



**Bharti School of
Telecommunication Technology and Management
IIT DELHI**

Assignment-6

ELP-718 Telecom software Labaratory

Basics of python

Sujeet Mandloi

2017JTM2186

September 12, 2017

Contents

1	Problem Statement-1	2
	Input Format	2
	Input Format	2
	Screenshots	3
	Source Code problem 1	3
2	Problem Statement-2	5
	Constraints	5
	Assumptions	5
	Implemenation	5
	Input Format	5
	output Format	5
	Terminal	5
	sample Output	5
	Source Code	6

1 Problem Statement-1

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 number of 1s in the bit string is even and has the value 0 otherwise, i.e., Odd Parity Check.

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 end of frame using bit oriented approach. It uses a special string of bits, called a flag for both idle fill and to indicate the beginning and the ending of frames. The string 0101 is used as the bit string or flag to indicate the end of the frame. The bit stuffing rule is to insert a 0 after each appearance of 010 in the original data. In addition, if the frame ends in 01, a 0 would be stuffed after the 1st 0 in the actual terminating string 0101.

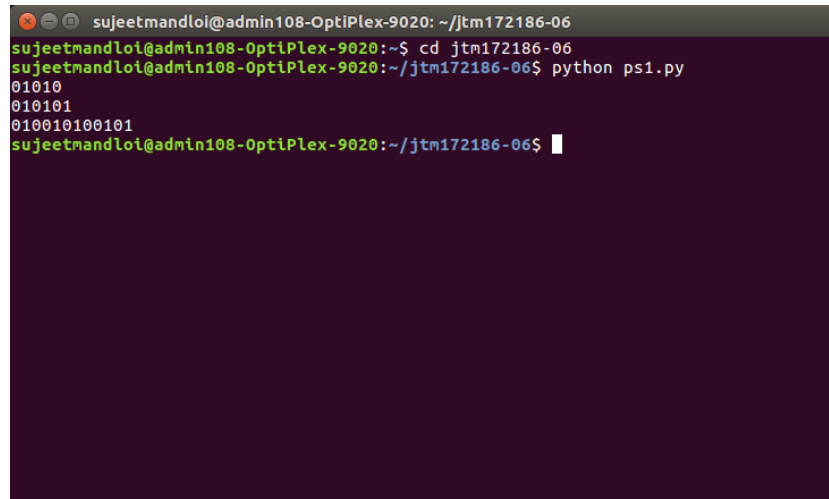
Input Format

Enter binary bit data which has to be transmitted.

Input Format

Print binary bit data with parity bit. Print the modified string received at the other end.

Screenshots

A screenshot of a terminal window with a dark background. The window title is 'sujeetmandloi@admin108-OptiPlex-9020: ~/jtm172186-06'. The prompt is 'sujeetmandloi@admin108-OptiPlex-9020:~\$'. The user enters 'cd jtm172186-06'. The prompt changes to 'sujeetmandloi@admin108-OptiPlex-9020:~/jtm172186-06\$'. The user enters 'python ps1.py'. The script outputs three lines of binary strings: '01010', '010101', and '010010100101'. The prompt returns to 'sujeetmandloi@admin108-OptiPlex-9020:~/jtm172186-06\$' with a cursor.

Source Code problem 1

```
s=raw_input(' ')    #input from user

i=s.count('1')      # counts the no of 1's

if i%2==0 :         # check whether even no of ones or odd if even then
    s=s+"1"
    print(s)
    k=s.replace('010','0100',s.count('010')) # insert 0 when 010 is
    if s[-2:]=="01": #if last two bit are 01 then replace with 0
        k=k[:-2]+"0100101"
        print(k)
    else :
        k=k+"0101"
        print(k)
```

```

else :
    print(s)
    s.replace('010','0100',s.count('010'))
    if s[-2:]=="01":
        k=s[:-2]+"0100101"
        print(k)
    else :
        k=k+"0101"
        print(k)

```

2 Problem Statement-2

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 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 start 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 cell might be necessary to complete a different line. Note Line can be horizontal, vertical or diagonal

Constraints

- $1 \leq Position \leq 9$
- $1 \leq Number \leq 9$
- $1 \leq Sum \leq 9$

Assumptions

Implementation

Input Format

C

output Format

Terminal

sample Output

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

Source Code

```
choices = [] # creating a list

for x in range (0, 9) :
    choices.append(str(x + 1))

playerOneTurn = True # starting with player 1 turn
winner = False

def printBoard() : # defining a function print Board
    print( '\n -----')
    print( '|' + choices[0] + '|' + choices[1] + '|' + cho
    print( ' -----')
    print( '|' + choices[3] + '|' + choices[4] + '|' + cho
    print( ' -----')
    print( '|' + choices[6] + '|' + choices[7] + '|' + cho
    print( ' -----\n')

while not winner :
    printBoard()

    if playerOneTurn :
        print( "Player 1:")
    else :
        print( "Player 2:")

    try:
        choice = int(input(">> "))
        choice = int(input(">>"))
    except:
```

```
print("please enter a valid no")  
continue
```

```
if playerOneTurn :  
    choices[choice - 1] = 'X'  
else :  
choices[choice - 1] = 'O'
```


References

- [1] <https://ddocs.python.org>.