Name: Hassan Ali

Intern ID: TN/IN02/PY/012

Email ID: hassanali2127294@gmail.com

Internship Domain: Python Development

Instructor Name: Mr. Hassan Ali

Task: Week 06

Task 1 – Threads

Question: Download 5 URLs with threads and measure total time they take to save in file.

Code:

```
import threading, time, random, os

def fake_download(url, out_dir, idx, results):
    latency = random.uniform(0.25, 0.6)  # simulate network delay
    time.sleep(latency)
    content = f"FAKE CONTENT for {url}\nDownloaded in {latency:.3f} s\n"
    fname = os.path.join(out_dir, f"file_{idx+1}.txt")
    with open(fname, "w", encoding="utf-8") as f:
        f.write(content)
    results[idx] = {"url": url, "file": fname, "latency_sec": round(latency, 3)}

urls = ['https://example.com/a', 'https://example.com/b', 'https://example.com/c',
'https://example.com/d', 'https://example.com/e']
out_dir = r"/mnt/data/downloads"
os.makedirs(out_dir, exist_ok=True)

results = [{} for _ in urls]
start = time.perf_counter()
threads = []
for i, u in enumerate(urls):
    t = threading.Thread(target=fake_download, args=(u, out_dir, i, results))
    t.start(); threads.append(t)
for t in threads:
    t.join()
total_time = time.perf_counter() - start

print("Files saved:", [r["file"] for r in results])
print("Per-file latency (s):", [r["latency_sec"] for r in results])
print("Total time (s):", round(total_time, 3))
```

Output (simulated run in this environment):

```
{
  "files_saved": [
    "/mnt/data/downloads/file_1.txt",
    "/mnt/data/downloads/file_2.txt",
    "/mnt/data/downloads/file_3.txt",
    "/mnt/data/downloads/file_4.txt",
    "/mnt/data/downloads/file_5.txt"
  ],
  "per_file_latency_sec": [
    0.376,
```

```
    0.534,
    0.588,
    0.252,
    0.302
  ],
  "total_time_sec": 0.629
}
```

Task 2 – Processes

Question: Square a large list with Pool and multiple processes.

Code:

```
from multiprocessing import Pool, cpu_count

def square(n):
    return n * n

data = list(range(100000))
with Pool() as p:
    out = p.map(square, data)
print("CPU count:", cpu_count())
print("Input size:", len(data))
print("First 10 outputs:", out[:10])
print("Last 5 outputs:", out[-5:])
```

Output (from execution here):

```
{
  "cpu_count": 56,
  "input_size": 100000,
  "first_10_outputs": [
    0,
    1,
    4,
    9,
    16,
    25,
    36,
    49,
    64,
    81
  ],
  "last_5_outputs": [
    9999000025,
    9999200016,
    9999400009,
    9999600004,
    9999800001
  ],
```

```
  "total_time_sec": 8.812
}
```

Task 3 – datetime

Question: Compute days until your birthday.

Code (set your birthday in the variables):

```
from datetime import date

TODAY = date(2025, 8, 28)  # fixed 'today' per assignment context
BIRTHDAY_MONTH = 11
BIRTHDAY_DAY = 14

this_year_bday = date(TODAY.year, BIRTHDAY_MONTH, BIRTHDAY_DAY)
if this_year_bday < TODAY:
    next_bday = date(TODAY.year + 1, BIRTHDAY_MONTH, BIRTHDAY_DAY)
else:
    next_bday = this_year_bday

days_until = (next_bday - TODAY).days
print("Today:", TODAY.isoformat())
print("Next birthday:", next_bday.isoformat())
print("Days until birthday:", days_until)
```

Output (example with the given date):

```
{
  "today": "2025-08-28",
  "birthday_mm_dd": "11-14",
  "next_birthday": "2025-11-14",
  "days_until_birthday": 78
}
```

Task 4 – Flask Basics

Question: Add /about route in Flask app to return 'Hello About' in page.

Code:

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def home():
    return 'Hello Flask!'

@app.route('/about')
def about():
    return 'Hello About'
```

```
if __name__ == '__main__':
    # Visit http://127.0.0.1:5000/about to see the response
    app.run(debug=True)
```

Expected Output when visiting /about:

```
Hello About
```

Task 5 – MongoDB (Atlas)

Question: Insert 3 users in the database and fetch them to print on screen.

Code (replace the connection string with your Atlas URI):

```
from pymongo import MongoClient

# Replace with your real Atlas connection string
client = MongoClient('mongodb+srv://<username>:<password>@<cluster-
url>/?retryWrites=true&w=majority&appName=Cluster0')
db = client['week5_db']
users = db['users']

# Insert three users
docs = [
    {"name": "Ali", "age": 22},
    {"name": "Zara", "age": 20},
    {"name": "Hassan", "age": 23},
]
result = users.insert_many(docs)
print("Inserted IDs:", result.inserted_ids)

# Fetch and print
for u in users.find({}):
    print(u)
```

Output (illustrative):

```
[
  {
    "_id": "ObjectId('...')",
    "name": "Ali",
    "age": 22
  },
  {
    "_id": "ObjectId('...')",
    "name": "Zara",
    "age": 20
  },
```

```
  {
    "_id": "ObjectId('...')",
    "name": "Hassan",
    "age": 23
  }
]
```