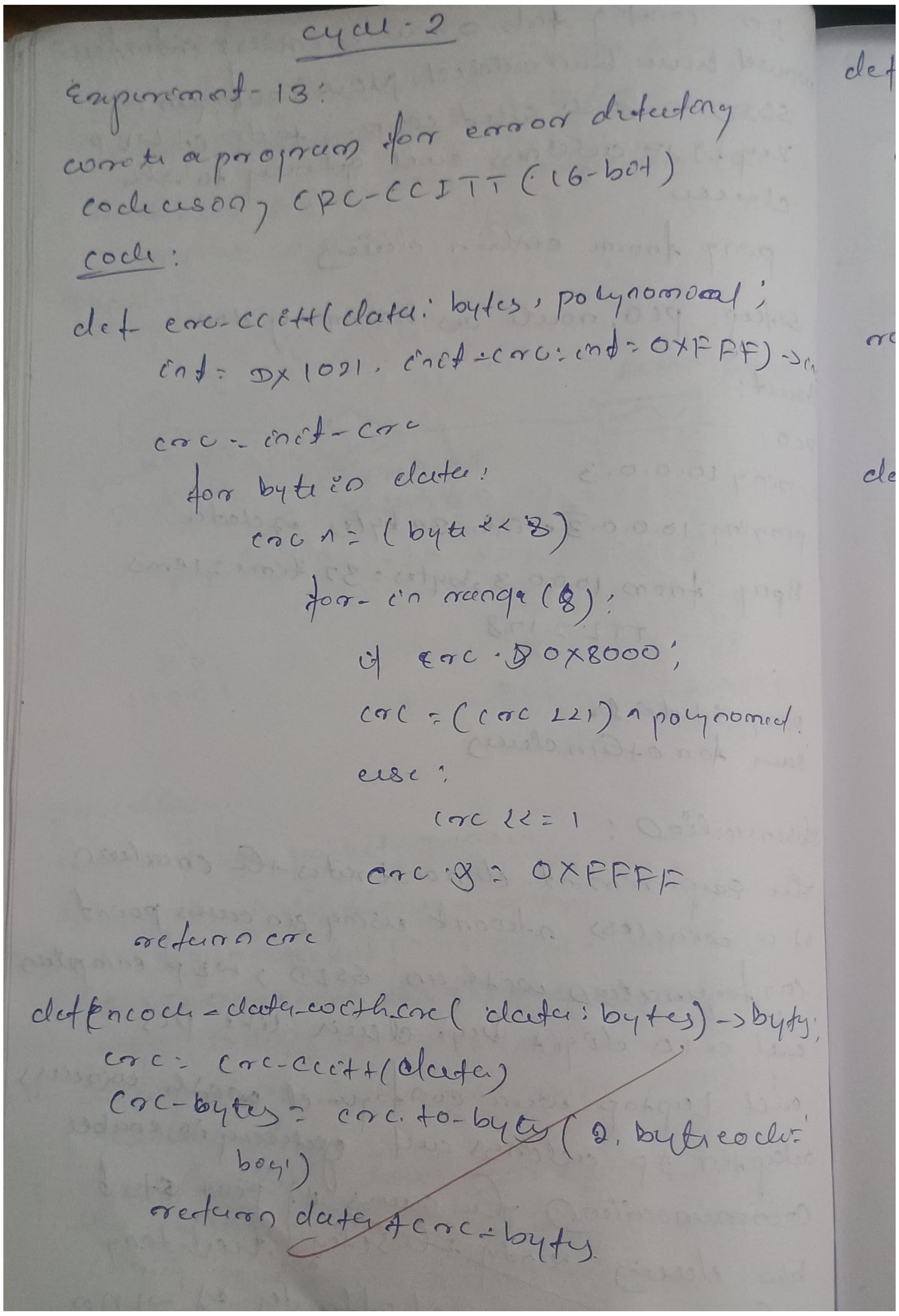
PART-B

Program 14
Write a program for error detecting code using CRC-CCITT (16-bits).

Code :

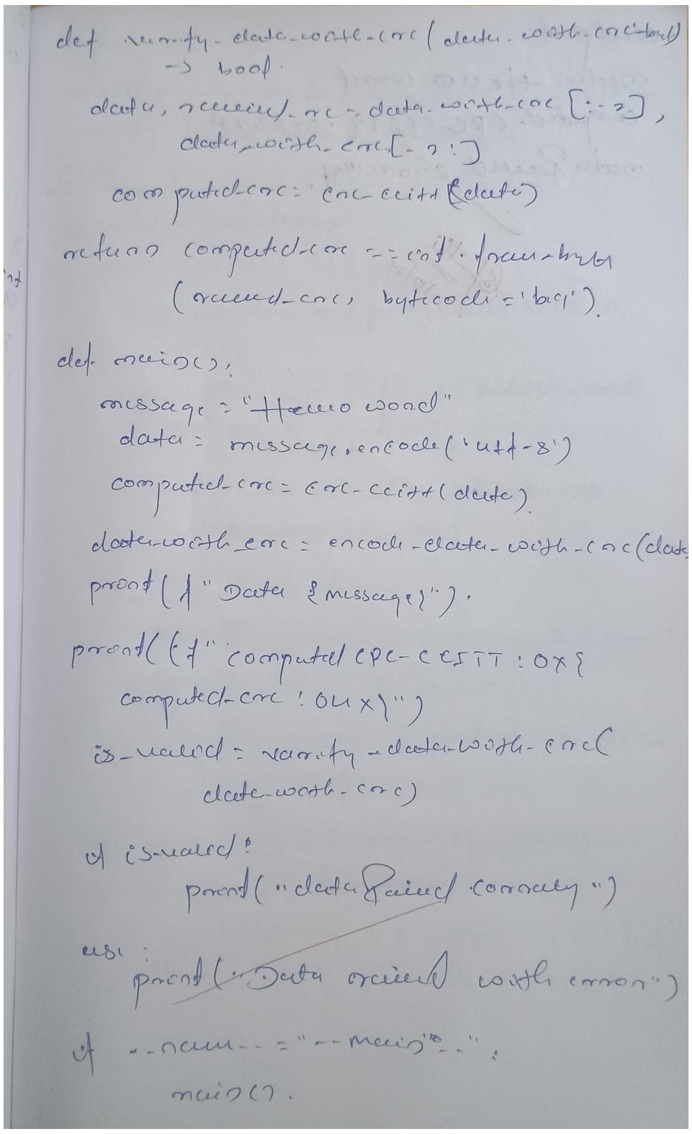


cycle - 2

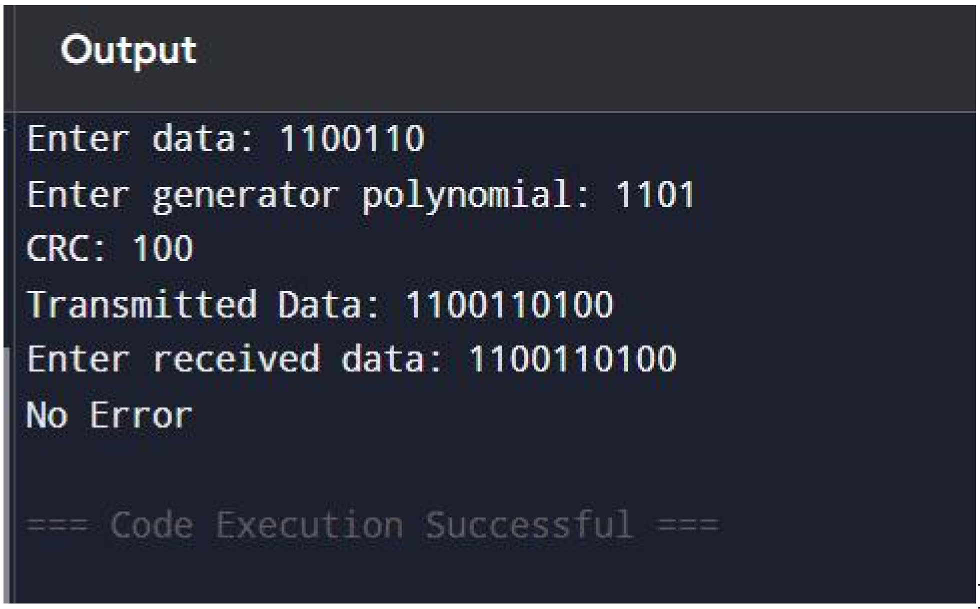
Experiment - 13:
write a program for error detecting
code using CRC-CCITT (16-bit)

code:

```
def crc-ccitt(data: bytes, polynomial;
        int= 0x1021, init-crc: int= 0xFFFF) -> in

    crc - init-crc
     for byte in data:
         crc ^= (byte << 8)
            for- in range (8):
                if crc & 0x8000:
                crc = ((crc <<1) ^ polynomial
                else:
                    crc <<= 1
                crc &= 0xFFFF
    return crc

def encode-data-with-crc( data: bytes) -> bytes:
   crc = crc-ccitt(data)
   crc-bytes = crc.to-bytes( 2, byteorder='
        big')
   return data + crc-bytes
```



```
def verify_data_with_crc(data_with_crc:bytes)
        -> bool:
    data, received_crc = data_with_crc [:-2],
            data_with_crc[-2:]
    computed_crc = crc_ccitt(data)
    return computed_crc == int.from_bytes
            (received_crc, byteorder='big').

def main():
    message = "Hello world"
    data = message.encode('utf-8')
    computed_crc = crc_ccitt(data).
    data_with_crc = encode_data_with_crc(data
    print(f"Data {message}").
    print((f"computed CRC-CCITT : 0x{
        computed_crc : 04x}")
    is_valid = verify_data_with_crc(
            data_with_crc)
    if is_valid:
            print("data received correctly")
    else :
        print("Data received with error")
if __name__ == "__main__":
    main().
```



```
Output

Enter data: 1100110
Enter generator polynomial: 1101
CRC: 100
Transmitted Data: 1100110100
Enter received data: 1100110100
No Error


=== Code Execution Successful ===
```

]