Assignment 8

ELP - 718 Telecommunication Software Laboratory

Yamini Singh

2019JTM2170

2019-2021

A report presented for the assignment on

Python basics

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

Bharti School Of

Telecommunication Technology and Management

IIT Delhi

India

September 18th, 2019

Contents

List of Figures

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

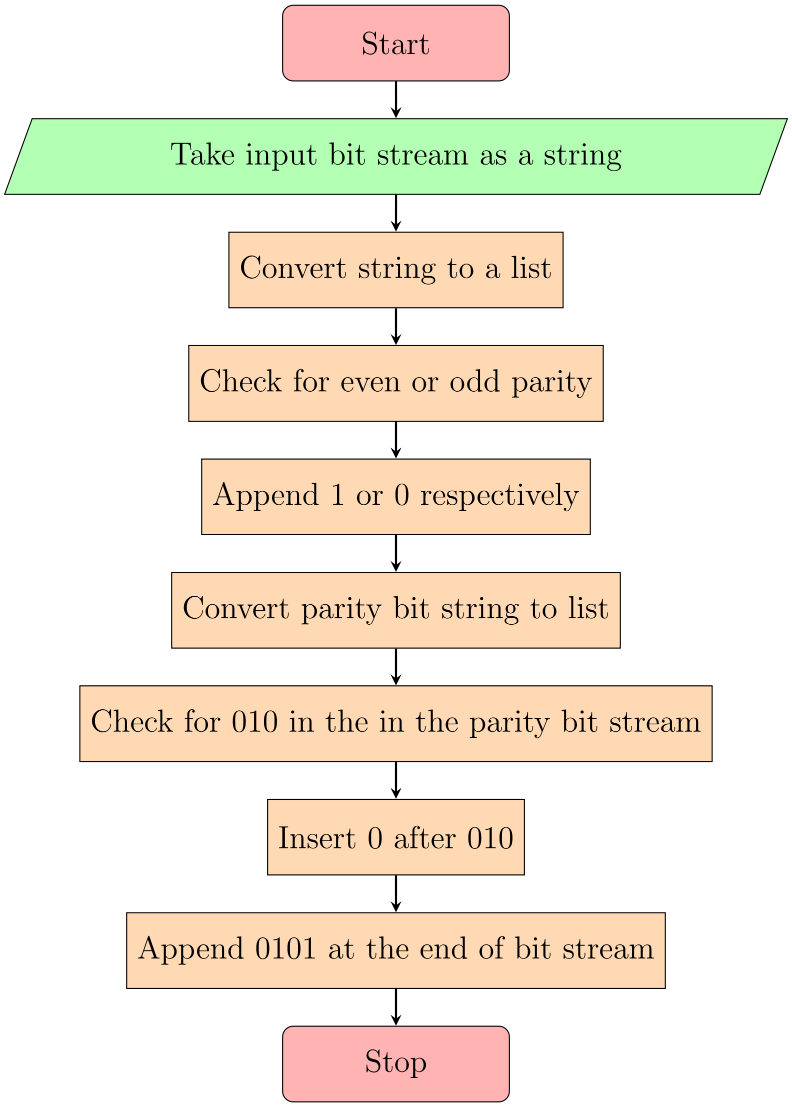
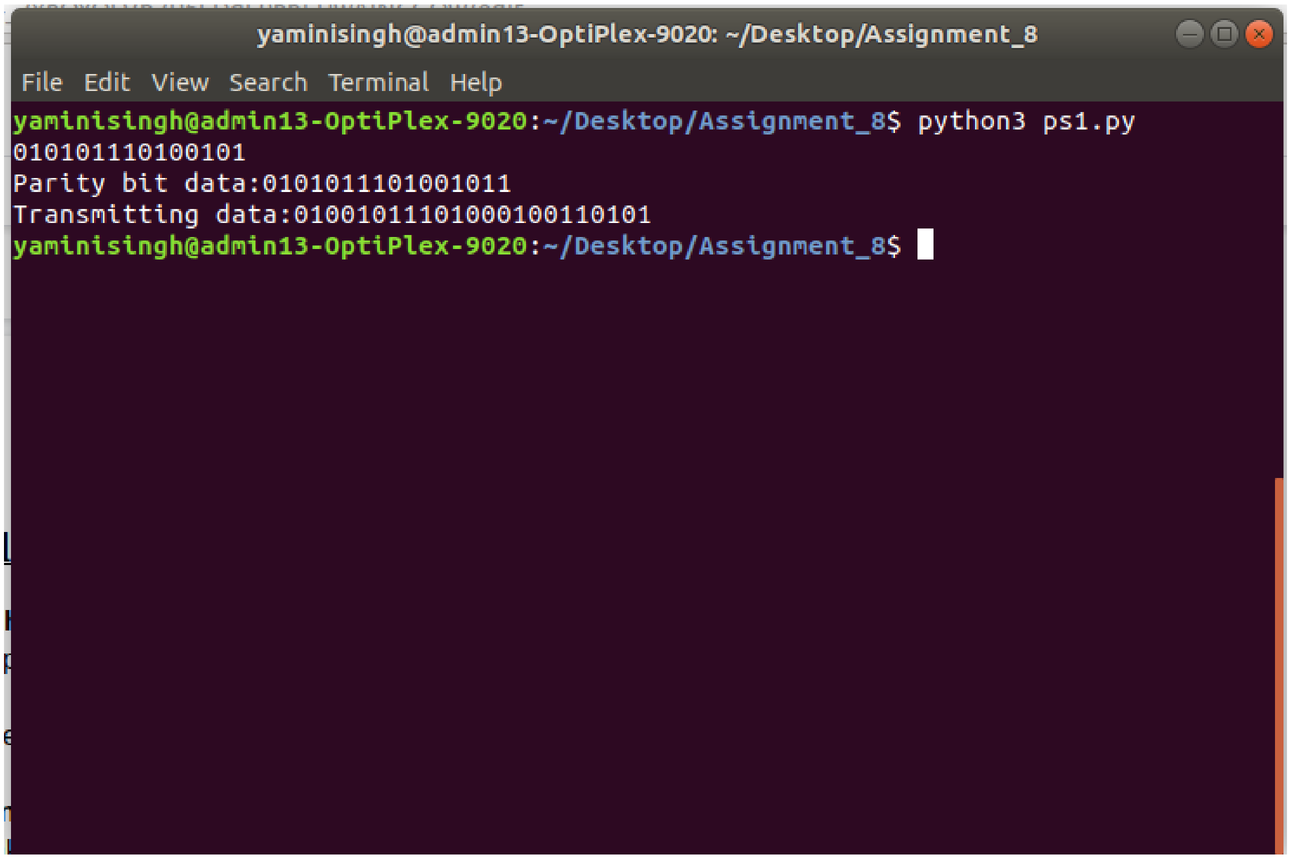

Start
Take input bit stream as a string
Convert string to a list
Check for even or odd parity
Append 1 or 0 respectively
Convert parity bit string to list
Check for 010 in the in the parity bit stream
Insert 0 after 010
Append 0101 at the end of bit stream
Stop

Figure 1: Problem 1 Flowchart

1.8 Screenshots


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<=Position<=9
- 1<=Number<=9

2.3 Assumptions

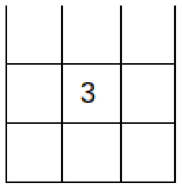
- Player enters data in a single line
- Numbers are between 1 to 9 only
- It is a 3*3 tic tac

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.

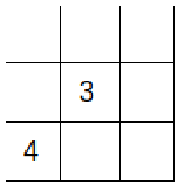
2.5 Output Format

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 priniting 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

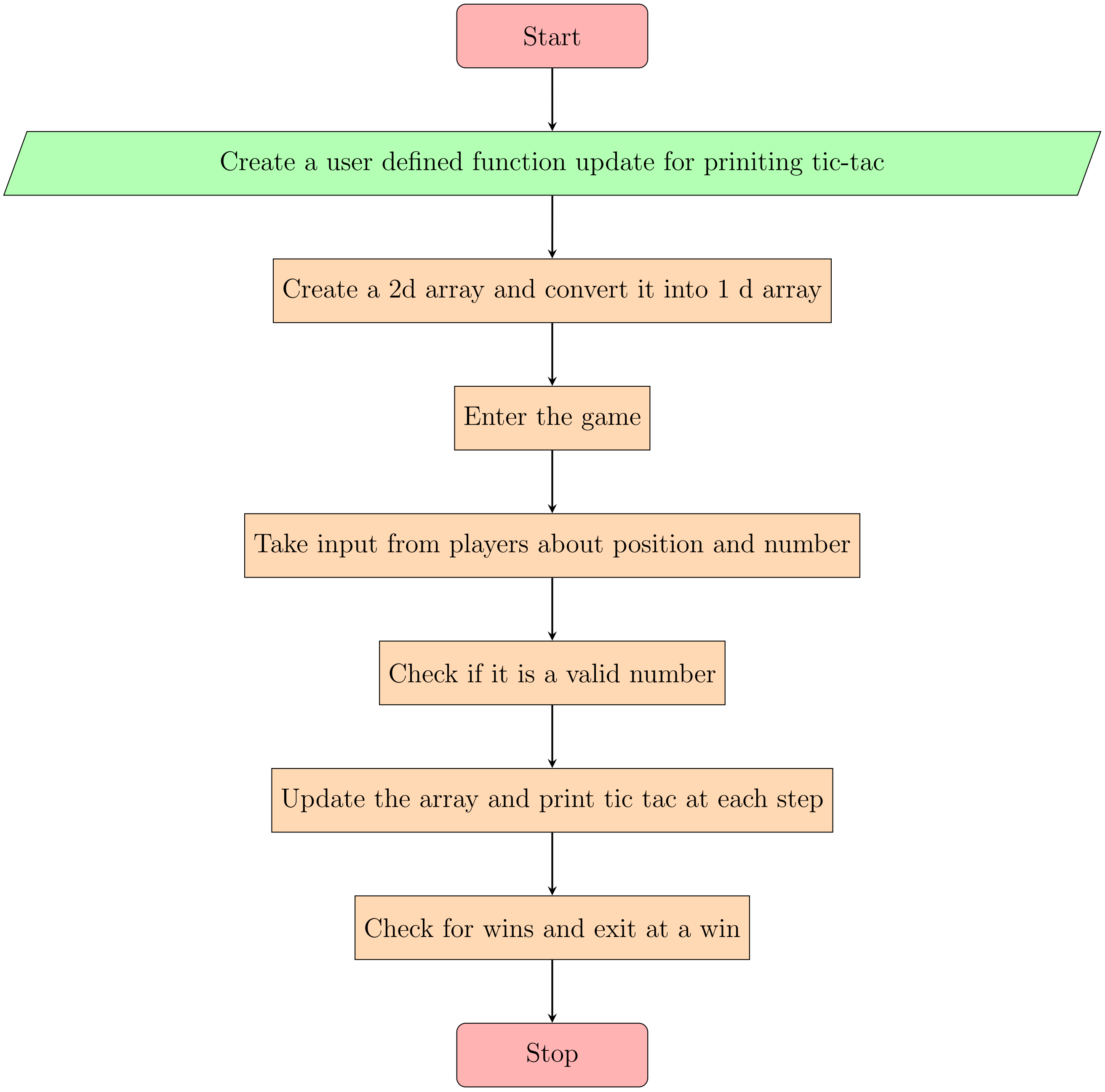
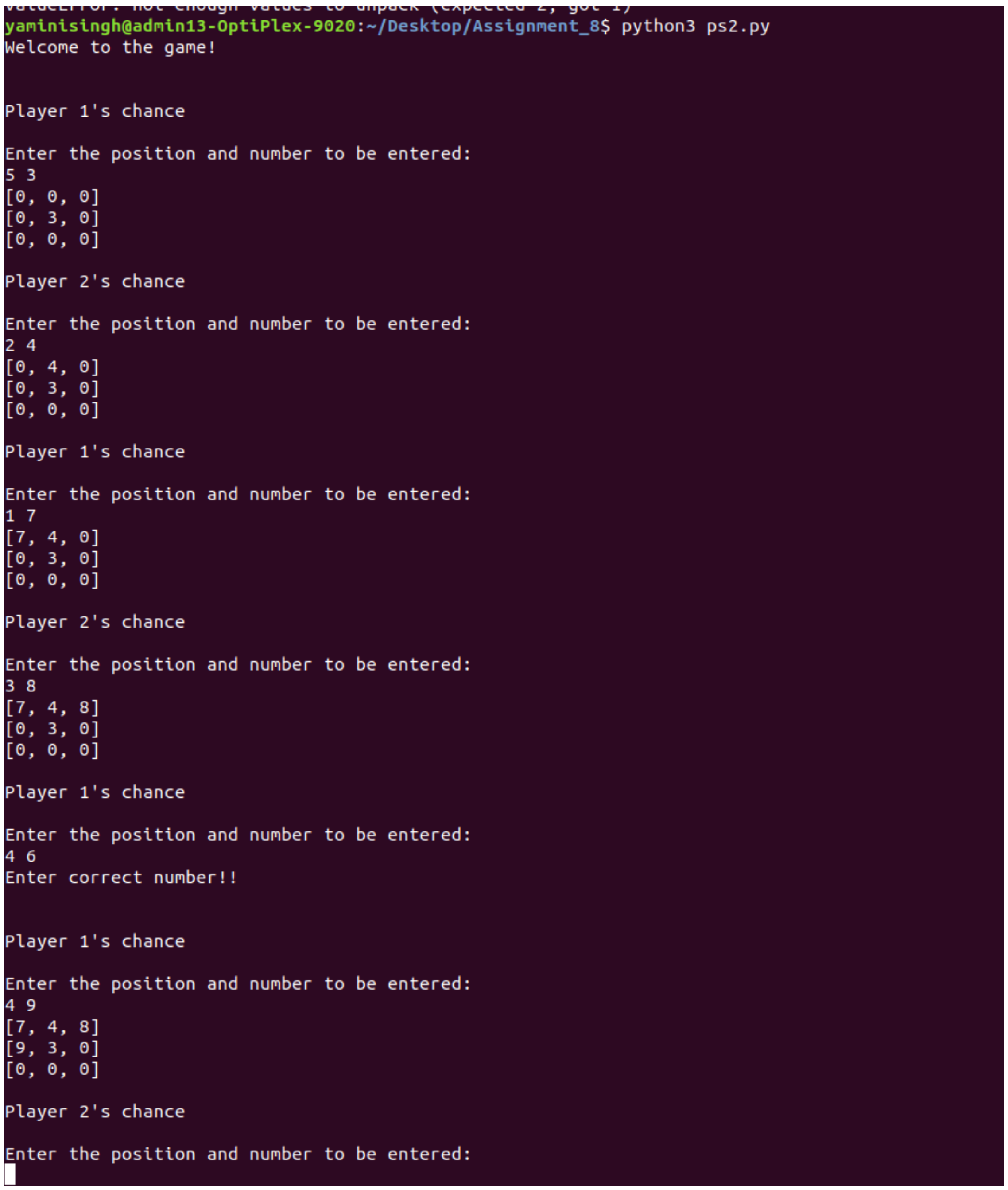

Start
Create a user defined function update for priniting tic-tac
Create a 2d array and convert it into 1 d array
Enter the game
Take input from players about position and number
Check if it is a valid number
Update the array and print tic tac at each step
Check for wins and exit at a win
Stop

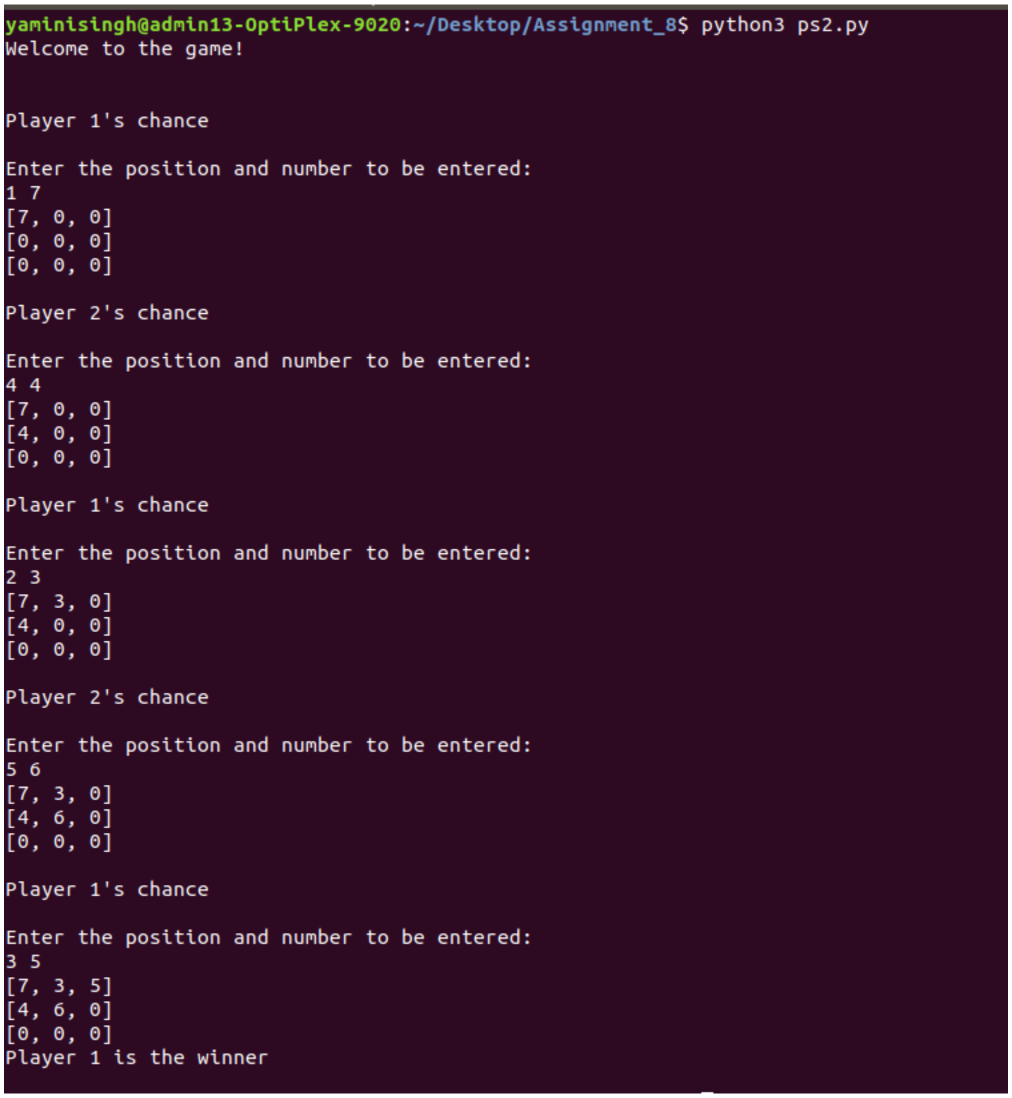
Figure 3: Problem 2 Flowchart
;

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

3.1 Code for ps1

```

count1=0 #initializing counters
count2=0
start=0
ps=""

#########PARITY##########################

st=input() #Take input bit stream as a string
l=[char for char in st] #converting string to a list
for c in l:
if c=='1':
count1+=1;
if count1%2==0: #checking for parity
l.append('1')
else:
l.append('0')
print("Parity bit data:",end='')
ps=ps.join(l) #priniting parity bit stream
print(ps)


########FRAMING####################

i=0
new=""
l1=[char for char in ps] #converting parity bit string to list
while(i<len(l1)):
if l1[i]=='0' and l1[i+1]=='1' and l1[i+2]=='0': #checking for "010"
l1.insert(i+3,'0') #inserting 0 after 010
i=i+4
else:
i=i+1

l1.append('0') #appending 0101 at the end of bit stream
l1.append('1')
l1.append('0')
l1.append('1')

print("Transmitting data:",end='')
new=new.join(l1) #printing transmitting bit stream
print(new)
```

3.2 Code for ps2.l

```
import sys
def update(new): #user defined function to show the updated tic-tac
print(new[0:3])
print(new[3:6])
print(new[6:9])

flag=1
new=[]
rows, cols = (3, 3)#rows and columns are 3,3
arr = [[0]*cols]*rows #defining 2d matrix of order 3

new=[j for sub in arr for j in sub] #converting 2d array to 1d array
print("Welcome to the game!\n") #entering the game
while(flag!=0):


########Player 1##############################################

print("\nPlayer 1's chance\n") #Player 1 chance
print("Enter the position and number to be entered:")
p,n=input().split() #enter position and number
p=int(p)
n=int(n)
if p<1 and p>9:
print("Enter correct position!!\n") #Checking position
continue
if n==2 or n==4 or n==6 or n==8 or n<1 or n>9: #Checking number
print("Enter correct number!!\n")
continue
new[int(p)-1]=int(n) #updating 1d array
update(new) #calling to print 2d array
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):
print("Player 1 is the winner\n")
flag=0
sys.exit()


#################Player 2##############################################

print("\nPlayer 2's chance\n") #Player 2 chance
print("Enter the position and number to be entered:") #enter position and
    ↪ number
p,n=input().split()
p=int(p)
n=int(n)
```

```
if p<1 and p>9: #checking position
print("Enter correct position!!\n")
continue
if n==1 or n==3 or n==5 or n==7 or n==9 or n<1 or n>9:#checking number
print("Enter correct number!!\n")
continue
new[int(p)-1]=int(n) #updating 1d array
update(new) #calling to print 2d array
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):
print("Player 2 is the winner\n")
flag=0
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/`.