

Report for internship task 5:

Name: Saad bin haroon

Internship iD: TN/IN02/PY/026:

GitHub link: <https://github.com/Saad-Bin-Haroon/internship-task-5.git>

Task Objective:

The objective of this task is for threads, processes, mongodb

Code snippets:

Task 4: Insert 3 users in Database and fetch them

```
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"),
```

```
    ("Saad", "saad@example.com"),
```

```
    ("Ayesha", "ayesha@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()
```

2: # 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)
```

screenshots:

The image displays three sequential screenshots of a Visual Studio Code editor window, illustrating a Python development workflow.

Top Screenshot: The editor shows a file named `practice.py` with the following code:

```
20     ("Ali", "ali@example.com"),
21     ("khan", "khan@example.com"),
22     ("abdullah", "abdullah@example.com")
23 ]
24
25 cursor.executemany("INSERT INTO users (name, email) VALUES (?, ?)", users)
26 conn.commit()
27
28 # Step 4: Fetch all users
29 cursor.execute("SELECT * FROM users")
30 rows = cursor.fetchall()
31
32 print("Users in Database:")
33 for row in rows:
34     print(row)
35
36 conn.close()
```

The terminal output shows the execution of the script, displaying the list of users in the database:

```
PS C:\Users\COMPUTER ARENA\OneDrive\Desktop\python practice\python.py & "C:/Users/COMPUTER ARENA/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users/COMPUTER 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')
```

Middle Screenshot: The editor shows the same `practice.py` file with updated code for a Flask application:

```
1 # Task 3: Add /about route in Flask app
2
3 from flask import Flask
4
5 app = Flask(__name__)
6
7 @app.route("/")
8 def home():
9     return "Hello Home"
10
11 @app.route("/about")
12 def about():
13     return "Hello About"
14
15 if __name__ == "__main__":
16     app.run(debug=True)
```

The terminal output shows the installation of required packages and the successful execution of the Flask application:

```
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>
```

Bottom Screenshot: The editor shows the `practice.py` file with code for a birthday calculator:

```
3 from datetime import date
4
5 def days_until_birthday(birthday_month: int, birthday_day: int) -> int:
6     today = date.today()
7     current_year = today.year
8
9     # This year's birthday
10    next_birthday = date(current_year, birthday_month, birthday_day)
11
12    # Agar birthday is saal guzar gaya hai, to next year ka le lo
13    if next_birthday < today:
14        next_birthday = date(current_year + 1, birthday_month, birthday_day)
15
16    delta = next_birthday - today
17    return delta.days
18
19 # Example: Suppose birthday is 15 October
20 month, day = 10, 15
```

The terminal output shows the execution of the script, displaying the result of the birthday calculation:

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

The image displays two screenshots of a Visual Studio Code editor. The top screenshot shows a Python script named 'practice.py' using the multiprocessing module. The code defines a 'square' function and a 'main' function that generates a large list of numbers, then uses a Pool of 8 processes to compute the squares of these numbers. The terminal output shows that 5,000,000 squares were computed in 1.87 seconds. The bottom screenshot shows another 'practice.py' script using the threading module. This script defines a list of URLs and a 'download' function that fetches the content of each URL and saves it to a file named 'downloaded_pages.txt'. The terminal output indicates that the total time taken for these operations was 3.89 seconds.

```
1 # Task 1: Square a large list using multiprocessing.Pool
2 # Works on windows/macOS/linux
3
4 import os
5 import time
6 from multiprocessing import Pool, cpu_count, get_start_method
7
8 # --- Pure function for workers ---
9 def square(x: int) -> int:
10     return x * x
11
12 def main():
13     # Large input list
14     N = 5_000_000 # 5 million items (adjust if RAM low)
15     data = list(range(N)) # e.g., [0,1,2,...]
16
17     # Choose number of worker processes (all cores by default)
18     processes = cpu_count() # or set manually, e.g., processes = 4
19
20     with Pool(processes) as pool:
21         pool.map(square, data)
22
23 if __name__ == '__main__':
24     main()
```

```
1 import threading
2 import requests
3 import time
4
5 # List of URLs
6 urls = [
7     "https://www.example.com",
8     "https://www.python.org",
9     "https://www.wikipedia.org",
10    "https://www.github.com",
11    "https://www.stackoverflow.com"
12]
13
14 # File to save data
15 output_file = "downloaded_pages.txt"
16
17 # Function to download and save
18 def download_url(url, file_handle):
19     start = time.time()
20     response = requests.get(url)
21     file_handle.write(response.text)
22     end = time.time()
23     print(f"Downloaded {url} in {end - start} seconds")
24
25 if __name__ == '__main__':
26     with open(output_file, 'w') as file:
27         threads = []
28         for url in urls:
29             thread = threading.Thread(target=download_url, args=(url, file))
30             threads.append(thread)
31             thread.start()
32         for thread in threads:
33             thread.join()
```

Reflections:

This task helped me learn how to use Python with a database. I created a table, added three users, and then fetched them to show on the screen. By doing this, I understood how data can be stored and read using simple SQL commands. It was a useful exercise because it gave me basic knowledge of database handling, which is important for building real applications.

 End of Report: