

Sub-Task 1: Initialize the logging configuration

Objective: Set up the logging system to log messages with timestamps and process names.

import logging

Setup logger

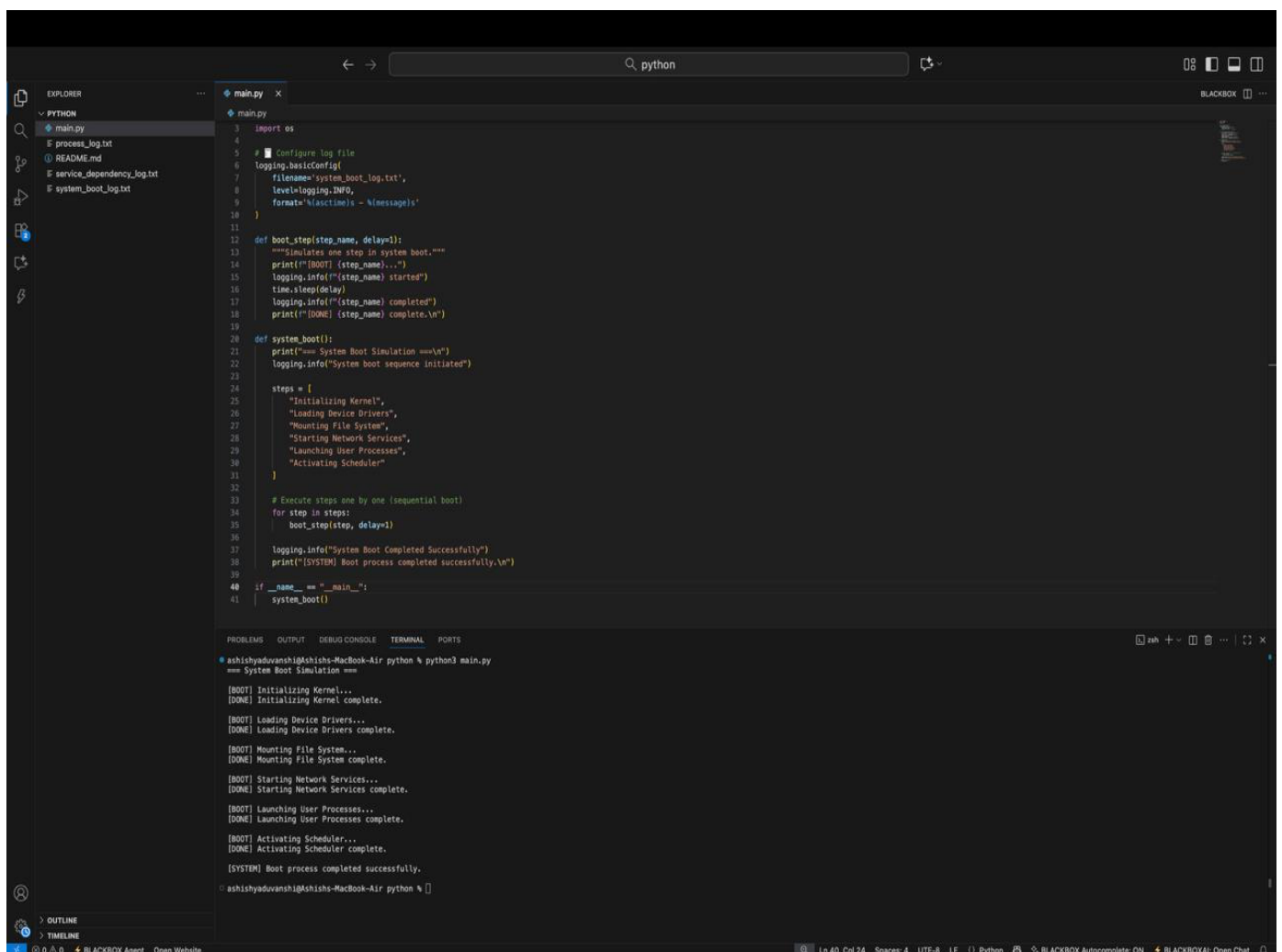
logging.basicConfig(

filename='process_log.txt',

level=logging.INFO,

format='%(asctime)s - %(processName)s - %(message)s'

)



```
3 import os
4
5 # Configure log file
6 logging.basicConfig(
7     filename='system_boot_log.txt',
8     level=logging.INFO,
9     format='%(asctime)s - %(message)s'
10 )
11
12 def boot_step(step_name, delay=1):
13     """Simulates one step in system boot."""
14     print(f"[BOOT] {step_name}...")
15     logging.info(f"[step_name] started")
16     time.sleep(delay)
17     logging.info(f"[step_name] completed")
18     print(f"[DONE] {step_name} complete.\n")
19
20 def system_boot():
21     print("== System Boot Simulation ==\n")
22     logging.info("System boot sequence initiated")
23
24     steps = [
25         "Initializing Kernel",
26         "Loading Device Drivers",
27         "Mounting File System",
28         "Starting Network Services",
29         "Launching User Processes",
30         "Activating Scheduler"
31     ]
32
33     # Execute steps one by one (sequential boot)
34     for step in steps:
35         boot_step(step, delay=1)
36
37     logging.info("System Boot Completed Successfully")
38     print(f"[SYSTEM] Boot process completed successfully.\n")
39
40 if __name__ == "__main__":
41     system_boot()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
ashishyaduvanshi@Ashishs-MacBook-Air python & python3 main.py
== System Boot Simulation ==

[BOOT] Initializing Kernel...
[DONE] Initializing Kernel complete.

[BOOT] Loading Device Drivers...
[DONE] Loading Device Drivers complete.

[BOOT] Mounting File System...
[DONE] Mounting File System complete.

[BOOT] Starting Network Services...
[DONE] Starting Network Services complete.

[BOOT] Launching User Processes...
[DONE] Launching User Processes complete.

[BOOT] Activating Scheduler...
[DONE] Activating Scheduler complete.

[SYSTEM] Boot process completed successfully.
```

Ln 40, Col 24 Spaces: 4 UTF-8 LF Python BLACKBOX Autocomplete: ON BLACKBOXAI: Open Chat

Sub-Task 2: Define a function that simulates a process task

Objective: Write a function that mimics the work of a system process.

```
import time
```

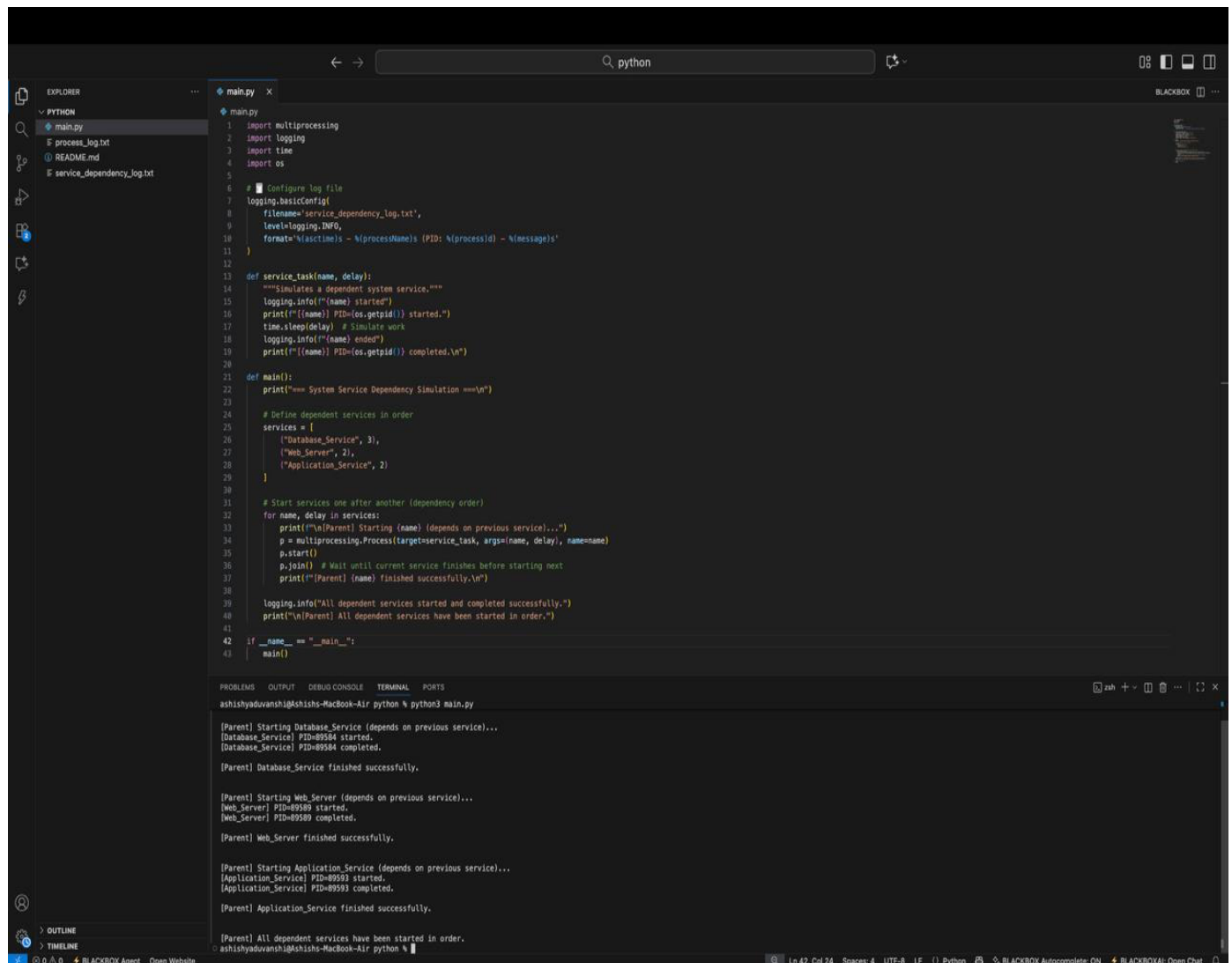
```
# Dummy function to simulate a task
```

```
def system_process(task_name):
```

```
    logging.info(f'{task_name} started')
```

```
    time.sleep(2) # Simulate task delay
```

```
    logging.info(f'{task_name} ended')
```



```
1 import multiprocessing
2 import logging
3 import time
4 import os
5
6 # Configure log file
7 logging.basicConfig(
8     filename='service_dependency_log.txt',
9     level=logging.INFO,
10    format='%(asctime)s - %(processName)s (PID: %(processid)s) - %(message)s'
11)
12
13 def service_task(name, delay):
14     """Simulates a dependent system service."""
15     logging.info(f'{name} started')
16     print(f'({name}) PID={os.getpid()} started.')
17     time.sleep(delay) # Simulate work
18     logging.info(f'{name} ended')
19     print(f'({name}) PID={os.getpid()} completed.\n')
20
21 def main():
22     print("=== System Service Dependency Simulation ===\n")
23
24     # Define dependent services in order
25     services = [
26         ("Database_Service", 3),
27         ("Web_Server", 2),
28         ("Application_Service", 2)
29     ]
30
31     # Start services one after another (dependency order)
32     for name, delay in services:
33         print(f'({name}) Starting {name} (depends on previous service)...')
34         p = multiprocessing.Process(target=service_task, args=(name, delay), name=name)
35         p.start()
36         p.join() # Wait until current service finishes before starting next
37         print(f'({name}) {name} finished successfully.\n')
38
39     logging.info("All dependent services started and completed successfully.")
40     print("\n(Parent) All dependent services have been started in order.")
41
42 if __name__ == "__main__":
43     main()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
ashishyaduvanshi@ashish-MacBook-Air python % python3 main.py
(Parent) Starting Database_Service (depends on previous service)...
(Database_Service) PID=89584 started.
(Database_Service) PID=89584 completed.
(Parent) Database_Service finished successfully.
(Parent) Starting Web_Server (depends on previous service)...
(Web_Server) PID=89589 started.
(Web_Server) PID=89589 completed.
(Parent) Web_Server finished successfully.
(Parent) Starting Application_Service (depends on previous service)...
(Application_Service) PID=89593 started.
(Application_Service) PID=89593 completed.
(Parent) Application_Service finished successfully.
(Parent) All dependent services have been started in order.
```

Ln 42, Col 24 Spaces: 4 UTF-8 LF Python BLACKBOX Autocomplete: ON BLACKBOX AI: Open Chat

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

Objective: Use the multiprocessing module to initiate parallel tasks.

```
import multiprocessing

if __name__ == '__main__':

    print("System Starting...")

    # Create processes

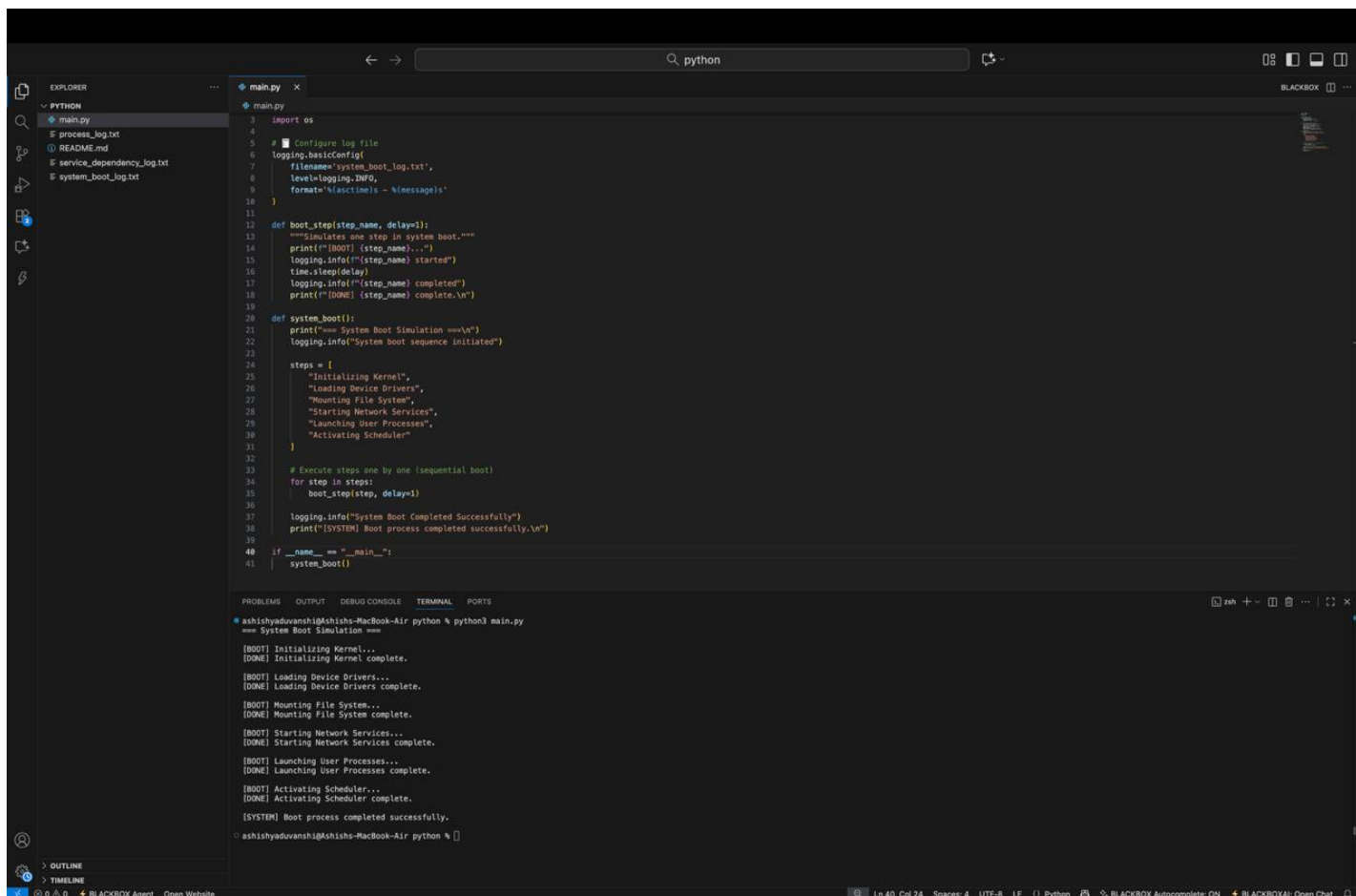
    p1 = multiprocessing.Process(target=system_process, args=('Process-1',))

    p2 = multiprocessing.Process(target=system_process, args=('Process-2',))

    # Start processes

    p1.start()

    p2.start()
```



The screenshot displays a code editor with a Python script named `main.py` and its execution output in the terminal. The script simulates a system boot process by creating two parallel tasks, `Process-1` and `Process-2`, using the `multiprocessing` module. The script includes logging for each step and a delay between steps to simulate a realistic boot sequence.

```
1 import os
2
3 # Configure log file
4 logging.basicConfig(
5     filename='system_boot_log.txt',
6     level=logging.DEBUG,
7     format='%(asctime)s - %(message)s'
8 )
9
10
11 def boot_step(step_name, delay=1):
12     """Simulates one step in system boot."""
13     print(f"[BOOT] {step_name}...")
14     logging.info(f"[step_name] started")
15     time.sleep(delay)
16     logging.info(f"[step_name] completed")
17     print(f"[DONE] {step_name} complete.\n")
18
19
20 def system_boot():
21     print("=== System Boot Simulation ===\n")
22     logging.info("System boot sequence initiated")
23
24     steps = [
25         "Initializing Kernel",
26         "Loading Device Drivers",
27         "Mounting File System",
28         "Starting Network Services",
29         "Launching User Processes",
30         "Activating Scheduler"
31     ]
32
33     # Execute steps one by one (sequential boot)
34     for step in steps:
35         boot_step(step, delay=1)
36
37     logging.info("System Boot Completed Successfully")
38     print("[SYSTEM] Boot process completed successfully.\n")
39
40 if __name__ == "__main__":
41     system_boot()
```

The terminal output shows the execution of the script, displaying the boot sequence steps and their completion status:

```
ashishyaduvanshi@MacBook-Air python & python3 main.py
=== System Boot Simulation ===
[BOOT] Initializing Kernel...
[DONE] Initializing Kernel complete.
[BOOT] Loading Device Drivers...
[DONE] Loading Device Drivers complete.
[BOOT] Mounting File System...
[DONE] Mounting File System complete.
[BOOT] Starting Network Services...
[DONE] Starting Network Services complete.
[BOOT] Launching User Processes...
[DONE] Launching User Processes complete.
[BOOT] Activating Scheduler...
[DONE] Activating Scheduler complete.
[SYSTEM] Boot process completed successfully.
```

Sub-Task 4: Ensure proper termination and verify logs

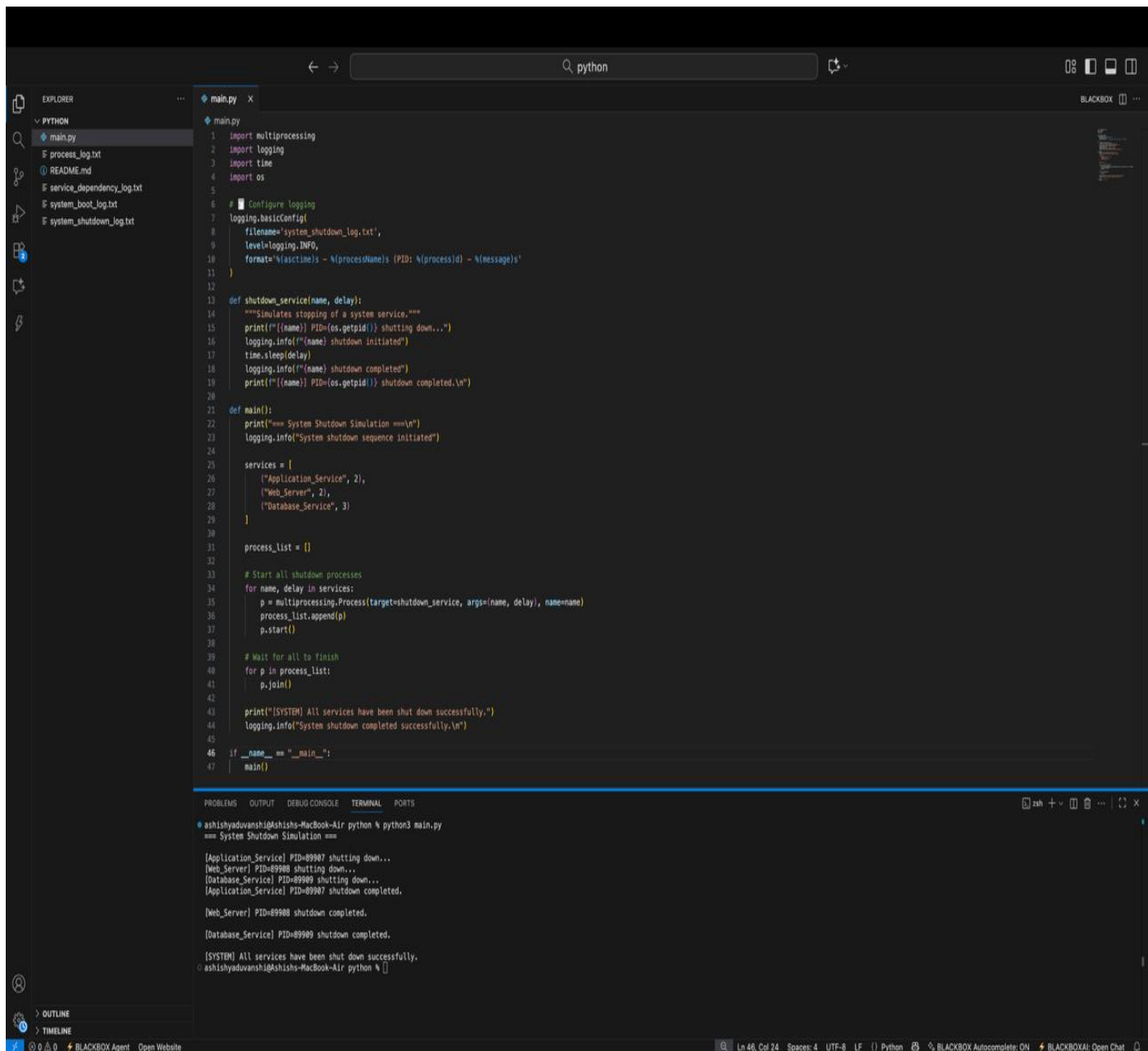
Objective: Wait for processes to complete and confirm the shutdown.

```
# Wait for processes to complete
```

```
p1.join()
```

```
p2.join()
```

```
print("System Shutdown.")
```



The screenshot displays a VS Code editor window with a Python file named `main.py` and a terminal window showing the execution output.

main.py Code:

```
1 import multiprocessing
2 import logging
3 import time
4 import os
5
6 # Configure logging
7 logging.basicConfig(
8     filename='system_shutdown_log.txt',
9     level=logging.INFO,
10    format='%(asctime)s - %(processName)s (PID: %(processId)s) - %(message)s'
11)
12
13 def shutdown_service(name, delay):
14     """Simulates stopping of a system service."""
15     print(f"({name}) PID={os.getpid()} shutting down...")
16     logging.info(f"({name}) shutdown initiated")
17     time.sleep(delay)
18     logging.info(f"({name}) shutdown completed")
19     print(f"({name}) PID={os.getpid()} shutdown completed.\n")
20
21 def main():
22     print("=== System Shutdown Simulation ===\n")
23     logging.info("System shutdown sequence initiated")
24
25     services = [
26         ("Application_Service", 2),
27         ("Web_Server", 2),
28         ("Database_Service", 3)
29     ]
30
31     process_list = []
32
33     # Start all shutdown processes
34     for name, delay in services:
35         p = multiprocessing.Process(target=shutdown_service, args=(name, delay), name=name)
36         process_list.append(p)
37         p.start()
38
39     # Wait for all to finish
40     for p in process_list:
41         p.join()
42
43     print("[SYSTEM] All services have been shut down successfully.")
44     logging.info("System shutdown completed successfully.\n")
45
46 if __name__ == "__main__":
47     main()
```

Terminal Output:

```
ashishyaduvanshi@shishu-MacBook-Air python % python3 main.py
=== System Shutdown Simulation ===

[Application_Service] PID=89987 shutting down...
[Web_Server] PID=89988 shutting down...
[Database_Service] PID=89989 shutting down...
[Application_Service] PID=89987 shutdown completed.

[Web_Server] PID=89988 shutdown completed.

[Database_Service] PID=89989 shutdown completed.

[SYSTEM] All services have been shut down successfully.
ashishyaduvanshi@shishu-MacBook-Air python %
```

The terminal output matches the expected behavior described in the objective, showing the shutdown sequence for three services and a final system status message.