

Assignment 8

ELP - 718 Telecommunication Software Laboratory

Yamini Singh

2019JTM2170

2019-2021

A report presented for the assignment on
Python basics



**Bharti School Of
Telecommunication Technology and Management
IIT Delhi
India**

September 18th, 2019

Contents

1	Problem Statement-1	4
1.1	Parity Check	4
1.2	Bit-Oriented Framing	4
1.3	Input Format	4
1.4	Output Format	4
1.5	Assumptions	4
1.6	Algorithm and Implementation	4
1.7	Program Structure	6
1.8	Screenshots	7
2	Problem Statement-2	8
2.1	Problem Statement	8
2.2	Constraints	8
2.3	Assumptions	8
2.4	Input Format	8
2.5	Output Format	9
2.6	Algorithm and Implementation	10
2.7	Program Structure	11
2.8	Screenshots	12
3	Appendix	14
3.1	Code for ps1	14
3.2	Code for ps2.1	15

List of Figures

1	Problem 1 Flowchart	6
2	Screenshot 1	7
3	Problem 2 Flowchart	11
4	Screenshot 2-1	12
5	Screenshot 2-2	13

1 Problem Statement-1

1.1 Parity Check

The simplest way of error detection is to append a single bit, called a parity check, to a string of data bits. This parity check bit has the value 1 if the number of 1s in the bit string is even and has the value 0 otherwise, i.e., Odd Parity Check.

1.2 Bit-Oriented Framing

Data Link Layer needs to pack bits into frames so that each frame is distinguishable from another. Frames can be fixed or variable size. In variable size framing, we define the end of the frame using a bit-oriented approach. It uses a special string of bits, called a flag for both idle fills and to indicate the beginning and the ending of frames.

The bit stuffing rule is to insert a 0 after each appearance of 010 in the original data. The string 0101 is used as the bit string or flag to indicate the end of the frame.

1.3 Input Format

Enter binary bit data that has to be transmitted. **Sample Input** 010101110100101

1.4 Output Format

Print binary bit data with parity bit. Print the modified string that is to be transmitted

Sample Output Parity bit data : 0101011101001011 Transmitting data: 01001011101000100110101

1.5 Assumptions

1. Input bit stream is continuous
2. Parity bit data is evaluated as per the problem statement

1.6 Algorithm and Implementation

1. Take input bit stream as a string
2. Convert string to a list
3. Check for even or odd parity
4. If even parity,append 1
5. If odd parity,append 0

6. Display the parity stream output
7. Convert parity bit string to list
8. Check for 010 in the in the parity bit stream
9. Insert 0 after 010
10. Append 0101 at the end of bit stream
11. Print transmitting bit stream

1.7 Program Structure

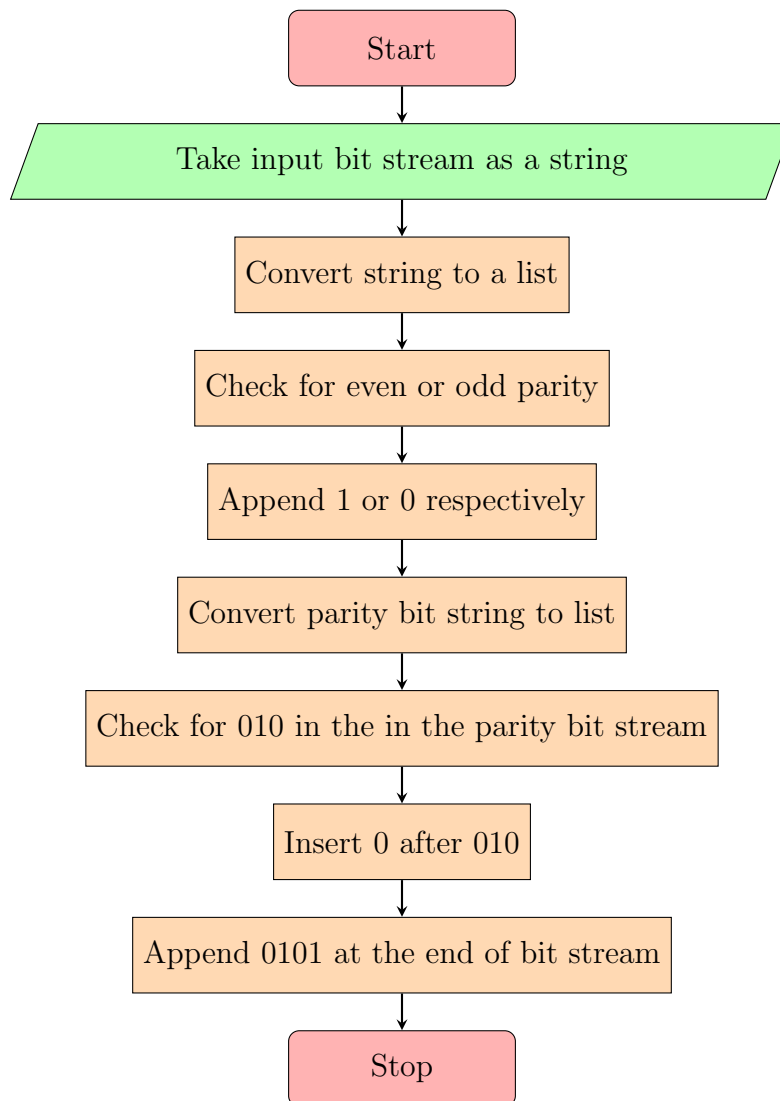
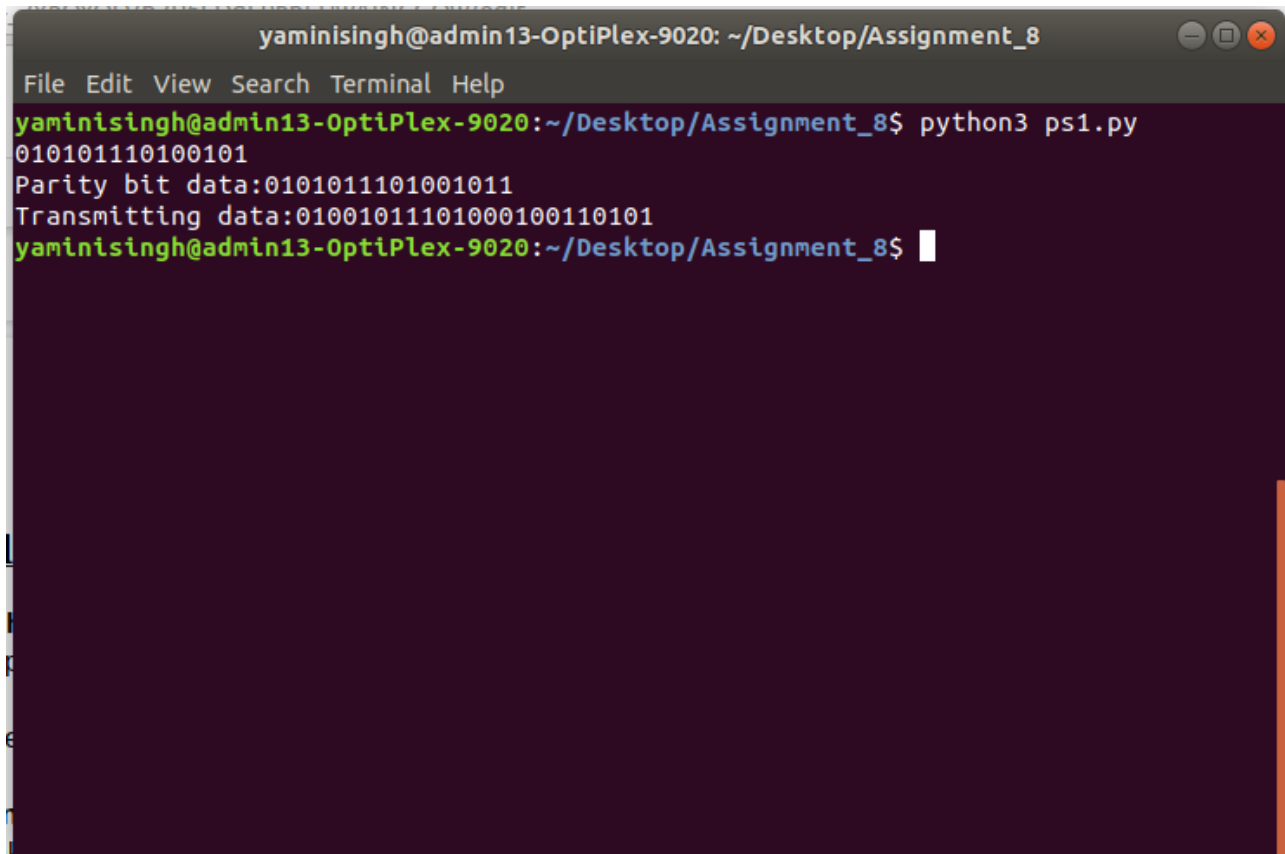


Figure 1: Problem 1 Flowchart

1.8 Screenshots



The screenshot shows a terminal window titled "yaminisingh@admin13-OptiPlex-9020: ~/Desktop/Assignment_8". The terminal has a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". The prompt is "yaminisingh@admin13-OptiPlex-9020:~/Desktop/Assignment_8\$". The user has entered the command "python3 ps1.py". The output of the script is displayed on the following lines: "010101110100101", "Parity bit data:0101011101001011", and "Transmitting data:01001011101000100110101". The prompt is now "yaminisingh@admin13-OptiPlex-9020:~/Desktop/Assignment_8\$" with a cursor.

```
yaminisingh@admin13-OptiPlex-9020: ~/Desktop/Assignment_8
File Edit View Search Terminal Help
yaminisingh@admin13-OptiPlex-9020:~/Desktop/Assignment_8$ python3 ps1.py
010101110100101
Parity bit data:0101011101001011
Transmitting data:01001011101000100110101
yaminisingh@admin13-OptiPlex-9020:~/Desktop/Assignment_8$
```

Figure 2: Screenshot 1

2 Problem Statement-2

2.1 Problem Statement

3X3 Numeric Tic-Tac-Toe (Use numbers 1 to 9 instead of Xs and Os) One player plays with the odd numbers (1, 3, 5, 7, 9) and the other player plays with the even numbers (2,4,6,8). All numbers can be used only once. The player who puts down 15 points in a line wins (sum of 3 numbers). Always Player with odd numbers starts the game. Once a line contains two numbers whose sum is 15 or greater, there is no way to complete that line, although filling in the remaining cells might be necessary to complete a different line. Note Line can be horizontal, vertical or diagonal

2.2 Constraints

- $1 \leq \text{Position} \leq 9$
- $1 \leq \text{Number} \leq 9$

2.3 Assumptions

- Player enters data in a single line
- Numbers are between 1 to 9 only
- It is a 3*3 tic tac

2.4 Input Format

- Print Welcome to the Game!
- Print whether it is Player 1s or Player 2s chance.
- Get the position and number to be entered from the user.
- Show tic tac toe with data.
- Continue till the game gets draw or some player wins and show the result.
- Ask the user whether to continue for the next game or exit.

2.5 Output Format

Welcome to the Game!

Player 1s chance

Enter the position and number to be entered: 5,3

	3	

Player 2s chance

Enter the position and number to be entered: 7,4

	3	
4		

2.6 Algorithm and Implementation

1. Create a user defined function update for printing tic-tac
2. Create a 2d array and convert it into 1 d array
3. Enter the game
4. Take input from players about position and number
5. Check if it is a valid number
6. Check for winning conditions
7. Update the array and print tic tac at each step
8. If a player wins,exit the game

2.7 Program Structure

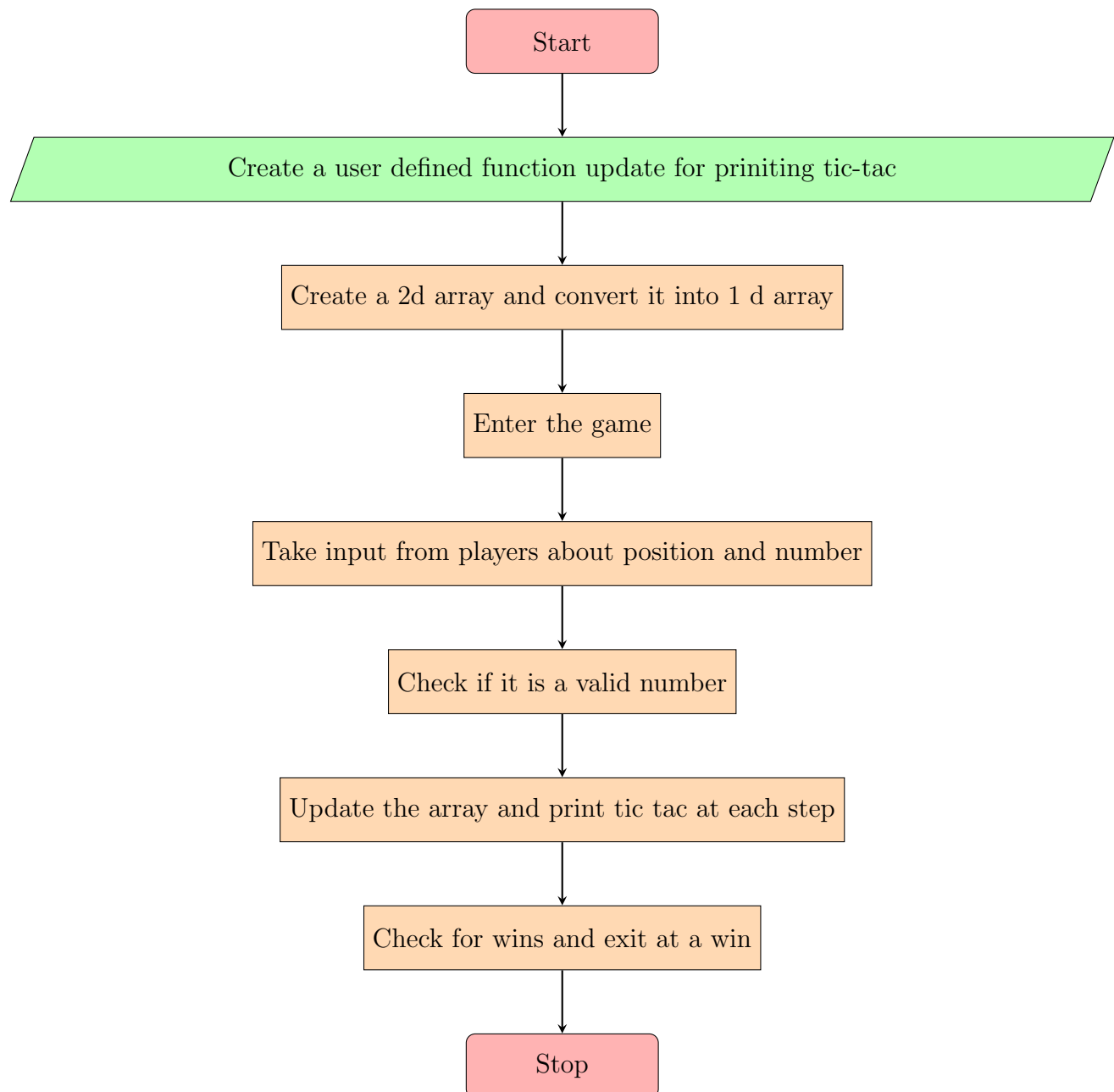


Figure 3: Problem 2 Flowchart

;

2.8 Screenshots

```
ValueError: not enough values to unpack (expected 2, got 1)
yaminisingh@admin13-OptiPlex-9020:~/Desktop/Assignment_8$ python3 ps2.py
Welcome to the game!

Player 1's chance

Enter the position and number to be entered:
5 3
[0, 0, 0]
[0, 3, 0]
[0, 0, 0]

Player 2's chance

Enter the position and number to be entered:
2 4
[0, 4, 0]
[0, 3, 0]
[0, 0, 0]

Player 1's chance

Enter the position and number to be entered:
1 7
[7, 4, 0]
[0, 3, 0]
[0, 0, 0]

Player 2's chance

Enter the position and number to be entered:
3 8
[7, 4, 8]
[0, 3, 0]
[0, 0, 0]

Player 1's chance

Enter the position and number to be entered:
4 6
Enter correct number!!

Player 1's chance

Enter the position and number to be entered:
4 9
[7, 4, 8]
[9, 3, 0]
[0, 0, 0]

Player 2's chance

Enter the position and number to be entered:
█
```

Figure 4: Screenshot 2-1

```
yaminisingh@admin13-OptiPlex-9020:~/Desktop/Assignment_8$ python3 ps2.py
Welcome to the game!

Player 1's chance

Enter the position and number to be entered:
1 7
[7, 0, 0]
[0, 0, 0]
[0, 0, 0]

Player 2's chance

Enter the position and number to be entered:
4 4
[7, 0, 0]
[4, 0, 0]
[0, 0, 0]

Player 1's chance

Enter the position and number to be entered:
2 3
[7, 3, 0]
[4, 0, 0]
[0, 0, 0]

Player 2's chance

Enter the position and number to be entered:
5 6
[7, 3, 0]
[4, 6, 0]
[0, 0, 0]

Player 1's chance

Enter the position and number to be entered:
3 5
[7, 3, 5]
[4, 6, 0]
[0, 0, 0]
Player 1 is the winner
```

Figure 5: Screenshot 2-2

3 Appendix

Github: https://github.com/2019JTM2170/Assignment_8

3.1 Code for ps1

```

1
2 count1=0 #initializing counters
3 count2=0
4 start=0
5 ps=""
6
7 #####PARITY#####
8
9 st=input() #Take input bit stream as a string
10 l=[char for char in st] #converting string to a list
11 for c in l:
12     if c=='1':
13         count1+=1;
14     if count1%2==0: #checking for parity
15         l.append('1')
16     else:
17         l.append('0')
18     print("Parity bit data:",end='')
19 ps=ps.join(l) #printing parity bit stream
20 print(ps)
21
22
23 #####FRAMING#####
24
25 i=0
26 new=""
27 l1=[char for char in ps] #converting parity bit string to list
28 while(i<len(l1)):
29     if l1[i]=='0' and l1[i+1]=='1' and l1[i+2]=='0': #checking for "010"
30         l1.insert(i+3,'0') #inserting 0 after 010
31         i=i+4
32     else:
33         i=i+1
34
35 l1.append('0') #appending 0101 at the end of bit stream
36 l1.append('1')
37 l1.append('0')
38 l1.append('1')
39
40 print("Transmitting data:",end='')
41 new=new.join(l1) #printing transmitting bit stream
42 print(new)

```

3.2 Code for ps2.1

```

1 import sys
2 def update(new): #user defined function to show the updated tic-tac
3     print(new[0:3])
4     print(new[3:6])
5     print(new[6:9])
6
7     flag=1
8     new=[]
9     rows, cols = (3, 3)#rows and columns are 3,3
10    arr = [[0]*cols]*rows #defining 2d matrix of order 3
11
12    new=[j for sub in arr for j in sub] #converting 2d array to 1d array
13    print("Welcome to the game!\n") #entering the game
14    while(flag!=0):
15
16
17    #####Player 1#####
18
19    print("\nPlayer 1's chance\n") #Player 1 chance
20    print("Enter the position and number to be entered:")
21    p,n=input().split() #enter position and number
22    p=int(p)
23    n=int(n)
24    if p<1 and p>9:
25        print("Enter correct position!!\n") #Checking position
26        continue
27    if n==2 or n==4 or n==6 or n==8 or n<1 or n>9: #Checking number
28        print("Enter correct number!!\n")
29        continue
30    new[int(p)-1]=int(n) #updating 1d array
31    update(new) #calling to print 2d array
32    if (new[0]+new[3]+new[6]==15) or (new[1]+new[4]+new[7]==15) or (new[2]+new
        ↳ [5]+new[8]==15) or (new[0]+new[1]+new[2]==15) or (new[3]+new[4]+new
        ↳ [5]==15) or (new[6]+new[7]+new[8]==15) or (new[0]+new[4]+new[8]==15) or
        ↳ (new[2]+new[4]+new[6]==15):
33    print("Player 1 is the winner\n")
34    flag=0
35    sys.exit()
36
37
38    #####Player 2#####
39
40    print("\nPlayer 2's chance\n") #Player 2 chance
41    print("Enter the position and number to be entered:") #enter position and
        ↳ number
42    p,n=input().split()
43    p=int(p)
44    n=int(n)

```

```
45 if p<1 and p>9: #checking position
46 print("Enter correct position!!\n")
47 continue
48 if n==1 or n==3 or n==5 or n==7 or n==9 or n<1 or n>9:#checking number
49 print("Enter correct number!!\n")
50 continue
51 new[int(p)-1]=int(n) #updating 1d array
52 update(new) #calling to print 2d array
53 if (new[0]+new[3]+new[6]==15) or (new[1]+new[4]+new[7]==15) or (new[2]+new
    ↳ [5]+new[8]==15) or (new[0]+new[1]+new[2]==15) or (new[3]+new[4]+new
    ↳ [5]==15) or (new[6]+new[7]+new[8]==15) or (new[0]+new[4]+new[8]==15) or
    ↳ (new[2]+new[4]+new[6]==15):
54 print("Player 2 is the winner\n")
55 flag=0
56 sys.exit()
```


References

- [1] Geeksforgeeks. *2d array*. <https://www.geeksforgeeks.org/python-ways-to-flatten-a-2d-list/>.
- [2] Programiz. *insert*. <https://www.programiz.com/python-programming/methods/list/insert>.
- [3] thispointer. *numpy*. <https://thispointer.com/python-numpy-select-rows-columns-by-index-from-a-2d-ndarray-multi-dimension/>.