Practical no 3

Aim: Threading and Single Thread Control Flow

3.1: Practice Thread Creation and basic thread lifecycle using standard libraries(e.g., pthreads and Java threads)

```
import threading
import time

def task(name, delay):
    print(f"Task {name} started")
    time.sleep(delay)
    print(f"Task {name} finished after {delay} seconds.")

def singleThreaded():
    startTime = time.time()
    task("A", 2)
    task("B", 2)
    endTime = time.time()
    print(f"Single-threaded execution time: {endTime - startTime} seconds")

def multiThreaded():
    startTime = time.time()

    threadA = threading.Thread(target=task, args=("A", 2))
    threadB = threading.Thread(target=task, args=("B", 2))

    threadA.start()
    threadB.start()

    threadA.join()
    threadB.join()

    endTime = time.time()
    print(f"Multi-threaded execution time: {endTime - startTime} seconds")

if __name__ == "__main__":
    print("Running single-threaded version")
    singleThreaded()

    print("\nRunning multi-threaded version")
    multiThreaded()
```

```
Running single-threaded version
Task A started
Task A finished after 2 seconds.
Task B started
Task B finished after 2 seconds.
Single-threaded execution time: 4.001172304153442 seconds

Running multi-threaded version
Task A started
Task B started
Task B finished after 2 seconds.
Task A finished after 2 seconds.
Multi-threaded execution time: 2.0047800540924072 seconds
```

3.2: Observe execution order, thread joining and delays

```python
import threading
import time

def task(name, delay):
    print(f"[{time.strftime('%H:%M:%S')}] Thread {name} starting")
    print(f"[{time.strftime('%H:%M:%S')}] Thread {name} sleeping for {delay} seconds")
    time.sleep(delay)
    print(f"[{time.strftime('%H:%M:%S')}] Thread {name} finished")

def main():
    print(f"[{time.strftime('%H:%M:%S')}] Main thread: Creating threads")

    threads = [
        threading.Thread(target=task, args=("A", 3), name="Thread-A"),
        threading.Thread(target=task, args=("B", 2), name="Thread-B"),
        threading.Thread(target=task, args=("C", 1), name="Thread-C"),
    ]

    for t in threads:
        print(f"[{time.strftime('%H:%M:%S')}] Main thread: Starting {t.name}")
        t.start()

    for t in threads:
        print(f"[{time.strftime('%H:%M:%S')}] Main thread: Waiting for {t.name} to finish")
        t.join()
        print(f"[{time.strftime('%H:%M:%S')}] Main thread: {t.name} finished")

    print(f"[{time.strftime('%H:%M:%S')}] Main thread: All threads completed")

if __name__ == "__main__":
    main()
```

```
[10:42:45] Main thread: Creating threads
[10:42:45] Main thread: Starting Thread-A
[10:42:45] Thread A starting
[10:42:45] Main thread: Starting Thread-B
[10:42:45] Thread A sleeping for 3 seconds
[10:42:45] Thread B starting
[10:42:45] Main thread: Starting Thread-C
[10:42:45] Thread B sleeping for 2 seconds
[10:42:45] Thread C starting
[10:42:45] Main thread: Waiting for Thread-A to finish
[10:42:45] Thread C sleeping for 1 seconds
[10:42:46] Thread C finished
[10:42:47] Thread B finished
[10:42:48] Thread A finished
[10:42:48] Main thread: Thread-A finished
[10:42:48] Main thread: Waiting for Thread-B to finish
[10:42:48] Main thread: Thread-B finished
[10:42:48] Main thread: Waiting for Thread-C to finish
[10:42:48] Main thread: Thread-C finished
[10:42:48] Main thread: All threads completed
```

3.3: Compare Execution time between:

1) Sequential (single-threaded) execution
2) Multi-threaded execution

```
import threading
import time

def task(name, delay):
    print(f"[{time.strftime('%H:%M:%S')}] Task {name} started")
    time.sleep(delay)
    print(f"[{time.strftime('%H:%M:%S')}] Task {name} finished")

def sequentialExecution():
    print("\n=== Sequential Execution ===")
    startTime = time.time()

    task("A", 3)
    task("B", 2)
    task("C", 1)

    endTime = time.time()
    print(f"Total time (Sequential): {endTime - startTime:.2f} seconds")

def multithreadedExecution():
    print("\n=== Multithreaded Execution ===")
    startTime = time.time()

    threads = [
        threading.Thread(target=task, args=("A", 3)),
        threading.Thread(target=task, args=("B", 2)),
        threading.Thread(target=task, args=("C", 1)),
    ]

    for t in threads:
        t.start()
    for t in threads:
        t.join()

    endTime = time.time()
    print(f"Total time (Multithreaded): {endTime - startTime:.2f} seconds")

# Run both executions
sequentialExecution()
multithreadedExecution()
```

```
=== Sequential Execution ===
[10:44:34] Task A started
[10:44:37] Task A finished
[10:44:37] Task B started
[10:44:39] Task B finished
[10:44:39] Task C started
[10:44:40] Task C finished
Total time (Sequential): 6.00 seconds

=== Multithreaded Execution ===
[10:44:40] Task A started
[10:44:40] Task B started
[10:44:40] Task C started
[10:44:41] Task C finished
[10:44:42] Task B finished
[10:44:43] Task A finished
Total time (Multithreaded): 3.00 seconds
```