneDrive/Desktop/python practice/python.py/practice.py"
Using processes: 8 (start method: spawn)
Computed 5000000 squares in 1.87 seconds
Sample: [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
PS C:\Users\COMPUTER ARENA\OneDrive\Desktop\python practice\python.py>
```

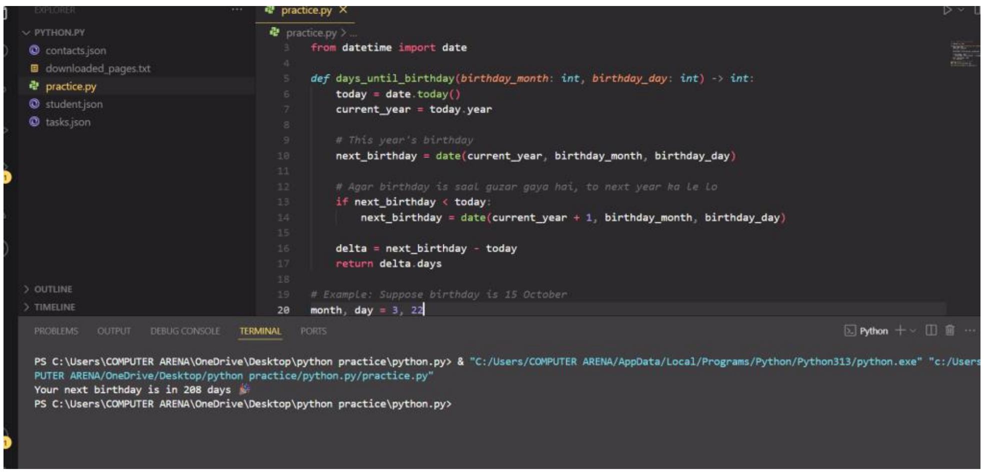
Task 3: datetime:

Compute days until your birthday.

Code:

```python
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:



```python
from datetime import date

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, 22
```

```
PS C:\Users\COMPUTER ARENA\OneDrive\Desktop\python practice\python.py> & "C:/Users/COMPUTER ARENA/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users
PUTER ARENA/OneDrive/Desktop/python practice/python.py/practice.py"
Your next birthday is in 208 days 🎉
PS C:\Users\COMPUTER ARENA\OneDrive\Desktop\python practice\python.py>
```

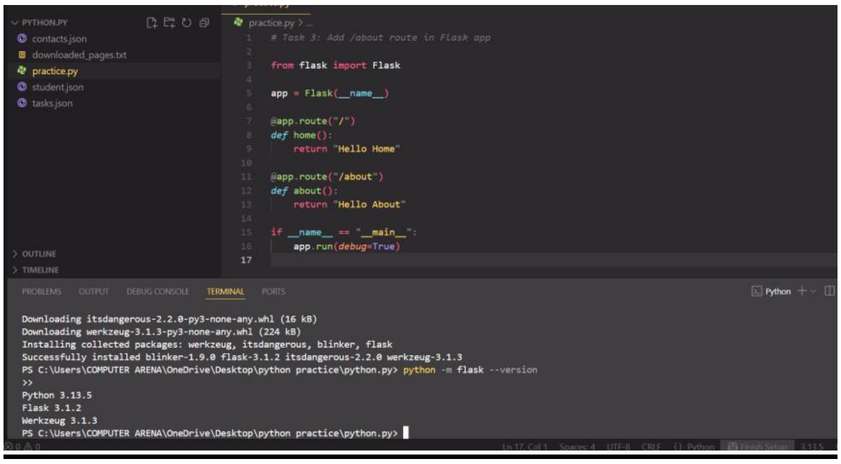
Task 4: flask basics:

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

Code:

```python
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:



```python
# Task 3: Add /about route in Flask app

from flask import Flask

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)
```

```
Downloading itsdangerous-2.2.0-py3-none-any.whl (16 kB)
Downloading werkzeug-3.1.3-py3-none-any.whl (224 kB)
Installing collected packages: werkzeug, itsdangerous, blinker, flask
Successfully installed blinker-1.9.0 flask-3.1.2 itsdangerous-2.2.0 werkzeug-3.1.3
PS C:\Users\COMPUTER ARENA\OneDrive\Desktop\python practice\python.py> python -m flask --version
>>
Python 3.13.5
Flask 3.1.2
Werkzeug 3.1.3
PS C:\Users\COMPUTER ARENA\OneDrive\Desktop\python practice\python.py>
```

Here I install the new version of flask :

Task 5 Mongo DB:

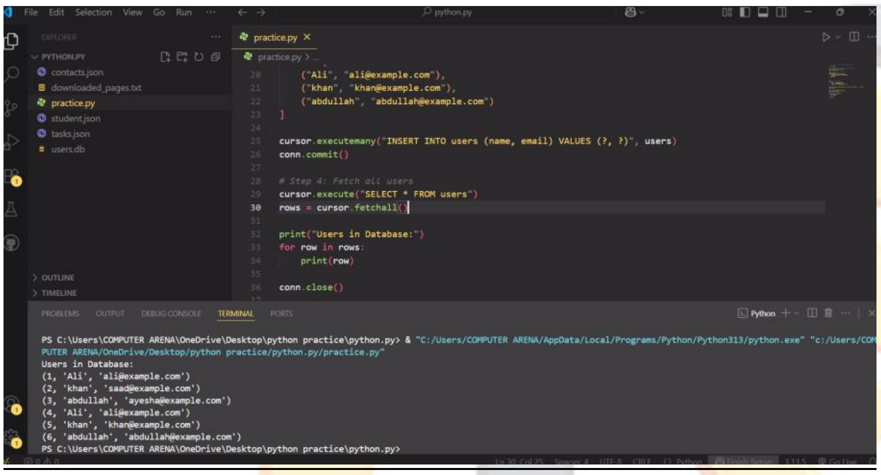
1. Insert 3 users in Data Base and fetch them to print in screen

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:



```
    ("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()
```

```
PS C:\Users\COMPUTER ARENA\OneDrive\Desktop\python practice\python.py> & "C:/Users/COMPUTER ARENA/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users/COM
PUTER ARENA/OneDrive/Desktop/python practice/python.py/practice.py"
Users in Database:
(1, 'Ali', 'ali@example.com')
(2, 'khan', 'saad@example.com')
(3, 'abdullah', 'ayesha@example.com')
(4, 'Ali', 'ali@example.com')
(5, 'khan', 'khan@example.com')
(6, 'abdullah', 'abdullah@example.com')
PS C:\Users\COMPUTER ARENA\OneDrive\Desktop\python practice\python.py>
```