



The screenshot shows a code editor window with several Python files listed in the sidebar: top_server.py, chat_client.py, chat_server.py, ping_program.py, hamming_code.py, util.py, jing.py, and top_client.py. The hamming_code.py file is open and contains code for a Hamming code implementation. The terminal tab shows the execution of the program and its output:

```
Microsoft Windows [Version 10.0.26990.7813]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Afrah H\Desktop\cn>python.exe "c:/Users/Afrah H/Desktop/cn/hamming_code.py"
Enter the data bits (e.g., 1011): 110110
Data with parity placeholders: 1100100
Encoded data (Hamming code): 1100110

Enter bit position to flip (0 for no error): 3
Received data: 1100010
✖ Error detected at bit position: 3
Corrected data: 1100110

C:\Users\Afrah H\Desktop\cn>
```

Result

A program to implement the Hamming Code was successfully written. The program demonstrated the ability to detect and correct a single-bit error in the transmitted data stream.

EXP:07	Flow control at Data Link Layer
--------	---------------------------------

Aim

To write a program to implement flow control at the data link layer using the **Sliding Window Protocol** and simulate the flow of frames from one node to another.

Algorithm / Procedure

1. **Define** the window size for the sender and receiver.
2. **Implement** the sender logic:
 - Maintain a sending window of sequence numbers.
 - Send frames within the window limit.
 - Start a timer for each unacknowledged frame.
3. **Implement** the receiver logic:
 - Maintain a receiving window.
 - Accept frames in order and send cumulative acknowledgments (ACKs).
 - Discard out-of-order frames (or buffer, depending on the specific protocol variant).

4. **Simulate** the flow of frames, including scenarios for successful transmission, lost frames, and lost ACKs, to demonstrate the flow control mechanism.

Code

```
import random
import time

# -----
# Sliding Window Protocol (Go-Back-N) Simulation
# -----


def sliding_window_simulation(total_frames, window_size):
    print("\n--- Sliding Window Protocol Simulation ---")
    print(f"Total Frames to Send: {total_frames}")
    print(f"Window Size: {window_size}\n")

    sent = 0 # Number of frames sent so far

    while sent < total_frames:
        # Determine frames in the current window
        window_end = min(sent + window_size, total_frames)
        current_window = list(range(sent + 1, window_end + 1))
        print(f"Sender: Sending frames {current_window}")

        # Simulate sending each frame in the window
        acked = True
        for frame in current_window:
            # Randomly simulate frame loss (20% chance)
            if random.random() < 0.2:
```

```
        print(f"Frame {frame} lost during transmission!")

        acked = False

        break

    else:

        print(f" Frame {frame} received successfully by Receiver")
        time.sleep(0.3)

if acked:

    # All frames acknowledged → Slide window forward

    print(f"Receiver: ACK {window_end} received by Sender")
    sent = window_end

else:

    # Go-Back-N retransmission

    print(f"Receiver: No ACK for Frame {frame}, retransmitting from Frame {frame}
onwards...")

    time.sleep(1)

    # Sender will retransmit from the lost frame

    sent = frame - 1

print("-" * 55)
time.sleep(0.5)

print("\nAll frames transmitted successfully!\n")

# -----
# MAIN PROGRAM
# -----
if __name__ == "__main__":
```

```
total_frames = int(input("Enter total number of frames to send: "))

window_size = int(input("Enter window size: "))
```

```
sliding_window_simulation(total_frames, window_size)
```

Output:

231501034
DEVESH D

COMPUTER NETWORKS
CS23532

Result

A program simulating the Sliding Window Protocol was successfully implemented. The simulation demonstrated effective flow control by managing the rate of frame transmission between the sender and receiver.