Lab Experiment Sheet - 02

- Course Code- ENCS351
- Course Name - Operating system
- Program Name: B.Tech. CSE- AI ML

- Student Name: SURYANSH DHAMA
- Roll No. 2301730126
- GitHub Repository Link : https://github.com/Suryansh-Dhama/OS-Assignment.git

Problem Title:
System Startup, Process Creation, and Termination Simulation in Python

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.

Tools/Technology Used:

• Python 3.x
• multiprocessing module
• time module
• logging module

Learning Objectives:

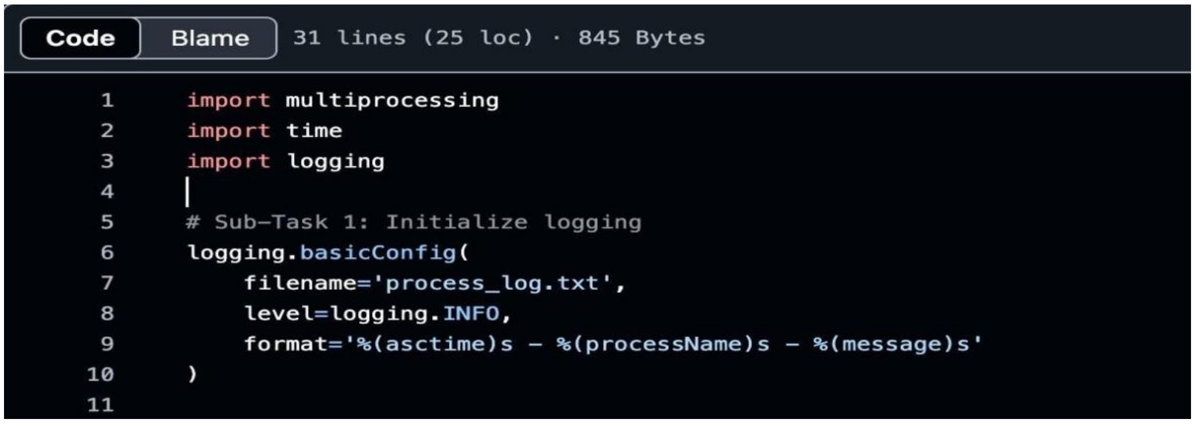
1. Upon completing this lab, students will be able to:
2. Understand the concepts of system booting, process creation, and termination.
3. Develop Python scripts using multiprocessing and logging modules.
4. Simulate system behaviour using programming constructs.

Assignment Tasks:

1. Write a Python script to simulate a basic system startup sequence.
2. Use the multiprocessing module to create at least two child processes that perform dummy tasks.
3. Implement proper logging to track process start and end times.
4. Generate a log file (process_log.txt) to reflect system-like behavior.
5. Submit the Python script and log file along with a short report explaining your implementation.

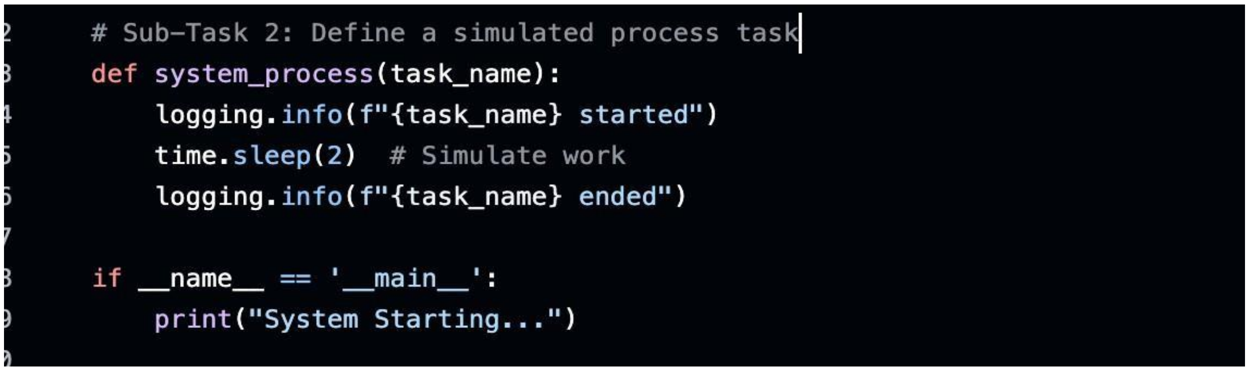
Sub-Tasks:

1. **Sub-Task 1:** Initialize the logging configuration to capture timestamped messages.



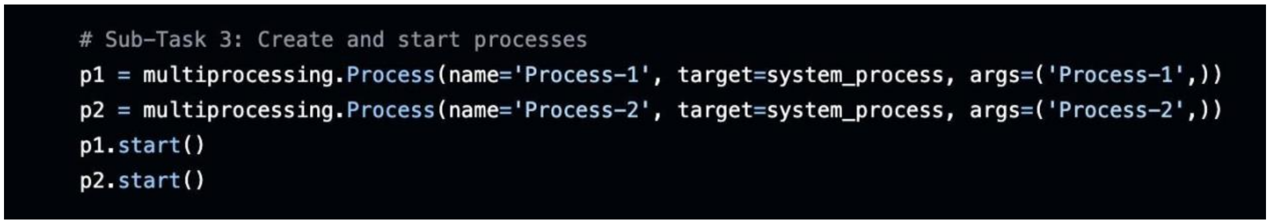
```python
import multiprocessing
import time
import logging

# Sub-Task 1: Initialize logging
logging.basicConfig(
    filename='process_log.txt',
    level=logging.INFO,
    format='%(asctime)s - %(processName)s - %(message)s'
)

```

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



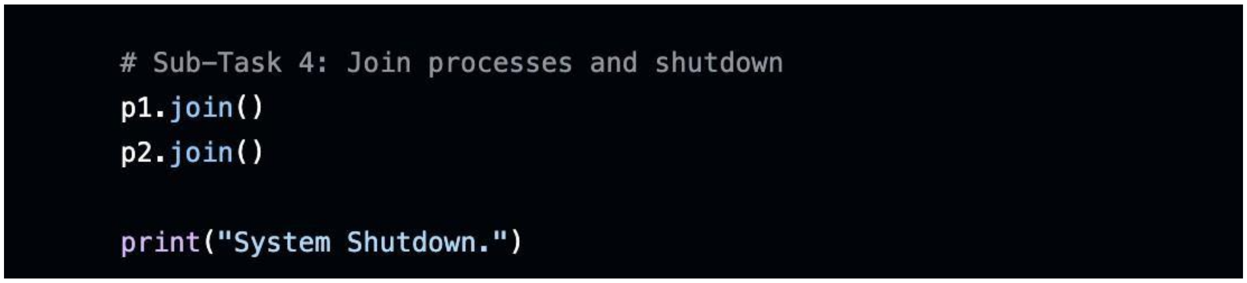
```python
# Sub-Task 2: Define a simulated process task
def system_process(task_name):
    logging.info(f"{task_name} started")
    time.sleep(2)  # Simulate work
    logging.info(f"{task_name} ended")

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

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



```python
# Sub-Task 3: Create and start processes
p1 = multiprocessing.Process(name='Process-1', target=system_process, args=('Process-1',))
p2 = multiprocessing.Process(name='Process-2', target=system_process, args=('Process-2',))
p1.start()
p2.start()
```

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

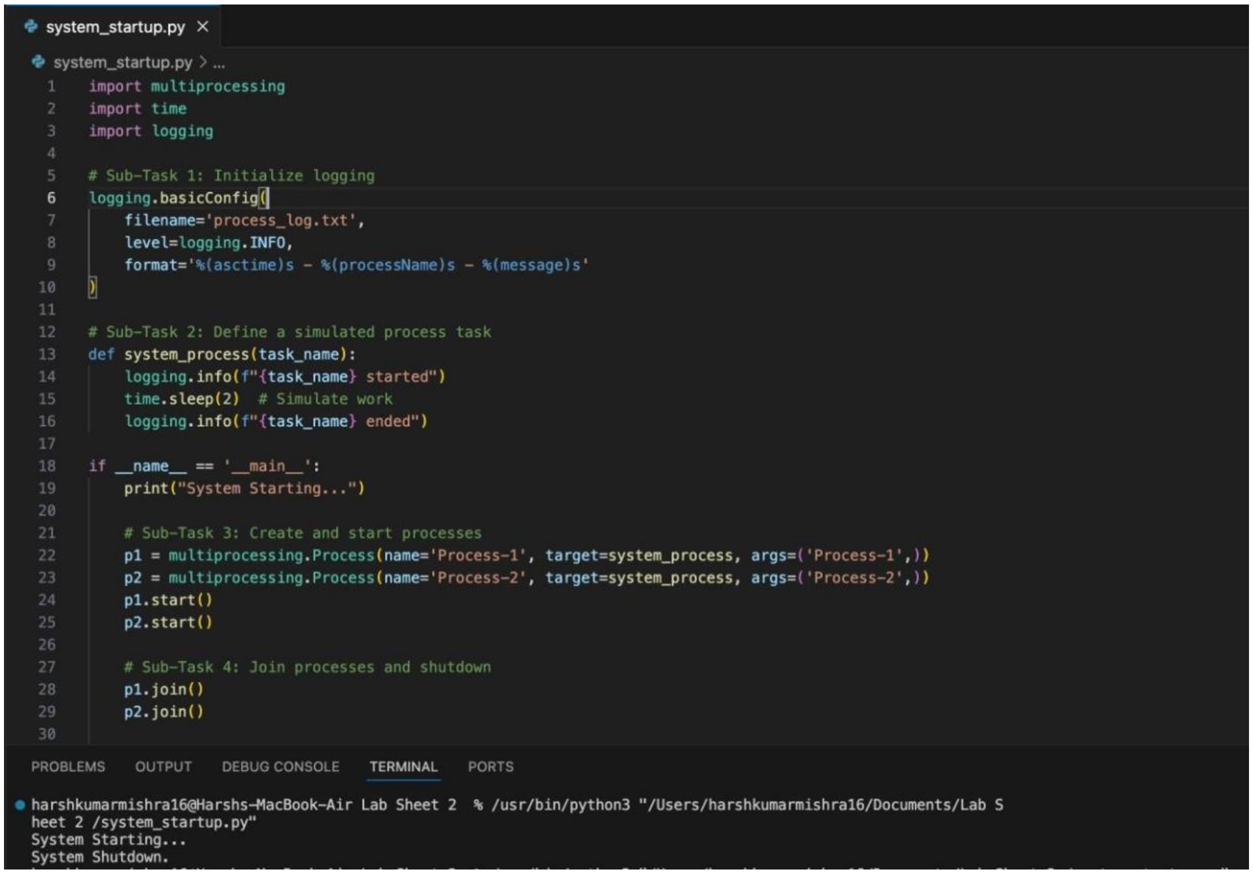


```python
# Sub-Task 4: Join processes and shutdown
p1.join()
p2.join()

print("System Shutdown.")
```

<u>Program Code with Explanation:</u>

The following Python script simulates a simple system startup. It creates a few dummy processes which perform simple tasks. Each process logs its start and end, and the logger records the process flow, simulating system behavior.

Code:



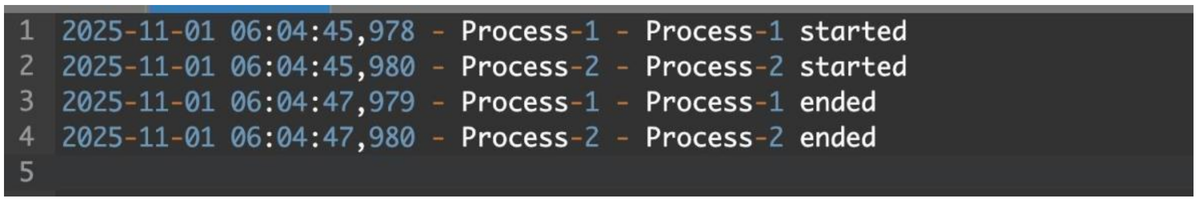
```python
import multiprocessing
import time
import logging

# Sub-Task 1: Initialize logging
logging.basicConfig(
    filename='process_log.txt',
    level=logging.INFO,
    format='%(asctime)s - %(processName)s - %(message)s'
)

# Sub-Task 2: Define a simulated process task
def system_process(task_name):
    logging.info(f"{task_name} started")
    time.sleep(2)  # Simulate work
    logging.info(f"{task_name} ended")

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

    # Sub-Task 3: Create and start processes
    p1 = multiprocessing.Process(name='Process-1', target=system_process, args=('Process-1',))
    p2 = multiprocessing.Process(name='Process-2', target=system_process, args=('Process-2',))
    p1.start()
    p2.start()

    # Sub-Task 4: Join processes and shutdown
    p1.join()
    p2.join()

```

```
harshkumarmishra16@Harshs-MacBook-Air Lab Sheet 2  % /usr/bin/python3 "/Users/harshkumarmishra16/Documents/Lab S
heet 2 /system_startup.py"
System Starting...
System Shutdown.
```

Expected Output:

• The terminal displays 'System Starting...' and 'System Shutdown.'

• A log file 'process_log.txt' is generated which records the start and end of each process.



```
2025-11-01 06:04:45,978 - Process-1 - Process-1 started
2025-11-01 06:04:45,980 - Process-2 - Process-2 started
2025-11-01 06:04:47,979 - Process-1 - Process-1 ended
2025-11-01 06:04:47,980 - Process-2 - Process-2 ended
```