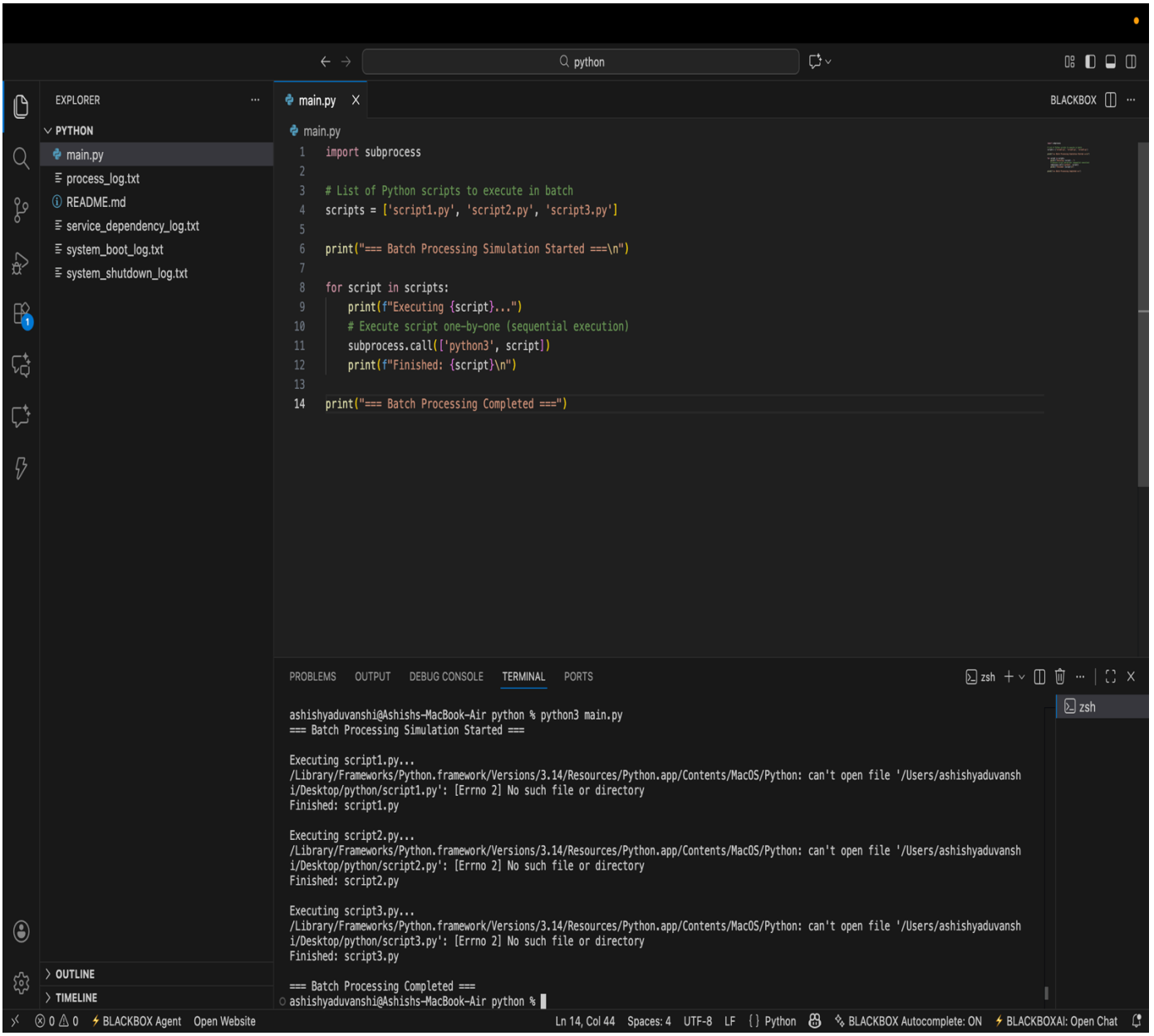
Task 1: Batch Processing Simulation (Python)

Write a Python script to execute multiple .py files sequentially, mimicking batch processing.

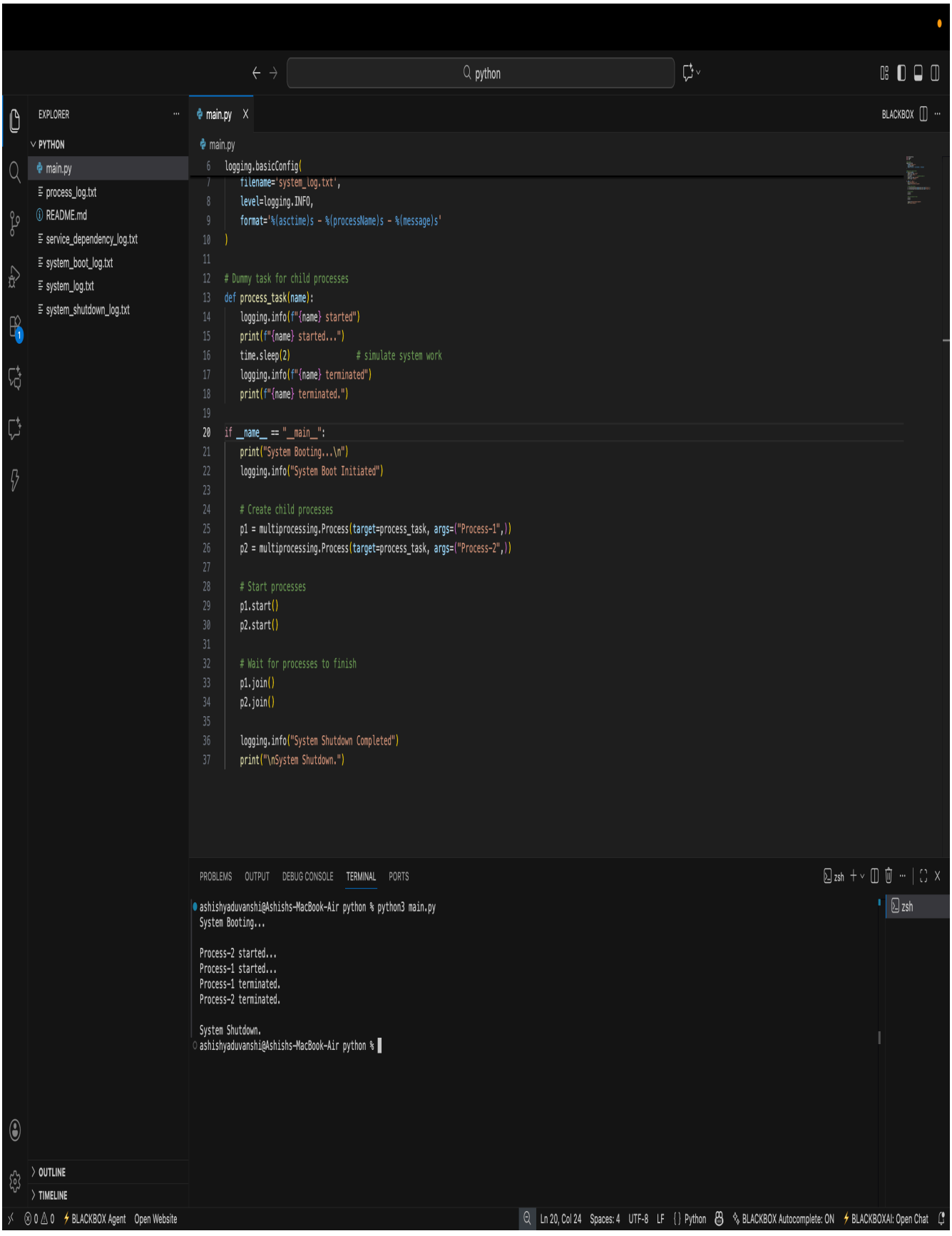
```
import subprocess

scripts = ['script1.py', 'script2.py', 'script3.py']

for script in scripts:
    print(f"Executing {script}...")
    subprocess.call(['python3', script])
```



Task 2: System Startup and Logging

Simulate system startup using Python by creating multiple processes and logging their start and end into a log file.

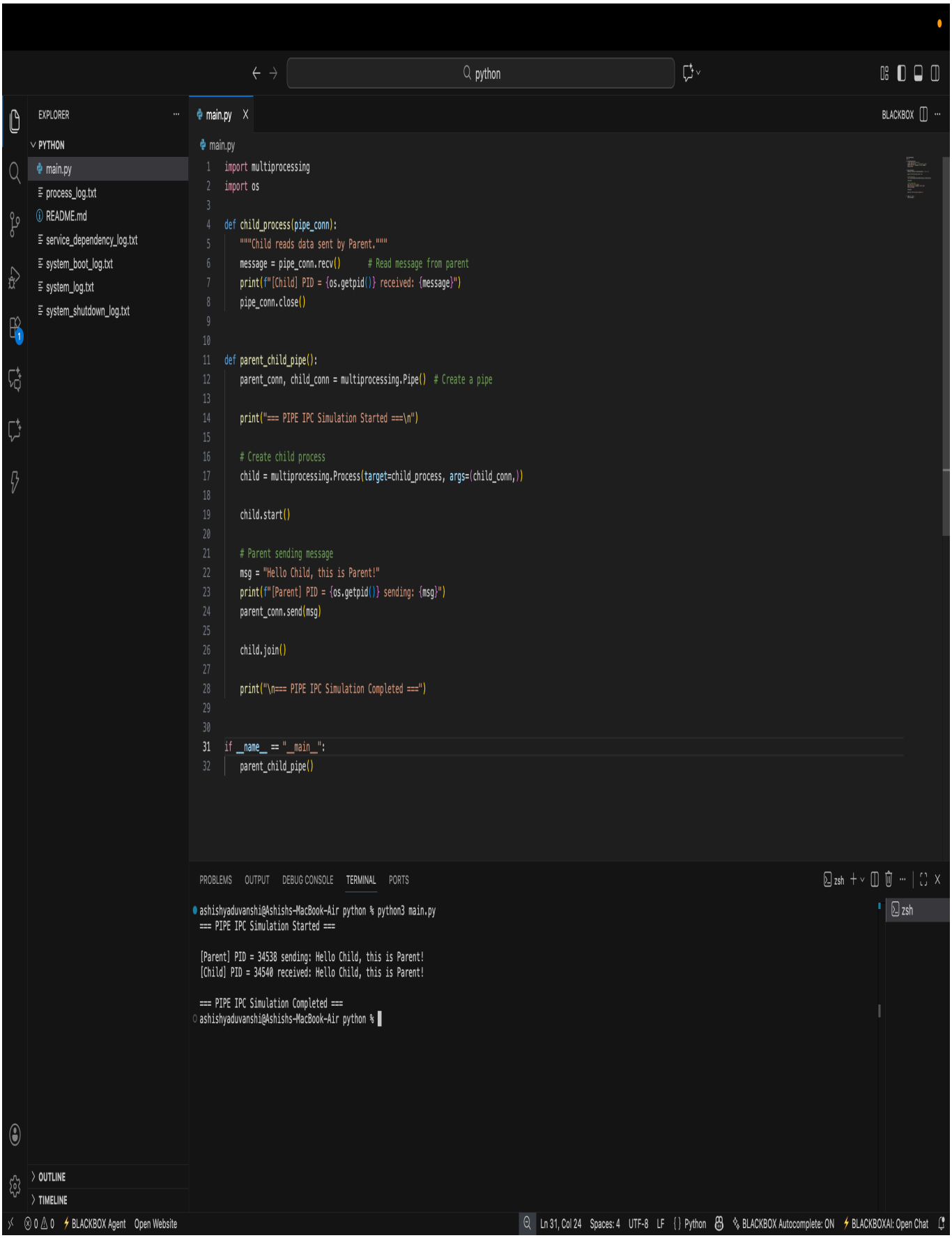
```
import multiprocessing
import logging
import time
logging.basicConfig(filename='system_log.txt', level=logging.INFO,
                    format='%(asctime)s - %(processName)s - %(message)s')
def process_task(name):
    logging.info(f"{name} started")
    time.sleep(2)
    logging.info(f"{name} terminated")
if __name__ == '__main__':
    print("System Booting...")
    p1 = multiprocessing.Process(target=process_task, args=("Process-1",))
    p2 = multiprocessing.Process(target=process_task, args=("Process-2",))
    p1.start()
    p2.start()
    p1.join()
    p2.join()
    print("System Shutdown.")
```


python
EXPLORER
PYTHON
main.py
process_log.txt
README.md
service_dependency_log.txt
system_boot_log.txt
system_log.txt
system_shutdown_log.txt
OUTLINE
TIMELINE
main.py
BLACKBOX
main.py
6 logging.basicConfig(
7 filename='system_log.txt',
8 level=logging.INFO,
9 format='%(asctime)s - %(processName)s - %(message)s'
10)
11
12 # Dummy task for child processes
13 def process_task(name):
14 logging.info(f"{name} started")
15 print(f"{name} started...")
16 time.sleep(2) # simulate system work
17 logging.info(f"{name} terminated")
18 print(f"{name} terminated.")
19
20 if __name__ == "__main__":
21 print("System Booting...\n")
22 logging.info("System Boot Initiated")
23
24 # Create child processes
25 p1 = multiprocessing.Process(target=process_task, args=("Process-1",))
26 p2 = multiprocessing.Process(target=process_task, args=("Process-2",))
27
28 # Start processes
29 p1.start()
30 p2.start()
31
32 # Wait for processes to finish
33 p1.join()
34 p2.join()
35
36 logging.info("System Shutdown Completed")
37 print("\nSystem Shutdown.")
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
zsh
ashishyaduvanshi@Ashishs-MacBook-Air python % python3 main.py
System Booting...
Process-2 started...
Process-1 started...
Process-1 terminated.
Process-2 terminated.
System Shutdown.
ashishyaduvanshi@Ashishs-MacBook-Air python %
0 0 BLACKBOX Agent Open Website
Ln 20, Col 24 Spaces: 4 UTF-8 LF Python BLACKBOX Autocomplete: ON BLACKBOXAI: Open Chat

Task 3: System Calls and IPC (Python - fork, exec, pipe)

Use system calls (fork(), exec(), wait()) and implement basic Inter-Process Communication using pipes in C or Python.

```
import os

r, w = os.pipe()

pid = os.fork()

if pid > 0:

    os.close(r)

    os.write(w, b"Hello from parent")

    os.close(w)

    os.wait()

else:

    os.close(w)

    message = os.read(r, 1024)

    print("Child received:", message.decode())

    os.close(r)
```


python
EXPLORER
PYTHON
main.py
process_log.txt
README.md
service_dependency_log.txt
system_boot_log.txt
system_log.txt
system_shutdown_log.txt
main.py
BLACKBOX
import multiprocessing
import os
def child_process(pipe_conn):
"""Child reads data sent by Parent."""
message = pipe_conn.recv() # Read message from parent
print(f"[Child] PID = {os.getpid()} received: {message}")
pipe_conn.close()
def parent_child_pipe():
parent_conn, child_conn = multiprocessing.Pipe() # Create a pipe
print("=== PIPE IPC Simulation Started ===\n")
# Create child process
child = multiprocessing.Process(target=child_process, args=(child_conn,))
child.start()
# Parent sending message
msg = "Hello Child, this is Parent!"
print(f"[Parent] PID = {os.getpid()} sending: {msg}")
parent_conn.send(msg)
child.join()
print("\n=== PIPE IPC Simulation Completed ===")
if __name__ == "__main__":
parent_child_pipe()
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
zsh
ashishyaduvanshi@Ashishs-MacBook-Air python % python3 main.py
=== PIPE IPC Simulation Started ===
[Parent] PID = 34538 sending: Hello Child, this is Parent!
[Child] PID = 34540 received: Hello Child, this is Parent!
=== PIPE IPC Simulation Completed ===
ashishyaduvanshi@Ashishs-MacBook-Air python %
OUTLINE
TIMELINE
BLACKBOX Agent
Open Website
Ln 31, Col 24
Spaces: 4
UTF-8
LF
Python
BLACKBOX Autocomplete: ON
BLACKBOXAI: Open Chat

Task 4: VM Detection and Shell Interaction

Create a shell script to print system details and a Python script to detect if the system is running inside a virtual machine.

#!/bin/bash

echo "Kernel Version:"

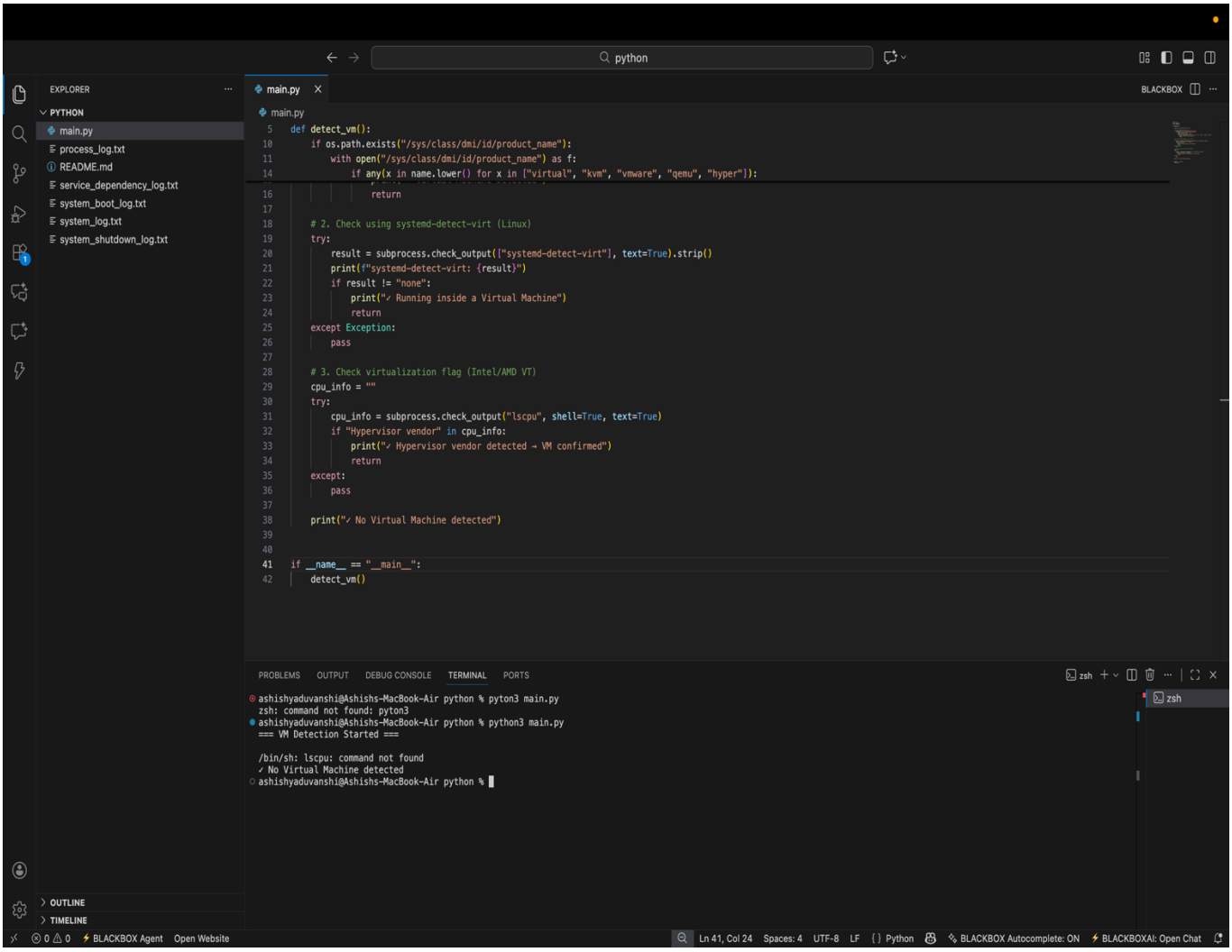
uname -r

echo "User:"

whoami

echo "Hardware Info:"

lscpu | grep 'Virtualization'



Task 5: CPU Scheduling Algorithms

Implement FCFS, SJF, Round Robin, and Priority Scheduling algorithms in Python to calculate WT and TAT.

Use existing Round Robin, FCFS, SJF, Priority scheduling Python codes from Lab 3)