ASSIGNMENT 2

Name: Payal Rathore

Roll No: 2301410022

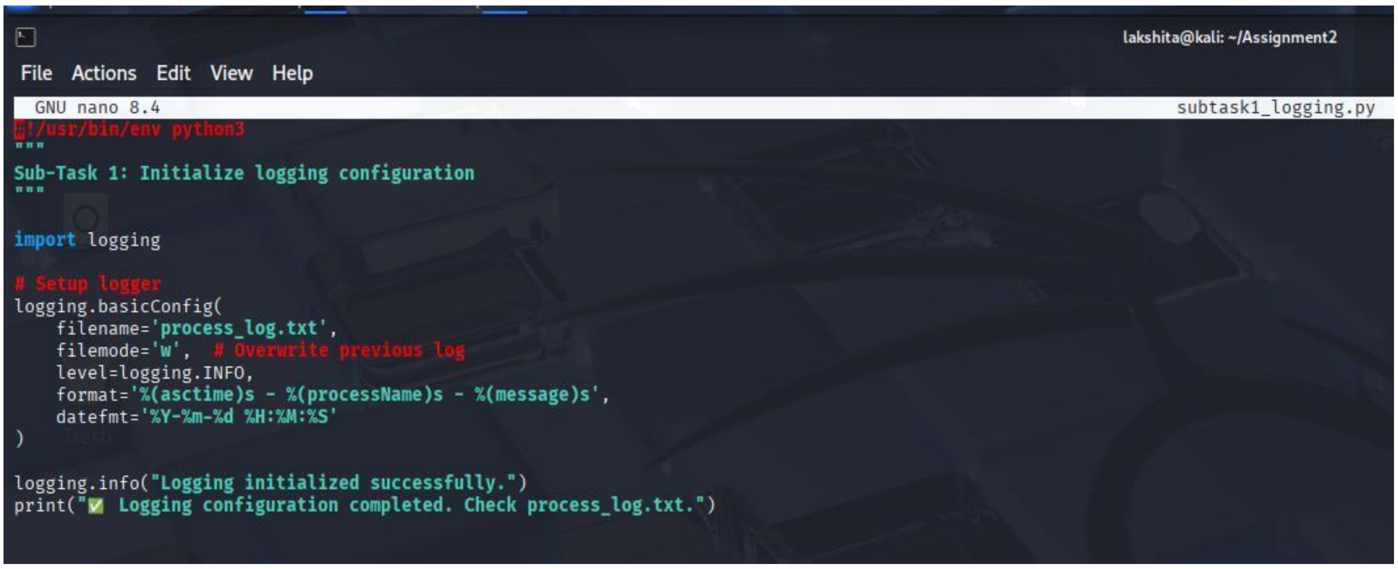
Course: BTech CSE (Cyber Security)

Problem Statement:

Modern operating systems are responsible for initializing system components, creating processes, managing execution, and gracefully shutting down. This lab aims to simulate these core concepts using Python, helping students visualize how processes are handled at the OS level. The focus is on creating a simplified startup mechanism that spawns multiple processes and logs their lifecycle using the multiprocessing and logging modules. This hands-on simulation enhances conceptual clarity and promotes coding proficiency in scripting real-world OS behavior.

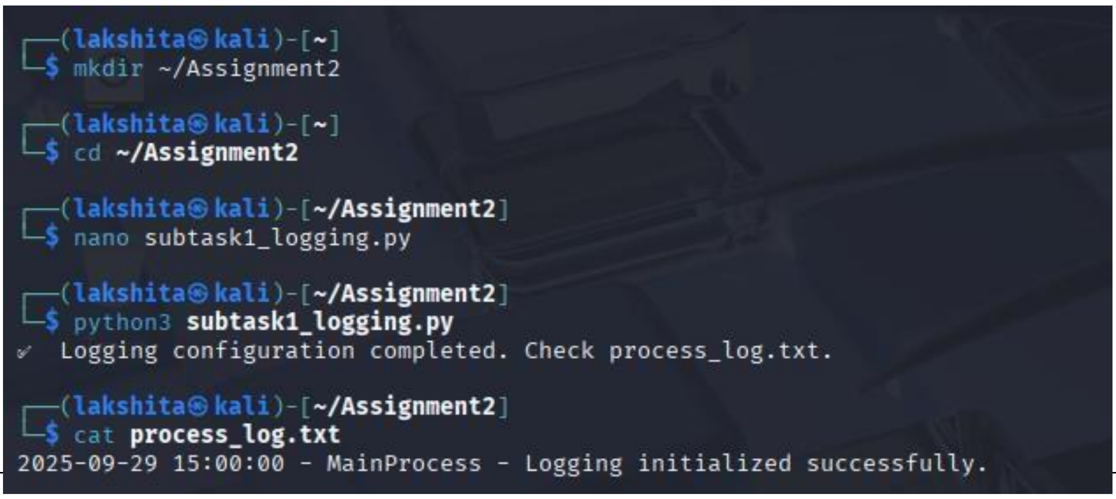
Sub-Task 1: Initialize the logging configuration

CODE



```
#!/usr/bin/env python3
"""
Sub-Task 1: Initialize logging configuration
"""

import logging

# Setup logger
logging.basicConfig(
    filename='process_log.txt',
    filemode='w',  # Overwrite previous log
    level=logging.INFO,
    format='%(asctime)s - %(processName)s - %(message)s',
    datefmt='%Y-%m-%d %H:%M:%S'
)

logging.info("Logging initialized successfully.")
print("✅ Logging configuration completed. Check process_log.txt.")
```

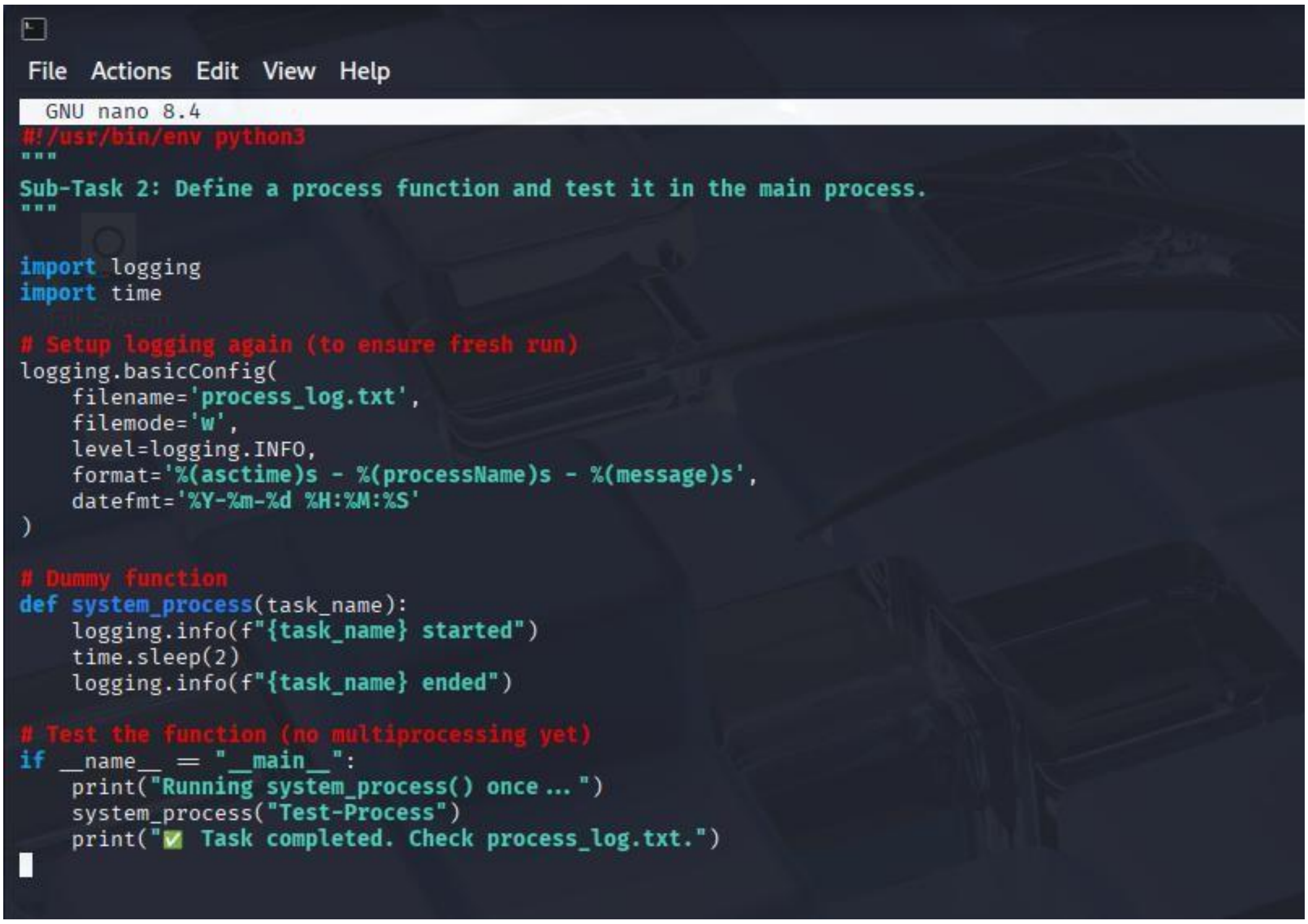
OUTPUT



```
┌──(lakshita㉿kali)-[~]
└─$ mkdir ~/Assignment2

┌──(lakshita㉿kali)-[~]
└─$ cd ~/Assignment2

┌──(lakshita㉿kali)-[~/Assignment2]
└─$ nano subtask1_logging.py

┌──(lakshita㉿kali)-[~/Assignment2]
└─$ python3 subtask1_logging.py
✔ Logging configuration completed. Check process_log.txt.

┌──(lakshita㉿kali)-[~/Assignment2]
└─$ cat process_log.txt
2025-09-29 15:00:00 - MainProcess - Logging initialized successfully.
```

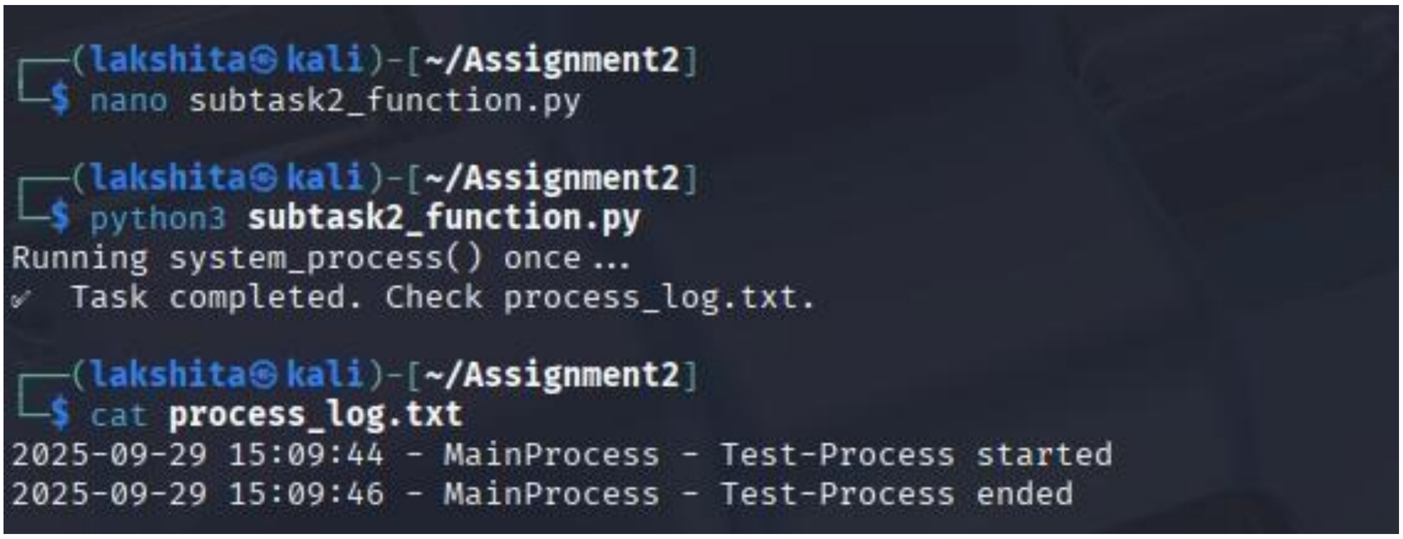
Sub-Task 2: Define a function that simulates a process task (e.g., sleep for 2 seconds).

CODE



```python
#!/usr/bin/env python3
"""
Sub-Task 2: Define a process function and test it in the main process.
"""

import logging
import time

# Setup logging again (to ensure fresh run)
logging.basicConfig(
    filename='process_log.txt',
    filemode='w',
    level=logging.INFO,
    format='%(asctime)s - %(processName)s - %(message)s',
    datefmt='%Y-%m-%d %H:%M:%S'
)

# Dummy function
def system_process(task_name):
    logging.info(f"{task_name} started")
    time.sleep(2)
    logging.info(f"{task_name} ended")

# Test the function (no multiprocessing yet)
if __name__ == "__main__":
    print("Running system_process() once ... ")
    system_process("Test-Process")
    print("✅ Task completed. Check process_log.txt.")
```

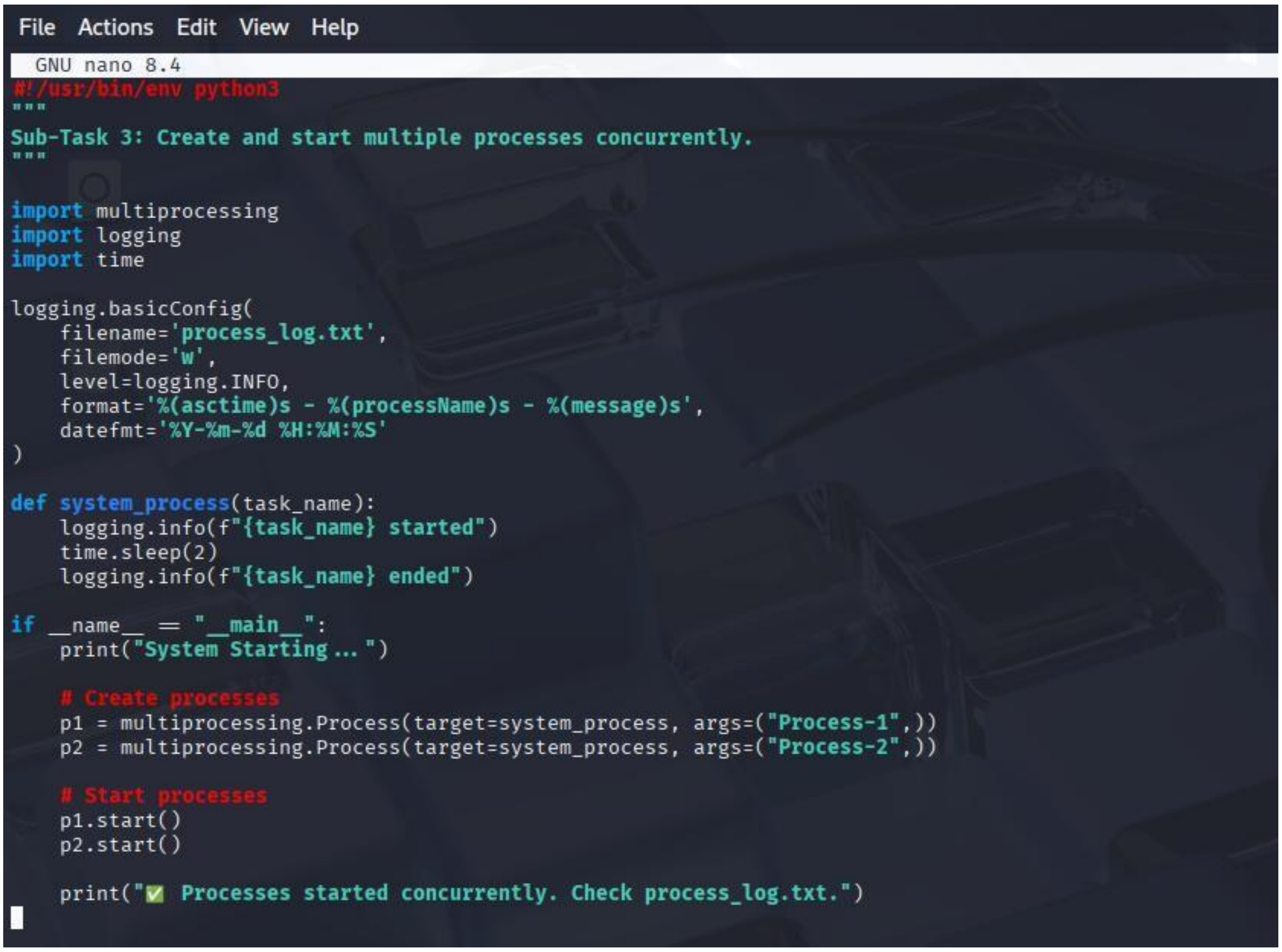
OUTPUT



```
┌──(lakshita㉿kali)-[~/Assignment2]
└─$ nano subtask2_function.py

┌──(lakshita㉿kali)-[~/Assignment2]
└─$ python3 subtask2_function.py
Running system_process() once ...
✔ Task completed. Check process_log.txt.

┌──(lakshita㉿kali)-[~/Assignment2]
└─$ cat process_log.txt
2025-09-29 15:09:44 - MainProcess - Test-Process started
2025-09-29 15:09:46 - MainProcess - Test-Process ended
```

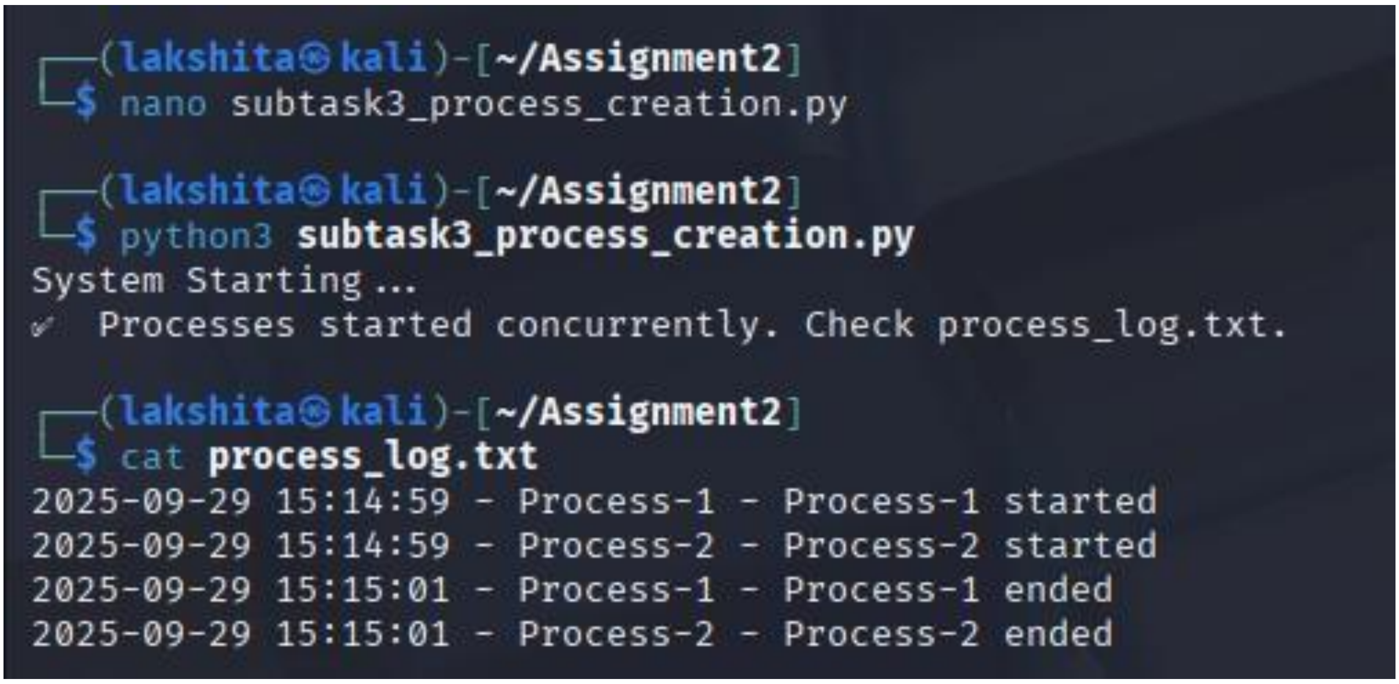
Sub-Task 3: Create at least two processes and start them concurrently.

CODE



```python
#!/usr/bin/env python3
"""
Sub-Task 3: Create and start multiple processes concurrently.
"""

import multiprocessing
import logging
import time

logging.basicConfig(
    filename='process_log.txt',
    filemode='w',
    level=logging.INFO,
    format='%(asctime)s - %(processName)s - %(message)s',
    datefmt='%Y-%m-%d %H:%M:%S'
)

def system_process(task_name):
    logging.info(f"{task_name} started")
    time.sleep(2)
    logging.info(f"{task_name} ended")

if __name__ == "__main__":
    print("System Starting ... ")

    # Create processes
    p1 = multiprocessing.Process(target=system_process, args=("Process-1",))
    p2 = multiprocessing.Process(target=system_process, args=("Process-2",))

    # Start processes
    p1.start()
    p2.start()

    print("✅ Processes started concurrently. Check process_log.txt.")
```

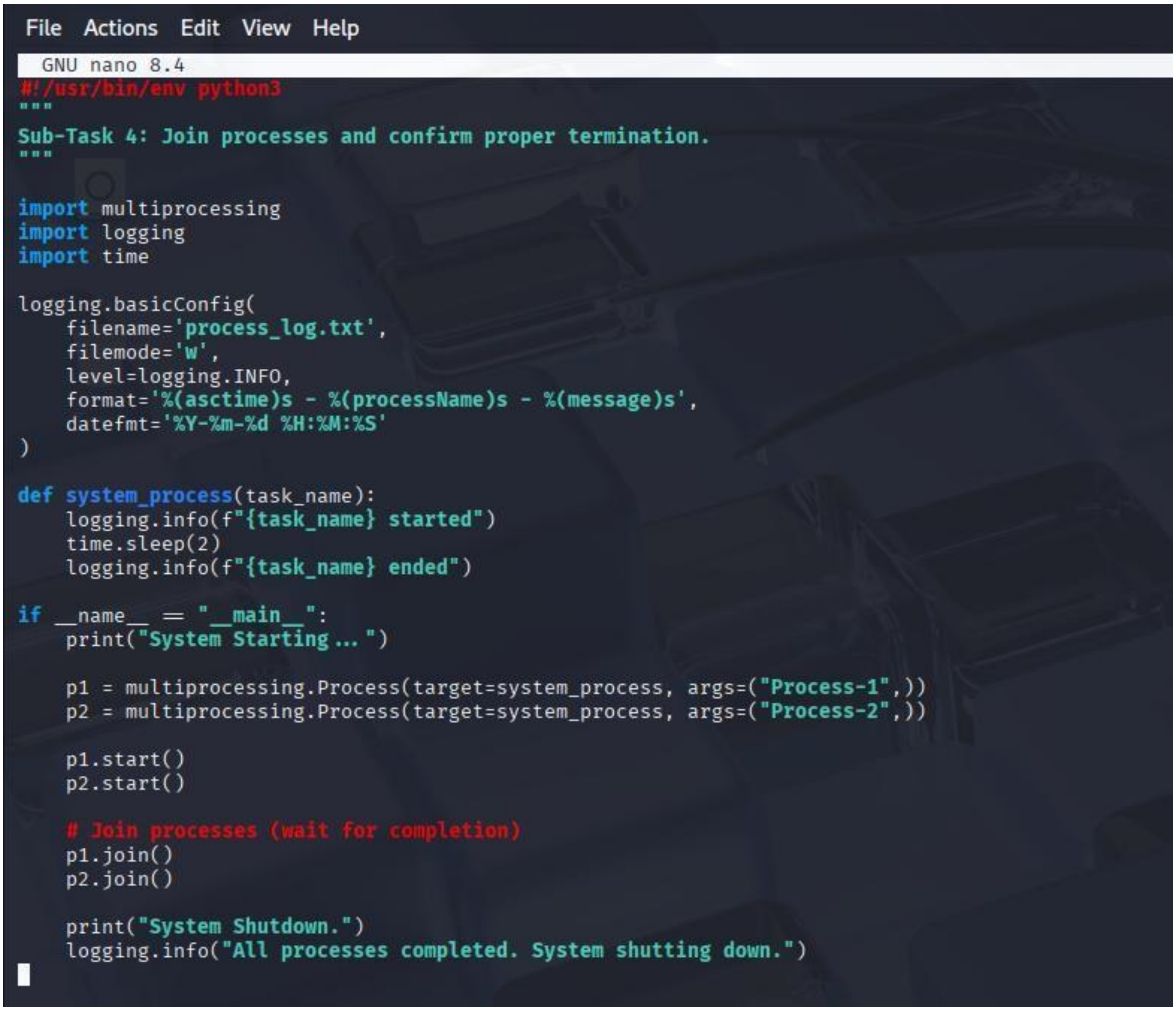
OUTPUT



```
┌──(lakshita㉿kali)-[~/Assignment2]
└─$ nano subtask3_process_creation.py

┌──(lakshita㉿kali)-[~/Assignment2]
└─$ python3 subtask3_process_creation.py
System Starting ...
✅ Processes started concurrently. Check process_log.txt.

┌──(lakshita㉿kali)-[~/Assignment2]
└─$ cat process_log.txt
2025-09-29 15:14:59 - Process-1 - Process-1 started
2025-09-29 15:14:59 - Process-2 - Process-2 started
2025-09-29 15:15:01 - Process-1 - Process-1 ended
2025-09-29 15:15:01 - Process-2 - Process-2 ended
```

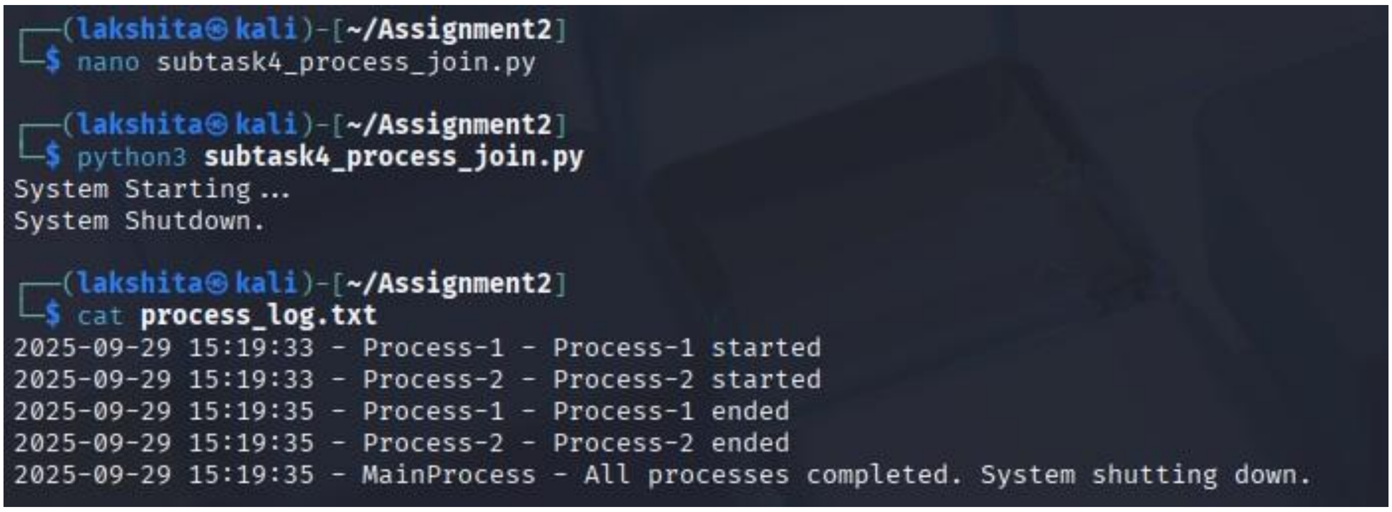
Sub-Task 4: Ensure proper termination and joining of processes, and verify the output in the log file.

CODE



```
#!/usr/bin/env python3
"""
Sub-Task 4: Join processes and confirm proper termination.
"""

import multiprocessing
import logging
import time

logging.basicConfig(
    filename='process_log.txt',
    filemode='w',
    level=logging.INFO,
    format='%(asctime)s - %(processName)s - %(message)s',
    datefmt='%Y-%m-%d %H:%M:%S'
)

def system_process(task_name):
    logging.info(f"{task_name} started")
    time.sleep(2)
    logging.info(f"{task_name} ended")

if __name__ == "__main__":
    print("System Starting...")

    p1 = multiprocessing.Process(target=system_process, args=("Process-1",))
    p2 = multiprocessing.Process(target=system_process, args=("Process-2",))

    p1.start()
    p2.start()

    # Join processes (wait for completion)
    p1.join()
    p2.join()

    print("System Shutdown.")
    logging.info("All processes completed. System shutting down.")
```

OUTPUT



```
┌──(lakshita㉿kali)-[~/Assignment2]
└─$ nano subtask4_process_join.py

┌──(lakshita㉿kali)-[~/Assignment2]
└─$ python3 subtask4_process_join.py
System Starting...
System Shutdown.

┌──(lakshita㉿kali)-[~/Assignment2]
└─$ cat process_log.txt
2025-09-29 15:19:33 - Process-1 - Process-1 started
2025-09-29 15:19:33 - Process-2 - Process-2 started
2025-09-29 15:19:35 - Process-1 - Process-1 ended
2025-09-29 15:19:35 - Process-2 - Process-2 ended
2025-09-29 15:19:35 - MainProcess - All processes completed. System shutting down.
```