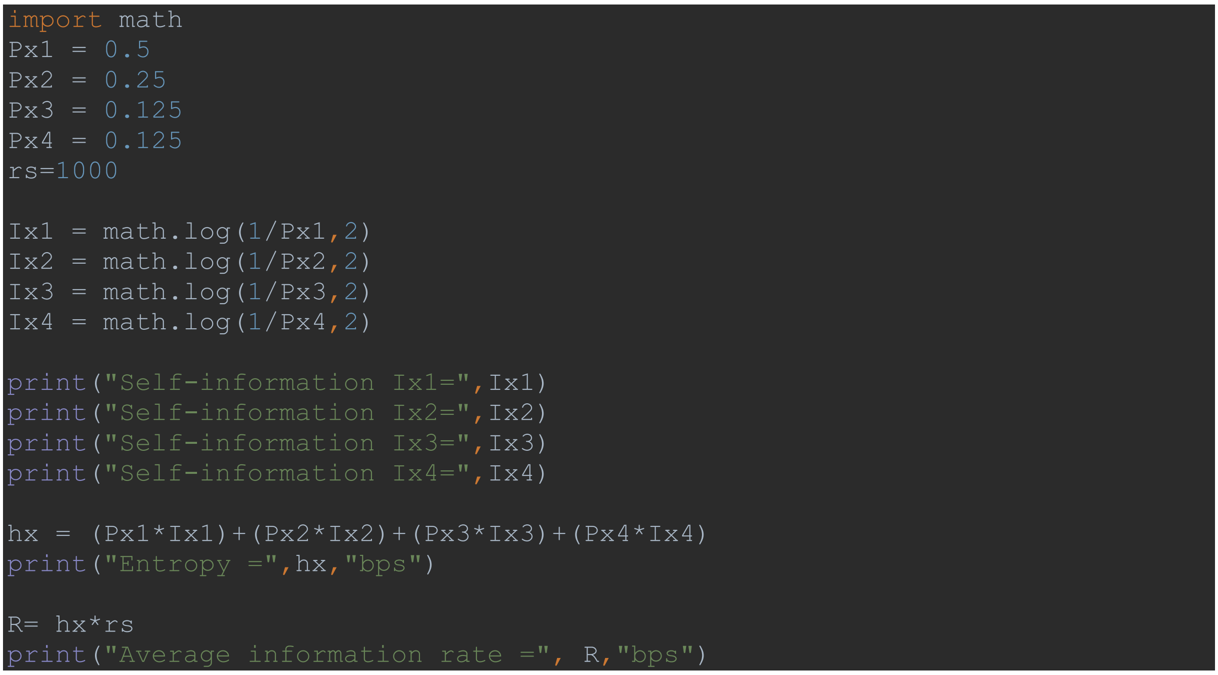
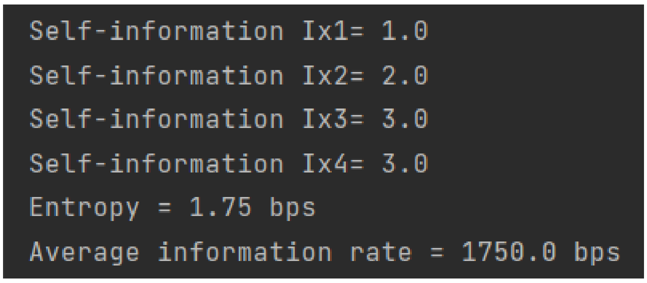
Q1 self information, information rate and source entropy

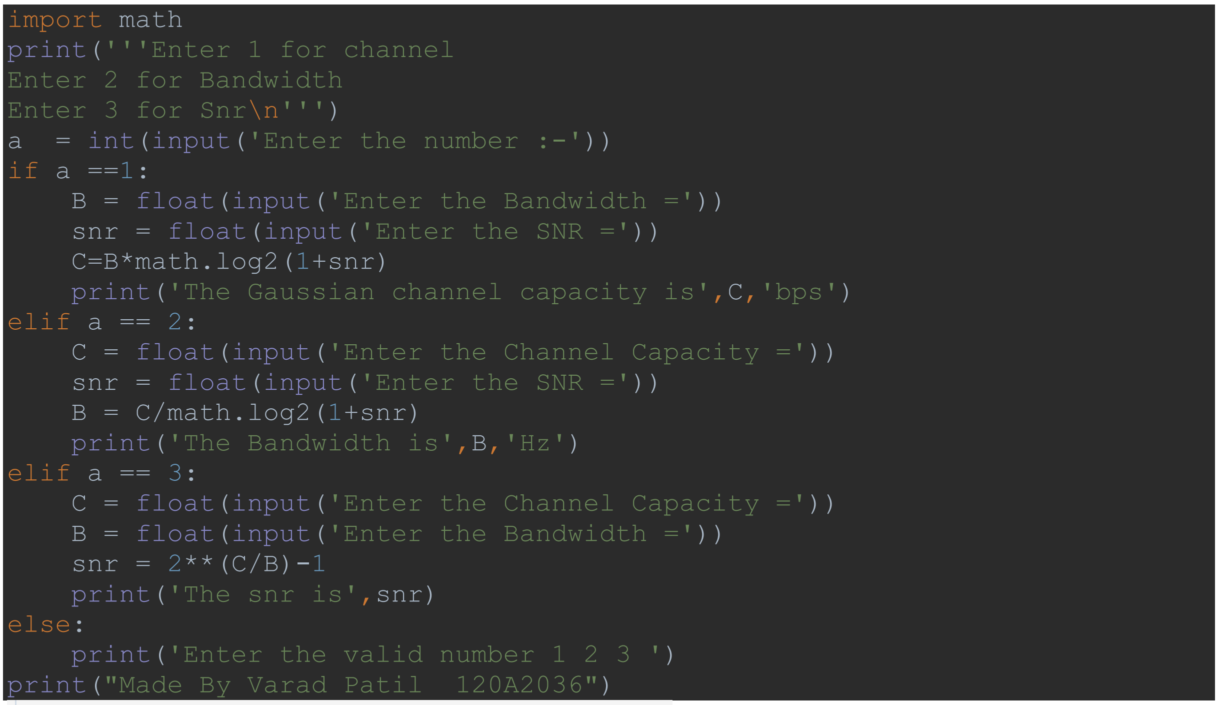


```python
import math
Px1 = 0.5
Px2 = 0.25
Px3 = 0.125
Px4 = 0.125
rs=1000

Ix1 = math.log(1/Px1,2)
Ix2 = math.log(1/Px2,2)
Ix3 = math.log(1/Px3,2)
Ix4 = math.log(1/Px4,2)

print("Self-information Ix1=",Ix1)
print("Self-information Ix2=",Ix2)
print("Self-information Ix3=",Ix3)
print("Self-information Ix4=",Ix4)

hx = (Px1*Ix1)+(Px2*Ix2)+(Px3*Ix3)+(Px4*Ix4)
print("Entropy =",hx,"bps")

R= hx*rs
print("Average information rate =", R,"bps")
```

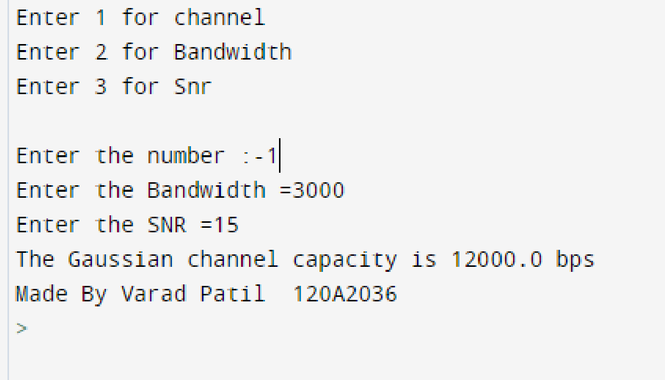


```
Self-information Ix1= 1.0
Self-information Ix2= 2.0
Self-information Ix3= 3.0
Self-information Ix4= 3.0
Entropy = 1.75 bps
Average information rate = 1750.0 bps
```

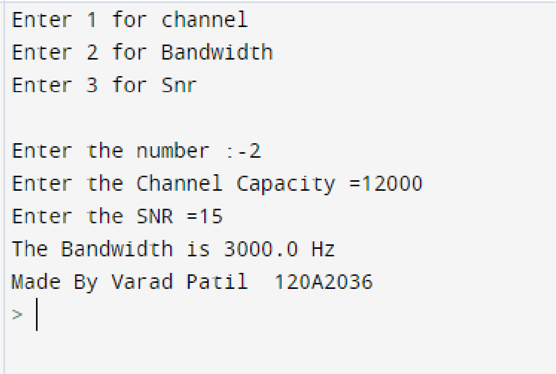
q2 channel capacity, bandwidth and snr



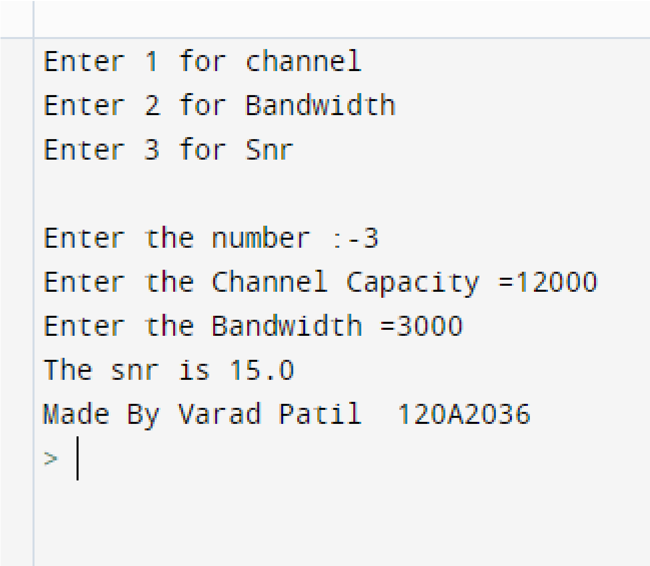
```
import math
print('''Enter 1 for channel
Enter 2 for Bandwidth
Enter 3 for Snr\n''')
a  = int(input('Enter the number :-'))
if a ==1:
    B = float(input('Enter the Bandwidth ='))
    snr = float(input('Enter the SNR ='))
    C=B*math.log2(1+snr)
    print('The Gaussian channel capacity is',C,'bps')
elif a == 2:
    C = float(input('Enter the Channel Capacity ='))
    snr = float(input('Enter the SNR ='))
    B = C/math.log2(1+snr)
    print('The Bandwidth is',B,'Hz')
elif a == 3:
    C = float(input('Enter the Channel Capacity ='))
    B = float(input('Enter the Bandwidth ='))
    snr = 2**(C/B)-1
    print('The snr is',snr)
else:
    print('Enter the valid number 1 2 3 ')
print("Made By Varad Patil  120A2036")
```



```
Enter 1 for channel
Enter 2 for Bandwidth
Enter 3 for Snr

Enter the number :-1
Enter the Bandwidth =3000
Enter the SNR =15
The Gaussian channel capacity is 12000.0 bps
Made By Varad Patil  120A2036
>
```

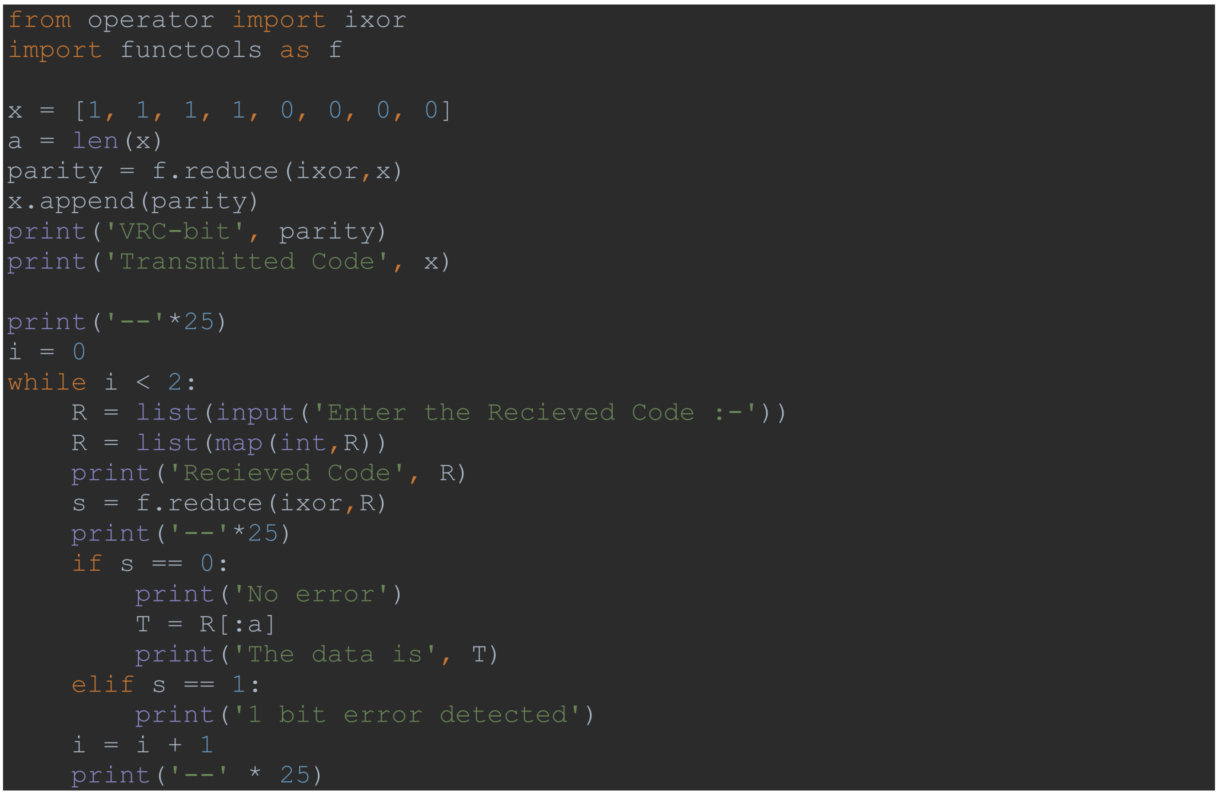


```
Enter 1 for channel
Enter 2 for Bandwidth
Enter 3 for Snr

Enter the number :-2
Enter the Channel Capacity =12000
Enter the SNR =15
The Bandwidth is 3000.0 Hz
Made By Varad Patil  120A2036
>
```

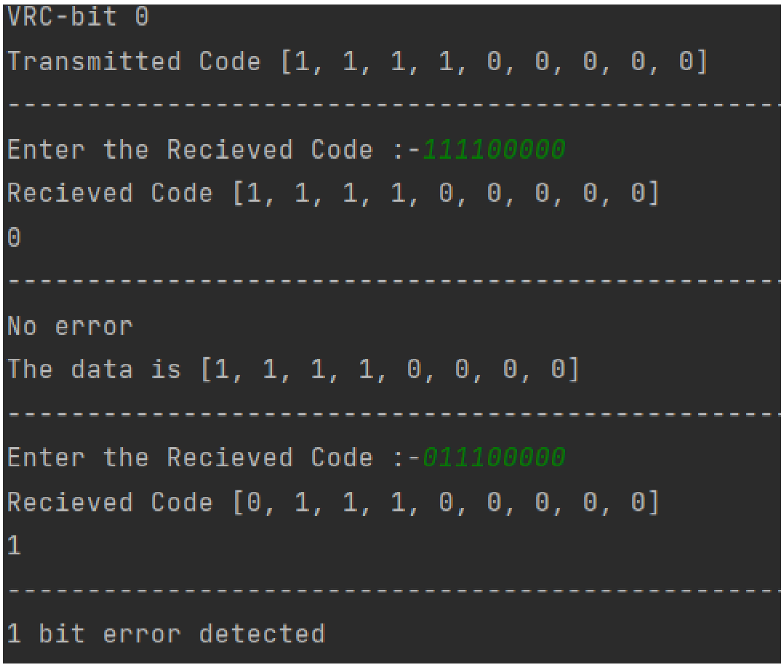


```
Enter 1 for channel
Enter 2 for Bandwidth
Enter 3 for Snr

Enter the number :-3
Enter the Channel Capacity =12000
Enter the Bandwidth =3000
The snr is 15.0
Made By Varad Patil  120A2036
>
```

Q3 write a program for VRC generation and error detection

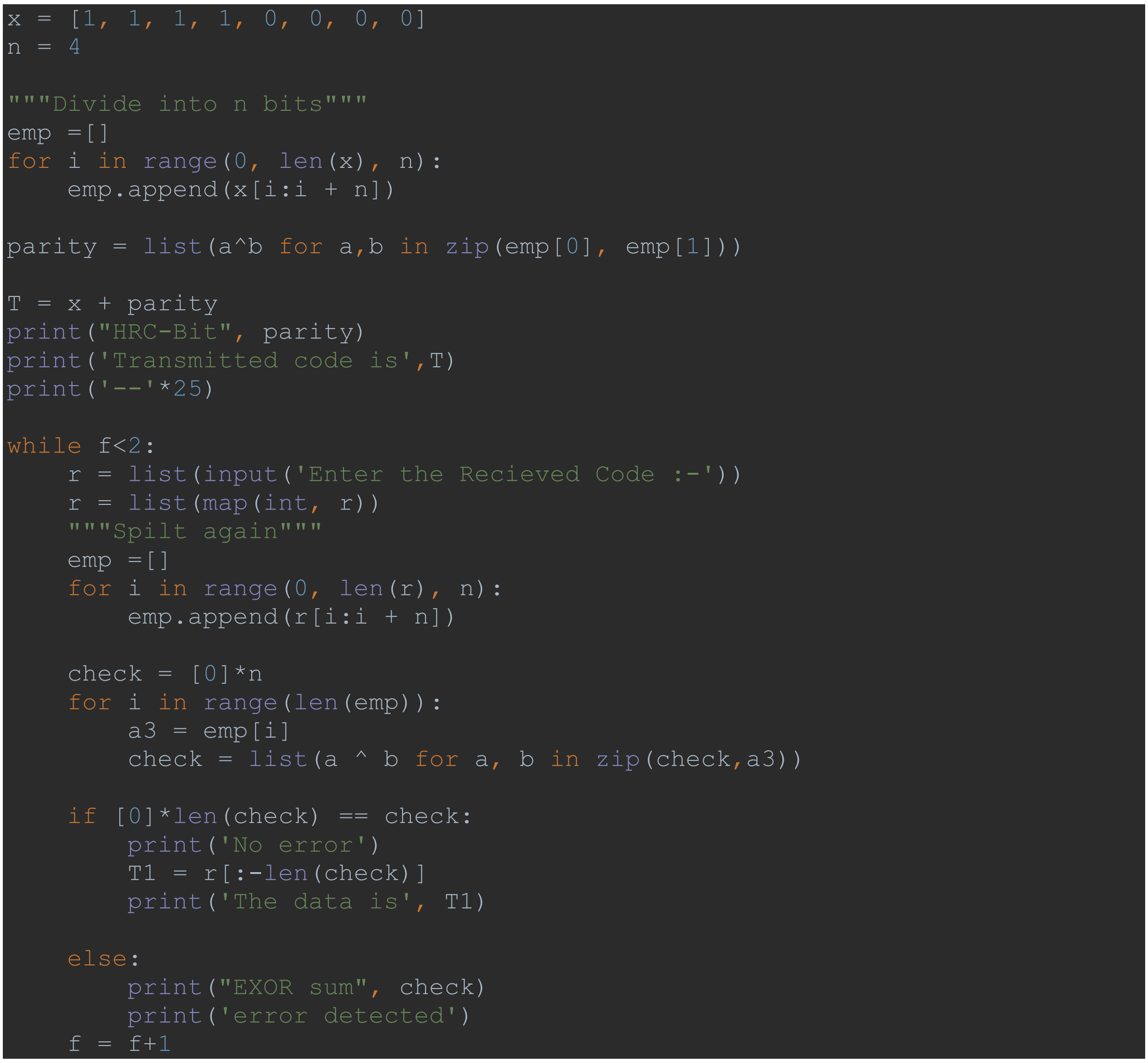


```
from operator import ixor
import functools as f

x = [1, 1, 1, 1, 0, 0, 0, 0]
a = len(x)
parity = f.reduce(ixor,x)
x.append(parity)
print('VRC-bit', parity)
print('Transmitted Code', x)

print('--'*25)
i = 0
while i < 2:
    R = list(input('Enter the Recieved Code :-'))
    R = list(map(int,R))
    print('Recieved Code', R)
    s = f.reduce(ixor,R)
    print('--'*25)
    if s == 0:
        print('No error')
        T = R[:a]
        print('The data is', T)
    elif s == 1:
        print('1 bit error detected')
    i = i + 1
    print('--' * 25)
```

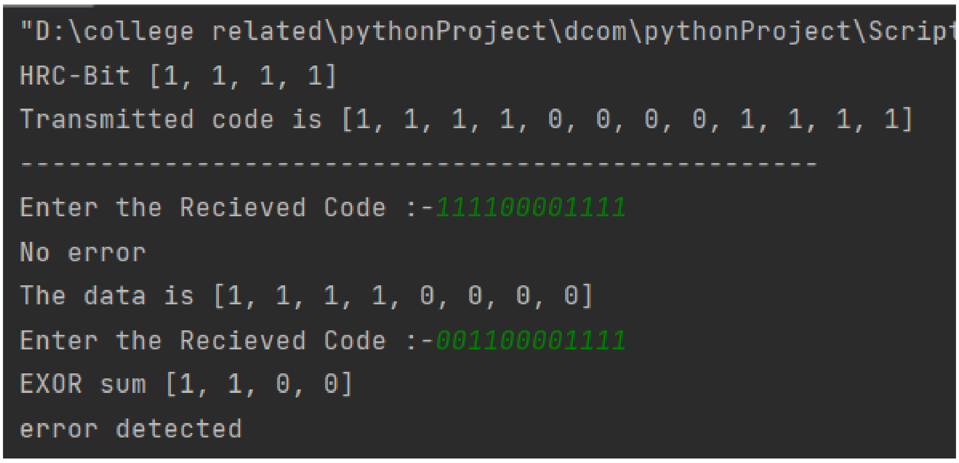


```
VRC-bit 0
Transmitted Code [1, 1, 1, 1, 0, 0, 0, 0, 0]
-------------------------------------------------
Enter the Recieved Code :-111100000
Recieved Code [1, 1, 1, 1, 0, 0, 0, 0, 0]
0
-------------------------------------------------
No error
The data is [1, 1, 1, 1, 0, 0, 0, 0]
-------------------------------------------------
Enter the Recieved Code :-011100000
Recieved Code [0, 1, 1, 1, 0, 0, 0, 0, 0]
1
-------------------------------------------------
1 bit error detected
```

Q4 Write a program for HRC code generation and error detection.

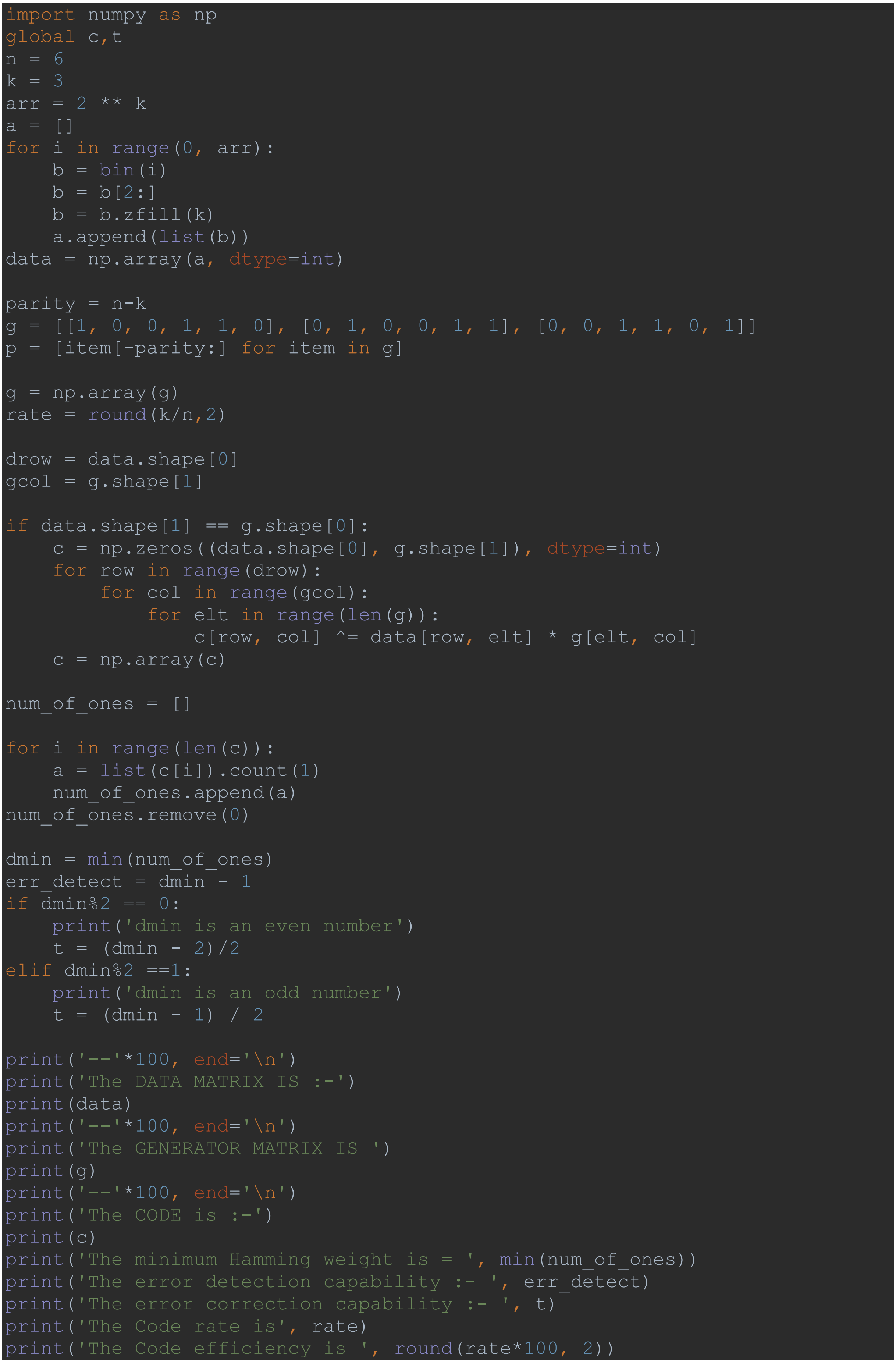


```python
x = [1, 1, 1, 1, 0, 0, 0, 0]
n = 4

"""Divide into n bits"""
emp =[]
for i in range(0, len(x), n):
    emp.append(x[i:i + n])

parity = list(a^b for a,b in zip(emp[0], emp[1]))

T = x + parity
print("HRC-Bit", parity)
print('Transmitted code is',T)
print('--'*25)

while f<2:
    r = list(input('Enter the Recieved Code :-'))
    r = list(map(int, r))
    """Spilt again"""
    emp =[]
    for i in range(0, len(r), n):
        emp.append(r[i:i + n])

    check = [0]*n
    for i in range(len(emp)):
        a3 = emp[i]
        check = list(a ^ b for a, b in zip(check,a3))

    if [0]*len(check) == check:
        print('No error')
        T1 = r[:-len(check)]
        print('The data is', T1)

    else:
        print("EXOR sum", check)
        print('error detected')
    f = f+1
```

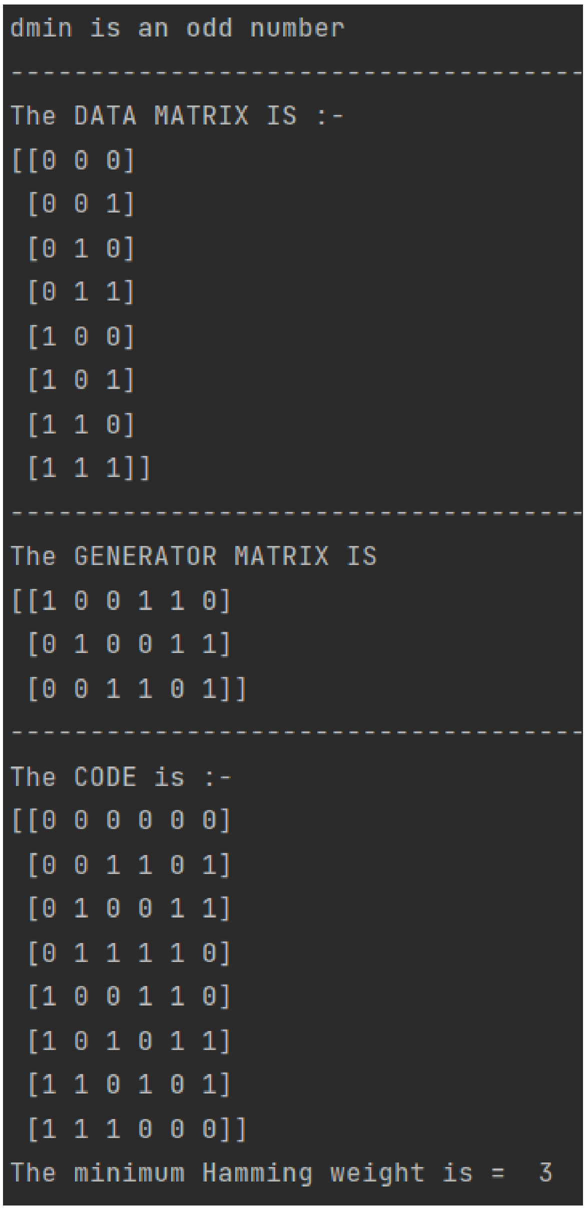


```
"D:\college related\pythonProject\dcom\pythonProject\Script
HRC-Bit [1, 1, 1, 1]
Transmitted code is [1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1]
--------------------------------------------------
Enter the Recieved Code :-111100001111
No error
The data is [1, 1, 1, 1, 0, 0, 0, 0]
Enter the Recieved Code :-001100001111
EXOR sum [1, 1, 0, 0]
error detected
```

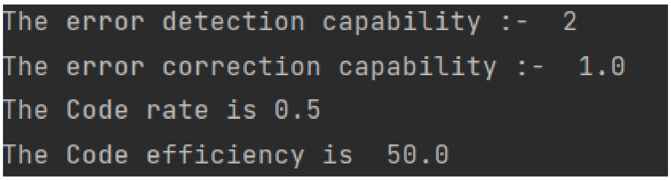
Q5 write a program for (6,3) codewords generation.



```
import numpy as np
global c,t
n = 6
k = 3
arr = 2 ** k
a = []
for i in range(0, arr):
    b = bin(i)
    b = b[2:]
    b = b.zfill(k)
    a.append(list(b))
data = np.array(a, dtype=int)

parity = n-k
g = [[1, 0, 0, 1, 1, 0], [0, 1, 0, 0, 1, 1], [0, 0, 1, 1, 0, 1]]
p = [item[-parity:] for item in g]

g = np.array(g)
rate = round(k/n,2)

drow = data.shape[0]
gcol = g.shape[1]

if data.shape[1] == g.shape[0]:
    c = np.zeros((data.shape[0], g.shape[1]), dtype=int)
    for row in range(drow):
        for col in range(gcol):
            for elt in range(len(g)):
                c[row, col] ^= data[row, elt] * g[elt, col]
    c = np.array(c)

num_of_ones = []

for i in range(len(c)):
    a = list(c[i]).count(1)
    num_of_ones.append(a)
num_of_ones.remove(0)

dmin = min(num_of_ones)
err_detect = dmin - 1
if dmin%2 == 0:
    print('dmin is an even number')
    t = (dmin - 2)/2
elif dmin%2 ==1:
    print('dmin is an odd number')
    t = (dmin - 1) / 2

print('--'*100, end='\n')
print('The DATA MATRIX IS :-')
print(data)
print('--'*100, end='\n')
print('The GENERATOR MATRIX IS ')
print(g)
print('--'*100, end='\n')
print('The CODE is :-')
print(c)
print('The minimum Hamming weight is = ', min(num_of_ones))
print('The error detection capability :- ', err_detect)
print('The error correction capability :- ', t)
print('The Code rate is', rate)
print('The Code efficiency is ', round(rate*100, 2))
```



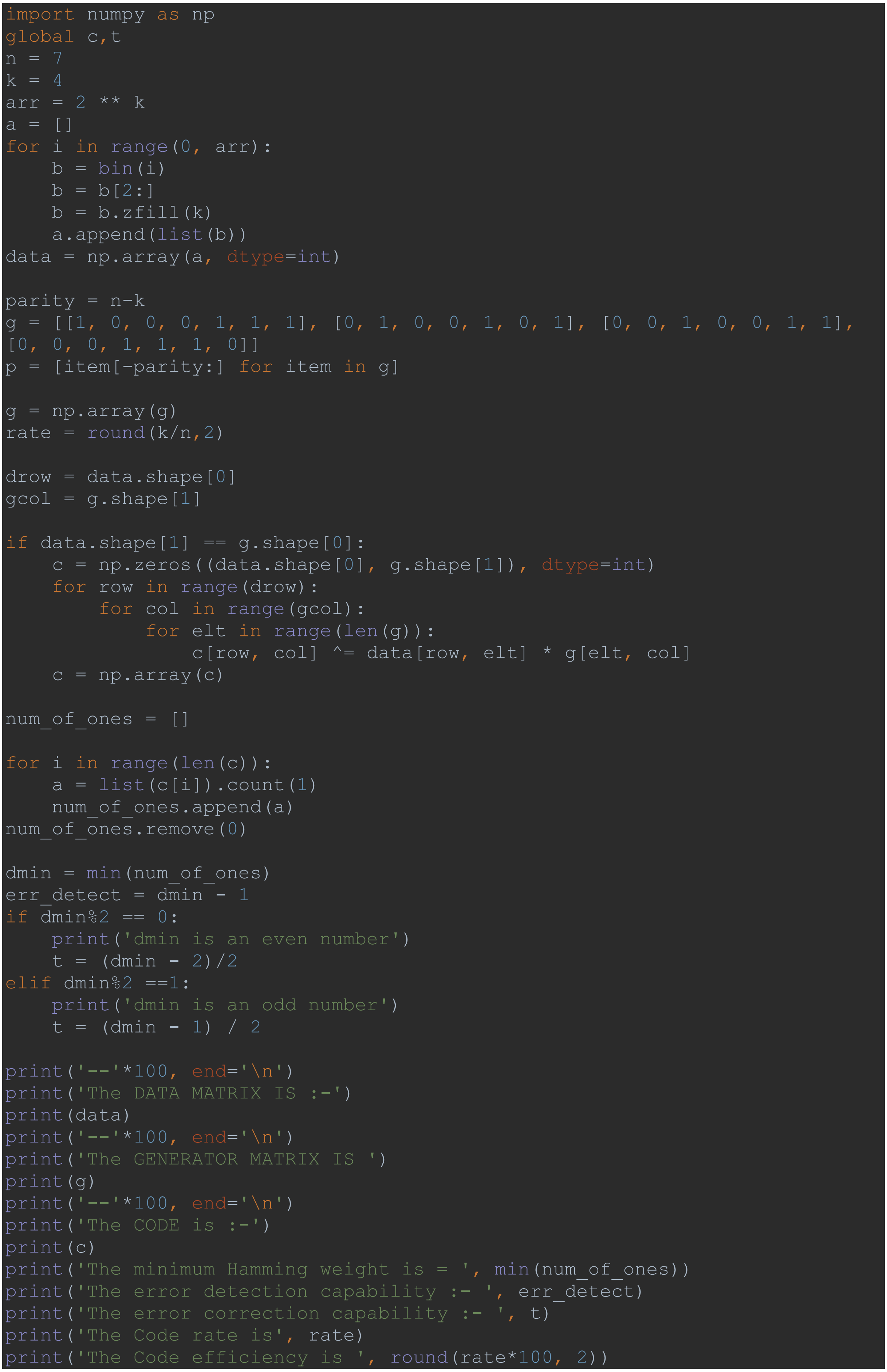
```
dmin is an odd number
------------------------------------
The DATA MATRIX IS :-
[[0 0 0]
 [0 0 1]
 [0 1 0]
 [0 1 1]
 [1 0 0]
 [1 0 1]
 [1 1 0]
 [1 1 1]]
------------------------------------
The GENERATOR MATRIX IS
[[1 0 0 1 1 0]
 [0 1 0 0 1 1]
 [0 0 1 1 0 1]]
------------------------------------
The CODE is :-
[[0 0 0 0 0 0]
 [0 0 1 1 0 1]
 [0 1 0 0 1 1]
 [0 1 1 1 1 0]
 [1 0 0 1 1 0]
 [1 0 1 0 1 1]
 [1 1 0 1 0 1]
 [1 1 1 0 0 0]]
The minimum Hamming weight is =  3
```



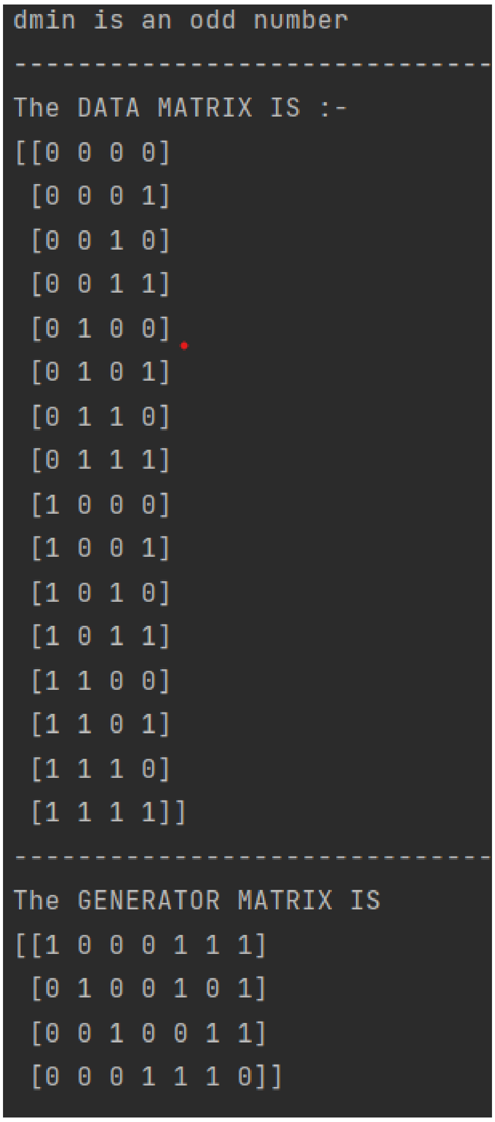
```
The error detection capability :-  2
The error correction capability :-  1.0
The Code rate is 0.5
The Code efficiency is  50.0
```

Q6 write a program for (7,4) codewords generation.

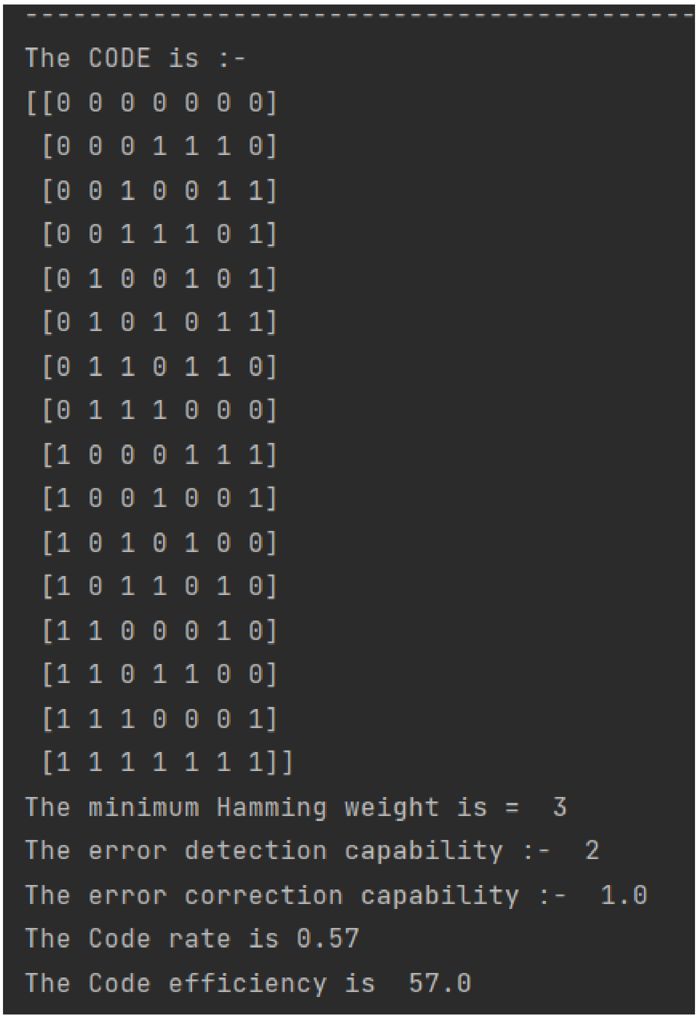
Code same hai as Q5 bas g ka value aur k,n ka value change kiya hai



```
import numpy as np
global c,t
n = 7
k = 4
arr = 2 ** k
a = []
for i in range(0, arr):
    b = bin(i)
    b = b[2:]
    b = b.zfill(k)
    a.append(list(b))
data = np.array(a, dtype=int)

parity = n-k
g = [[1, 0, 0, 0, 1, 1, 1], [0, 1, 0, 0, 1, 0, 1], [0, 0, 1, 0, 0, 1, 1],
[0, 0, 0, 1, 1, 1, 0]]
p = [item[-parity:] for item in g]

g = np.array(g)
rate = round(k/n,2)

drow = data.shape[0]
gcol = g.shape[1]

if data.shape[1] == g.shape[0]:
    c = np.zeros((data.shape[0], g.shape[1]), dtype=int)
    for row in range(drow):
        for col in range(gcol):
            for elt in range(len(g)):
                c[row, col] ^= data[row, elt] * g[elt, col]
    c = np.array(c)

num_of_ones = []

for i in range(len(c)):
    a = list(c[i]).count(1)
    num_of_ones.append(a)
num_of_ones.remove(0)

dmin = min(num_of_ones)
err_detect = dmin - 1
if dmin%2 == 0:
    print('dmin is an even number')
    t = (dmin - 2)/2
elif dmin%2 ==1:
    print('dmin is an odd number')
    t = (dmin - 1) / 2

print('--'*100, end='\n')
print('The DATA MATRIX IS :-')
print(data)
print('--'*100, end='\n')
print('The GENERATOR MATRIX IS ')
print(g)
print('--'*100, end='\n')
print('The CODE is :-')
print(c)
print('The minimum Hamming weight is = ', min(num_of_ones))
print('The error detection capability :- ', err_detect)
print('The error correction capability :- ', t)
print('The Code rate is', rate)
print('The Code efficiency is ', round(rate*100, 2))
```

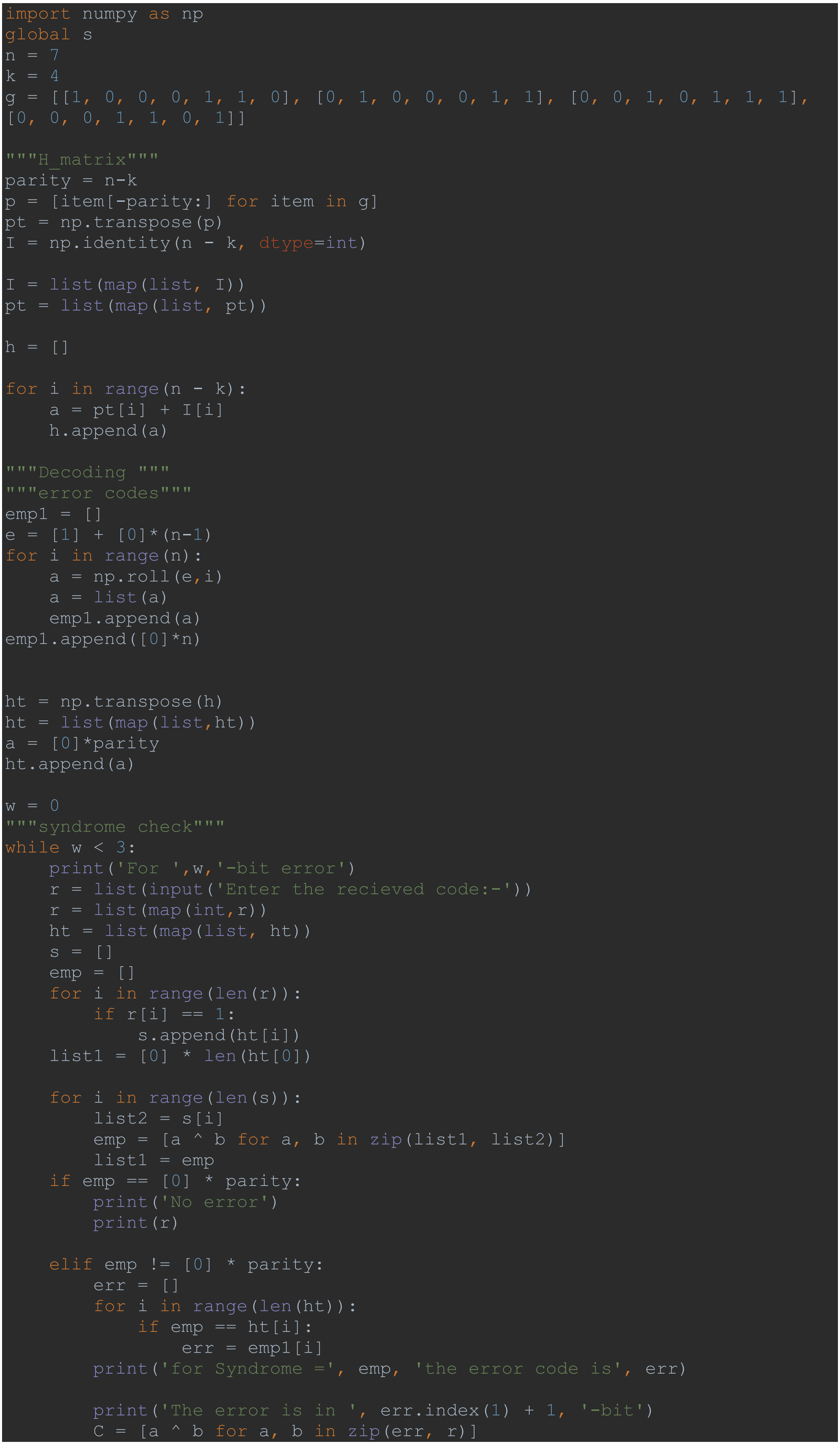


```
dmin is an odd number
------------------------------
The DATA MATRIX IS :-
[[0 0 0 0]
 [0 0 0 1]
 [0 0 1 0]
 [0 0 1 1]
 [0 1 0 0]
 [0 1 0 1]
 [0 1 1 0]
 [0 1 1 1]
 [1 0 0 0]
 [1 0 0 1]
 [1 0 1 0]
 [1 0 1 1]
 [1 1 0 0]
 [1 1 0 1]
 [1 1 1 0]
 [1 1 1 1]]
------------------------------
The GENERATOR MATRIX IS
[[1 0 0 0 1 1 1]
 [0 1 0 0 1 0 1]
 [0 0 1 0 0 1 1]
 [0 0 0 1 1 1 0]]
```

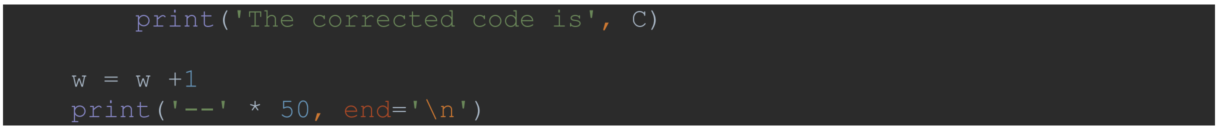


```
------------------------------------------
The CODE is :-
[[0 0 0 0 0 0 0]
 [0 0 0 1 1 1 0]
 [0 0 1 0 0 1 1]
 [0 0 1 1 1 0 1]
 [0 1 0 0 1 0 1]
 [0 1 0 1 0 1 1]
 [0 1 1 0 1 1 0]
 [0 1 1 1 0 0 0]
 [1 0 0 0 1 1 1]
 [1 0 0 1 0 0 1]
 [1 0 1 0 1 0 0]
 [1 0 1 1 0 1 0]
 [1 1 0 0 0 1 0]
 [1 1 0 1 1 0 0]
 [1 1 1 0 0 0 1]
 [1 1 1 1 1 1 1]]
The minimum Hamming weight is =  3
The error detection capability :-  2
The error correction capability :-  1.0
The Code rate is 0.57
The Code efficiency is  57.0
```

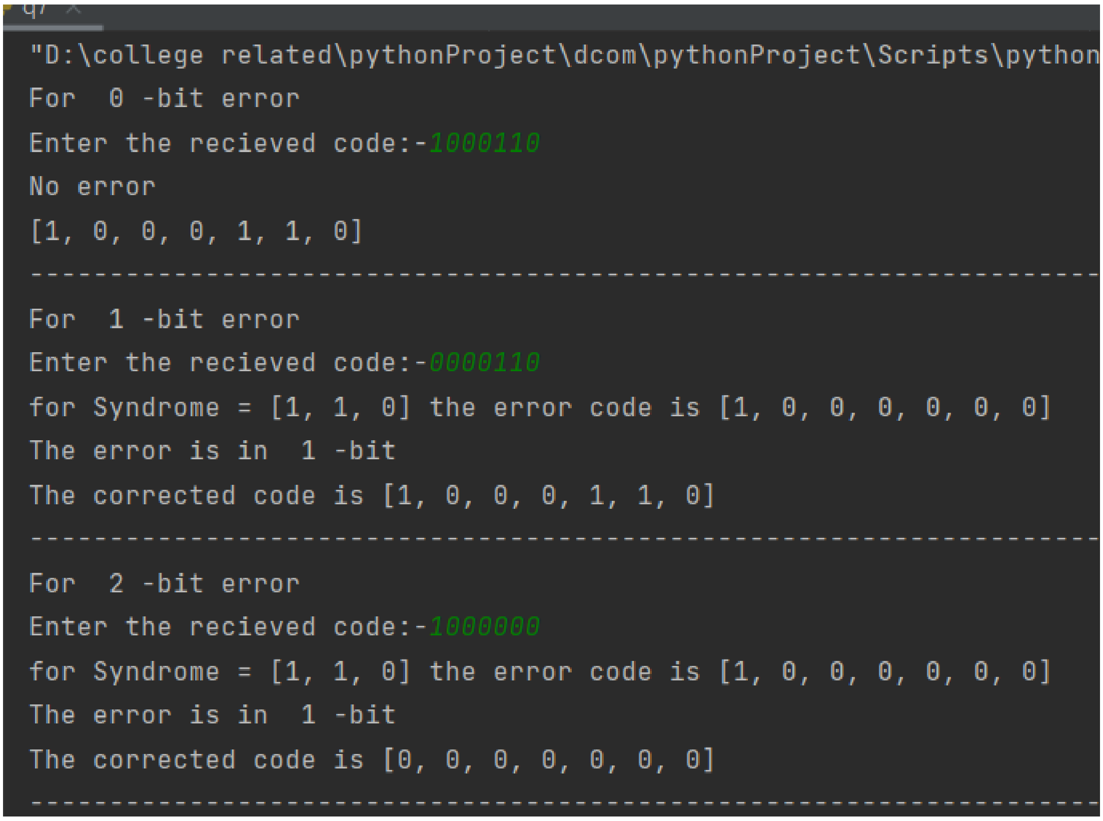
Q6 write a program for (7,4) Hamming code, error detection and correction.



```
import numpy as np
global s
n = 7
k = 4
g = [[1, 0, 0, 0, 1, 1, 0], [0, 1, 0, 0, 0, 1, 1], [0, 0, 1, 0, 1, 1, 1],
[0, 0, 0, 1, 1, 0, 1]]

"""H_matrix"""
parity = n-k
p = [item[-parity:] for item in g]
pt = np.transpose(p)
I = np.identity(n - k, dtype=int)

I = list(map(list, I))
pt = list(map(list, pt))

h = []

for i in range(n - k):
    a = pt[i] + I[i]
    h.append(a)

"""Decoding """
"""error codes"""
emp1 = []
e = [1] + [0]*(n-1)
for i in range(n):
    a = np.roll(e,i)
    a = list(a)
    emp1.append(a)
emp1.append([0]*n)


ht = np.transpose(h)
ht = list(map(list,ht))
a = [0]*parity
ht.append(a)

w = 0
"""syndrome check"""
while w < 3:
    print('For ',w,'-bit error')
    r = list(input('Enter the recieved code:-'))
    r = list(map(int,r))
    ht = list(map(list, ht))
    s = []
    emp = []
    for i in range(len(r)):
        if r[i] == 1:
            s.append(ht[i])
    list1 = [0] * len(ht[0])

    for i in range(len(s)):
        list2 = s[i]
        emp = [a ^ b for a, b in zip(list1, list2)]
        list1 = emp
    if emp == [0] * parity:
        print('No error')
        print(r)

    elif emp != [0] * parity:
        err = []
        for i in range(len(ht)):
            if emp == ht[i]:
                err = emp1[i]
        print('for Syndrome =', emp, 'the error code is', err)

        print('The error is in ', err.index(1) + 1, '-bit')
        C = [a ^ b for a, b in zip(err, r)]
```

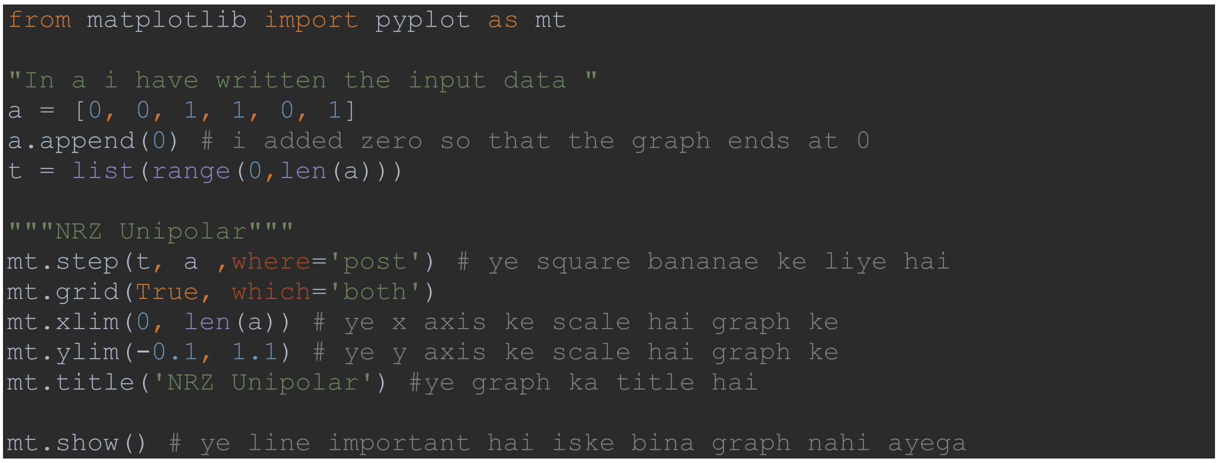


```
        print('The corrected code is', C)

    w = w +1
    print('--' * 50, end='\n')
```

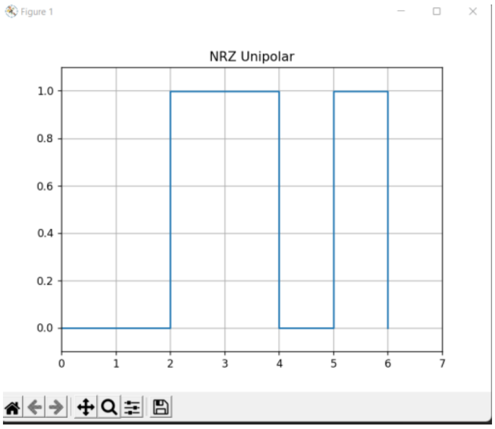


```
"D:\college related\pythonProject\dcom\pythonProject\Scripts\python
For  0 -bit error
Enter the recieved code:-1000110
No error
[1, 0, 0, 0, 1, 1, 0]
--------------------------------------------------------------------
For  1 -bit error
Enter the recieved code:-0000110
for Syndrome = [1, 1, 0] the error code is [1, 0, 0, 0, 0, 0, 0]
The error is in  1 -bit
The corrected code is [1, 0, 0, 0, 1, 1, 0]
--------------------------------------------------------------------
For  2 -bit error
Enter the recieved code:-1000000
for Syndrome = [1, 1, 0] the error code is [1, 0, 0, 0, 0, 0, 0]
The error is in  1 -bit
The corrected code is [0, 0, 0, 0, 0, 0, 0]
--------------------------------------------------------------------
```

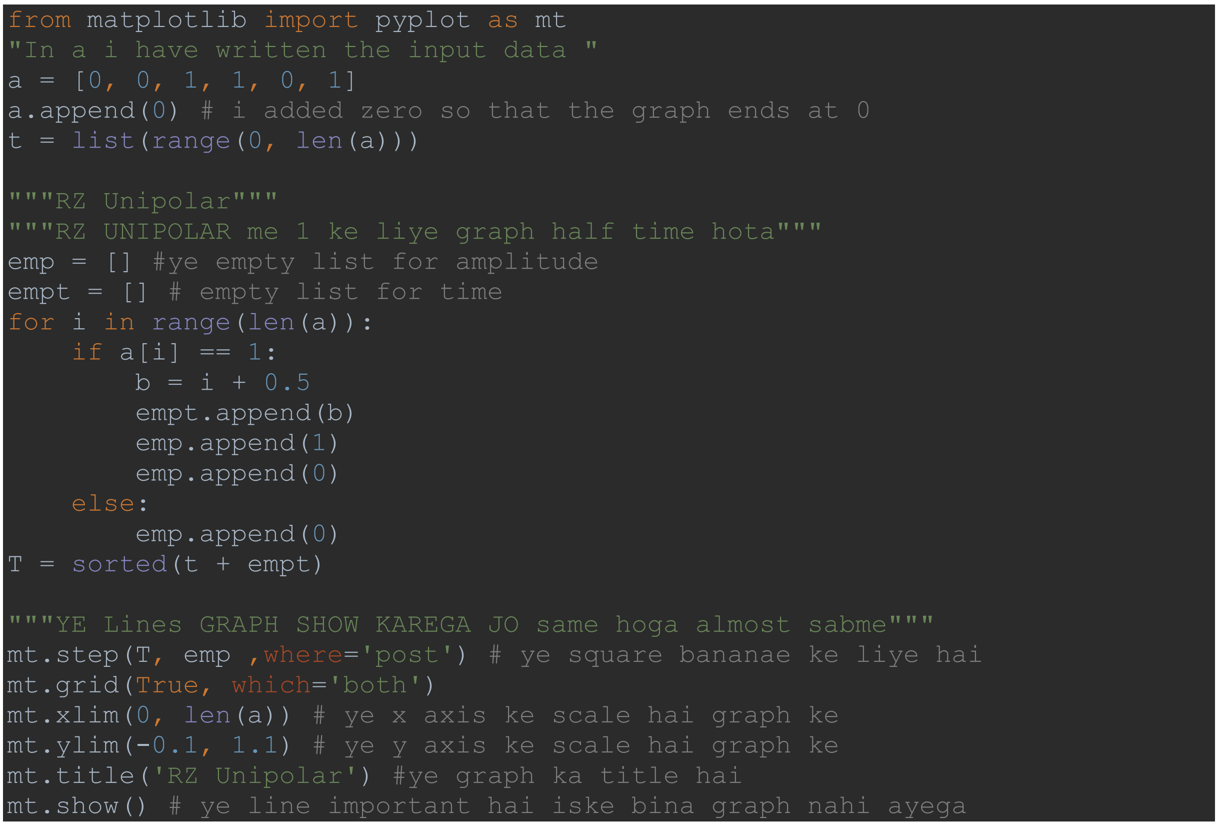
Q 8 Write a program for generating nrz unipolar code for the data 001101



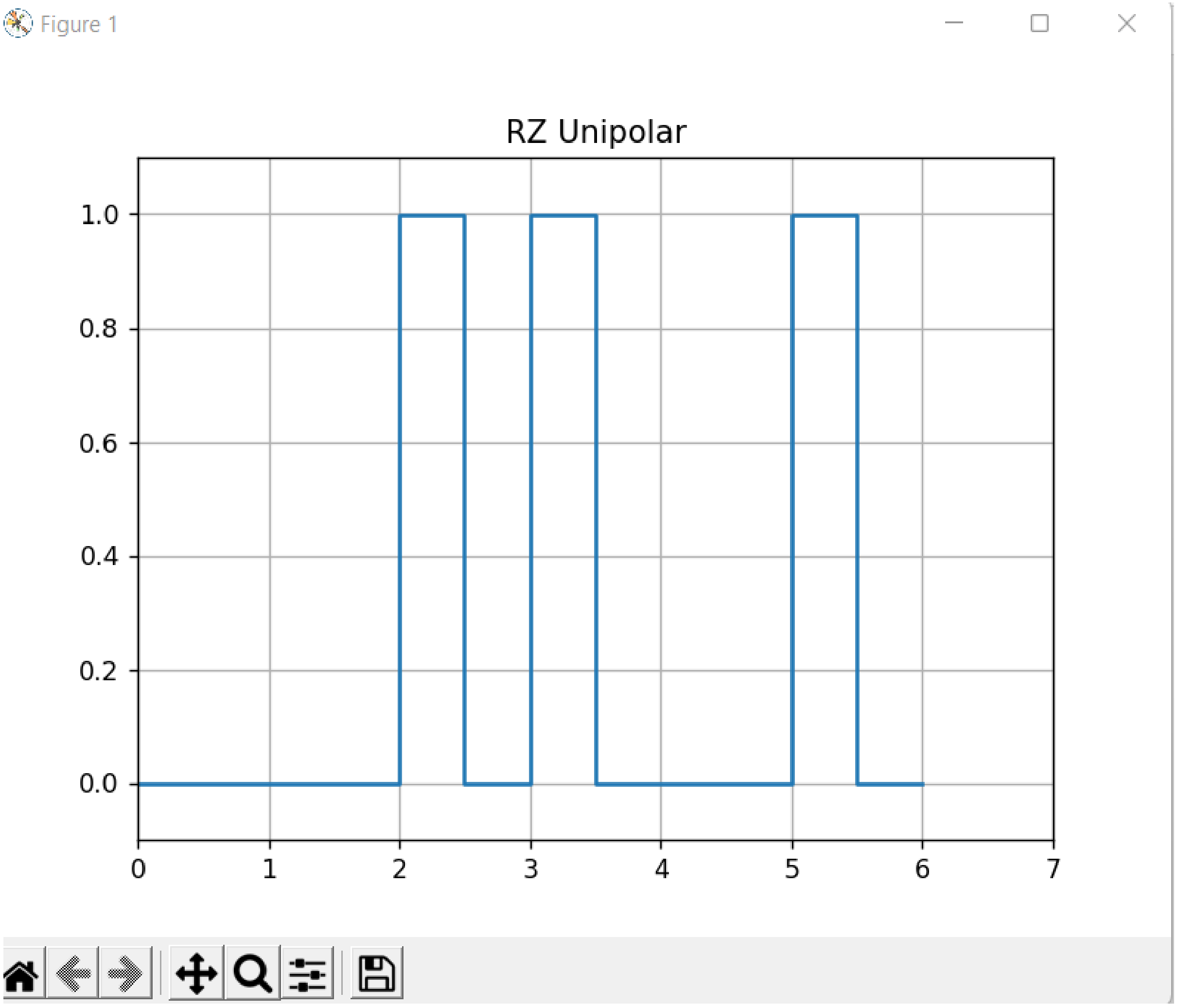
```
from matplotlib import pyplot as mt

"In a i have written the input data "
a = [0, 0, 1, 1, 0, 1]
a.append(0) # i added zero so that the graph ends at 0
t = list(range(0,len(a)))

"""NRZ Unipolar"""
mt.step(t, a ,where='post') # ye square bananae ke liye hai
mt.grid(True, which='both')
mt.xlim(0, len(a)) # ye x axis ke scale hai graph ke
mt.ylim(-0.1, 1.1) # ye y axis ke scale hai graph ke
mt.title('NRZ Unipolar') #ye graph ka title hai

mt.show() # ye line important hai iske bina graph nahi ayega
```



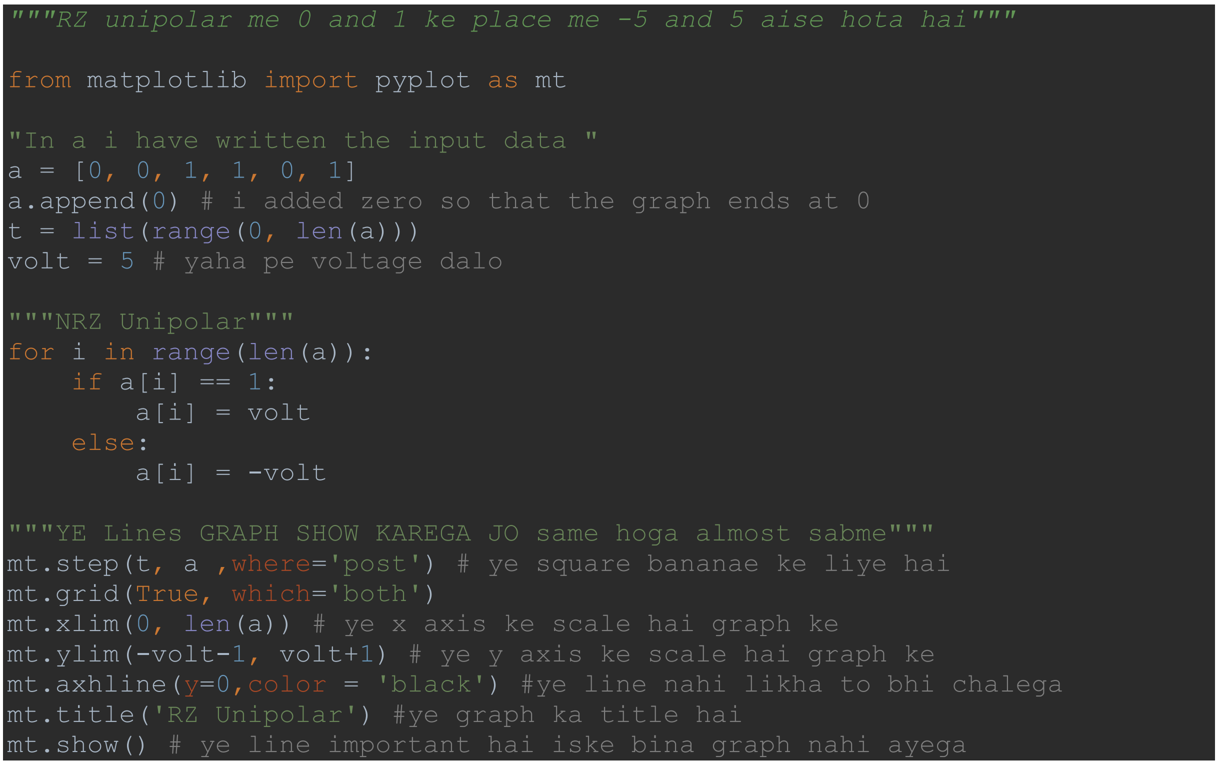
Q 9 Write a program for generating rz unipolar code for the data 001101



```
from matplotlib import pyplot as mt
"In a i have written the input data "
a = [0, 0, 1, 1, 0, 1]
a.append(0) # i added zero so that the graph ends at 0
t = list(range(0, len(a)))

"""RZ Unipolar"""
"""RZ UNIPOLAR me 1 ke liye graph half time hota"""
emp = [] #ye empty list for amplitude
empt = [] # empty list for time
for i in range(len(a)):
    if a[i] == 1:
        b = i + 0.5
        empt.append(b)
        emp.append(1)
        emp.append(0)
    else:
        emp.append(0)
T = sorted(t + empt)

"""YE Lines GRAPH SHOW KAREGA JO same hoga almost sabme"""
mt.step(T, emp ,where='post') # ye square bananae ke liye hai
mt.grid(True, which='both')
mt.xlim(0, len(a)) # ye x axis ke scale hai graph ke
mt.ylim(-0.1, 1.1) # ye y axis ke scale hai graph ke
mt.title('RZ Unipolar') #ye graph ka title hai
mt.show() # ye line important hai iske bina graph nahi ayega
```



Q 10 Write a program for generating nrz polar code for the data 001101



```
"""RZ unipolar me 0 and 1 ke place me -5 and 5 aise hota hai"""

from matplotlib import pyplot as mt

"In a i have written the input data "
a = [0, 0, 1, 1, 0, 1]
a.append(0) # i added zero so that the graph ends at 0
t = list(range(0, len(a)))
volt = 5 # yaha pe voltage dalo

"""NRZ Unipolar"""
for i in range(len(a)):
    if a[i] == 1:
        a[i] = volt
    else:
        a[i] = -volt

"""YE Lines GRAPH SHOW KAREGA JO same hoga almost sabme"""
mt.step(t, a ,where='post') # ye square bananae ke liye hai
mt.grid(True, which='both')
mt.xlim(0, len(a)) # ye x axis ke scale hai graph ke
mt.ylim(-volt-1, volt+1) # ye y axis ke scale hai graph ke
mt.axhline(y=0,color = 'black') #ye line nahi likha to bhi chalega
mt.title('RZ Unipolar') #ye graph ka title hai
mt.show() # ye line important hai iske bina graph nahi ayega
```

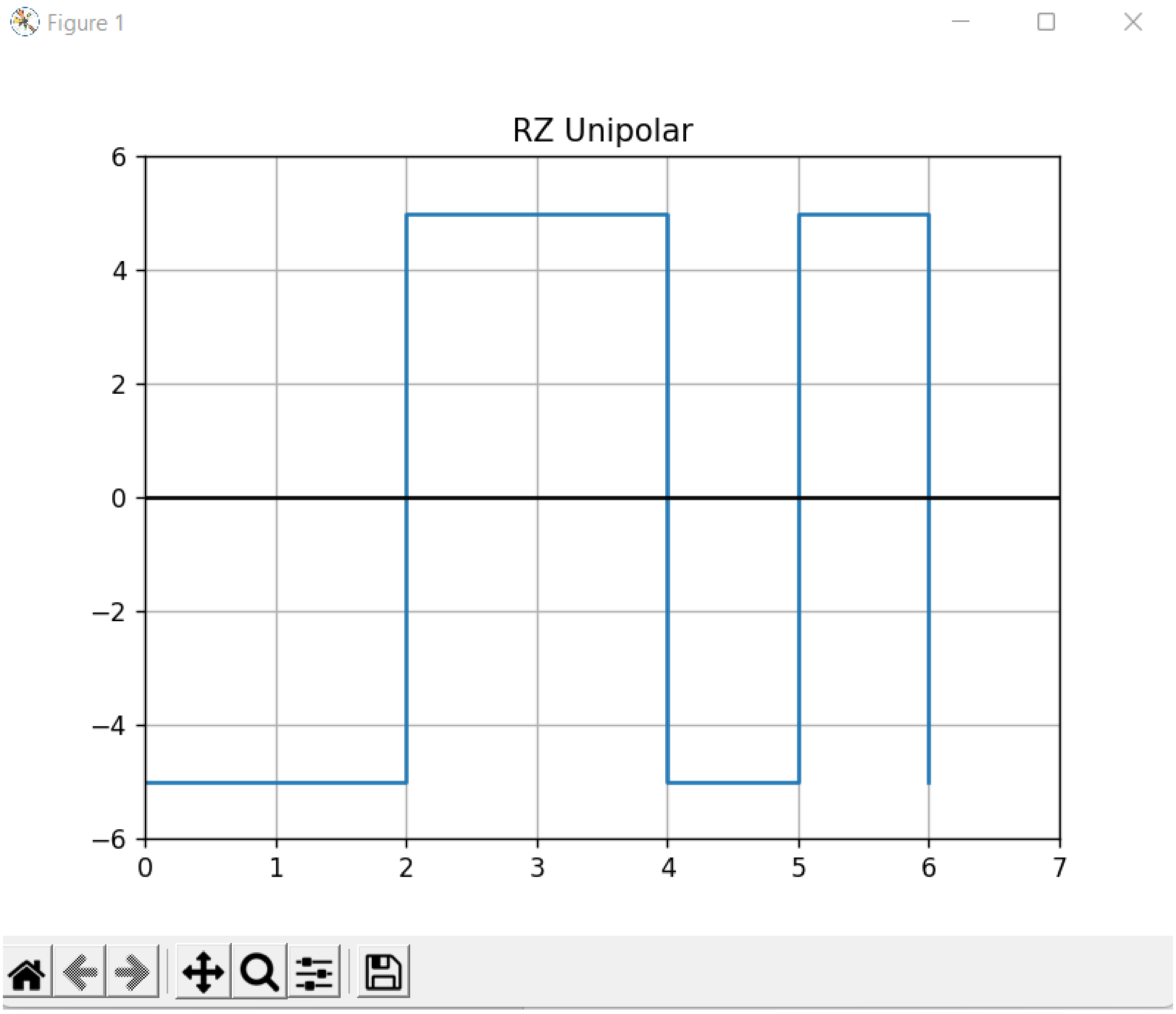
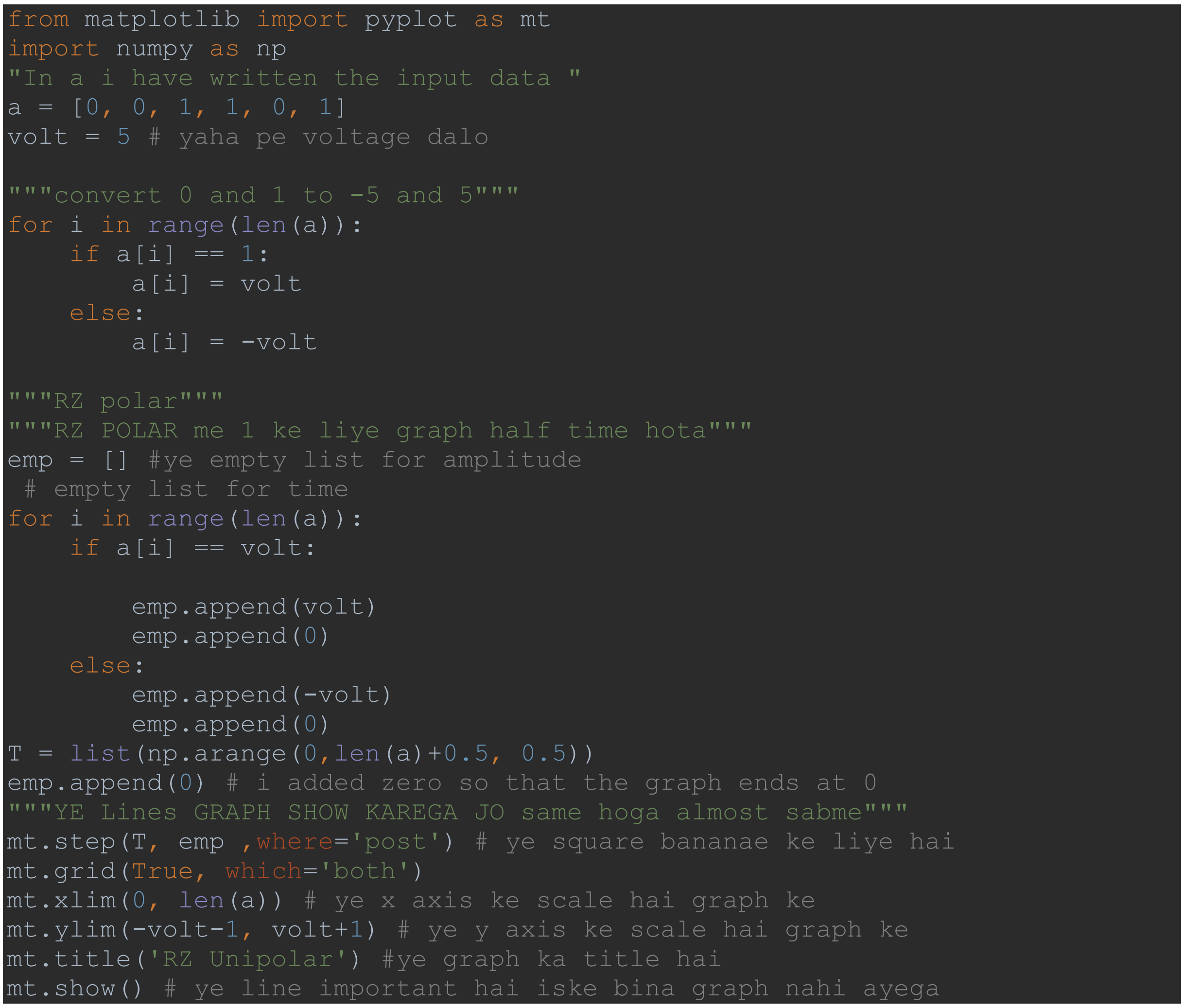


Figure 1
RZ Unipolar
6
4
2
0
−2
−4
−6
0
1
2
3
4
5
6
7

Q 11 Write a program for generating rz polar code for the data 001101



```
from matplotlib import pyplot as mt
import numpy as np
"In a i have written the input data "
a = [0, 0, 1, 1, 0, 1]
volt = 5 # yaha pe voltage dalo

"""convert 0 and 1 to -5 and 5"""
for i in range(len(a)):
    if a[i] == 1:
        a[i] = volt
    else:
        a[i] = -volt

"""RZ polar"""
"""RZ POLAR me 1 ke liye graph half time hota"""
emp = [] #ye empty list for amplitude
 # empty list for time
for i in range(len(a)):
    if a[i] == volt:

        emp.append(volt)
        emp.append(0)
    else:
        emp.append(-volt)
        emp.append(0)
T = list(np.arange(0,len(a)+0.5, 0.5))
emp.append(0) # i added zero so that the graph ends at 0
"""YE Lines GRAPH SHOW KAREGA JO same hoga almost sabme"""
mt.step(T, emp ,where='post') # ye square bananae ke liye hai
mt.grid(True, which='both')
mt.xlim(0, len(a)) # ye x axis ke scale hai graph ke
mt.ylim(-volt-1, volt+1) # ye y axis ke scale hai graph ke
mt.title('RZ Unipolar') #ye graph ka title hai
mt.show() # ye line important hai iske bina graph nahi ayega
```

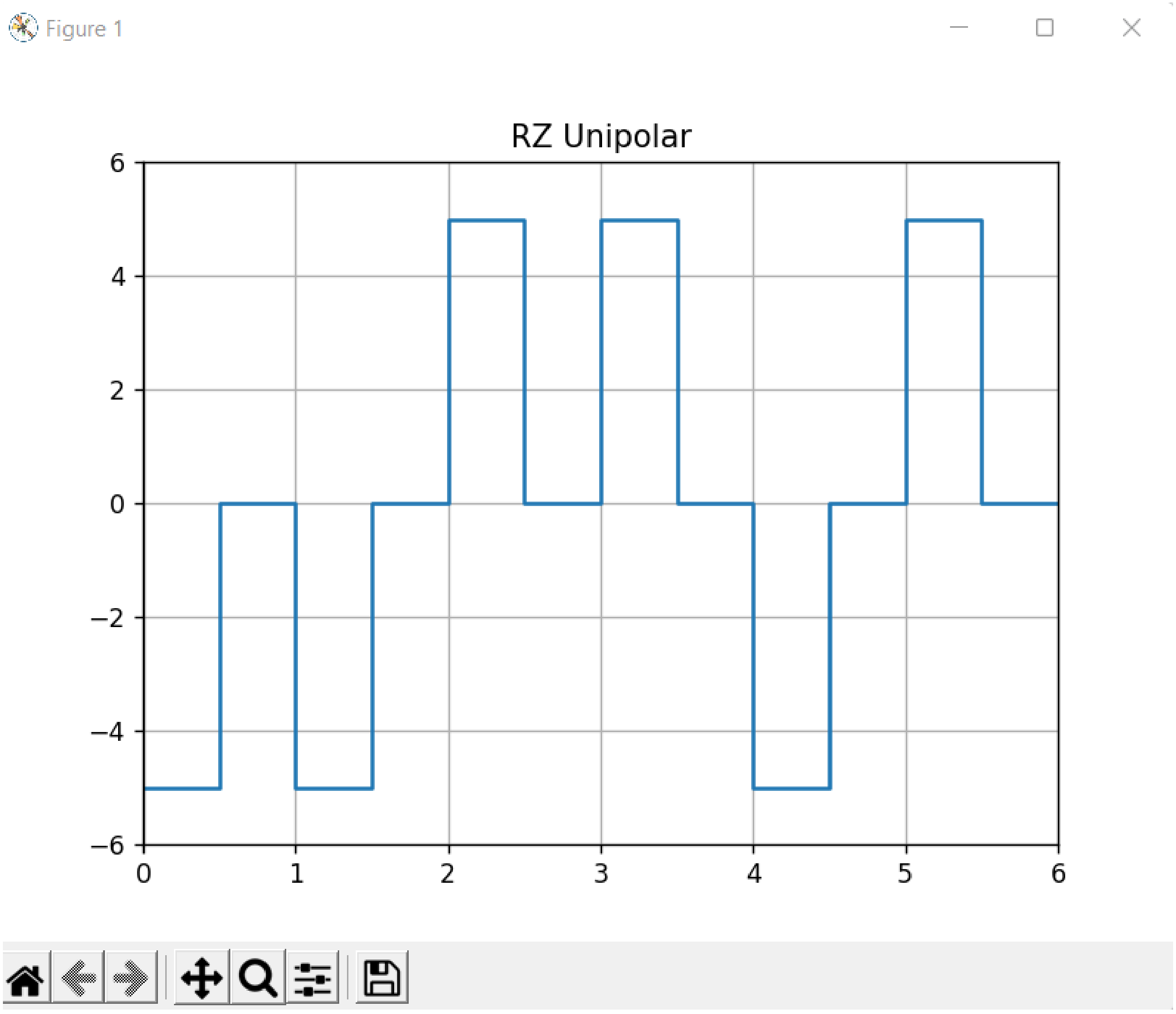
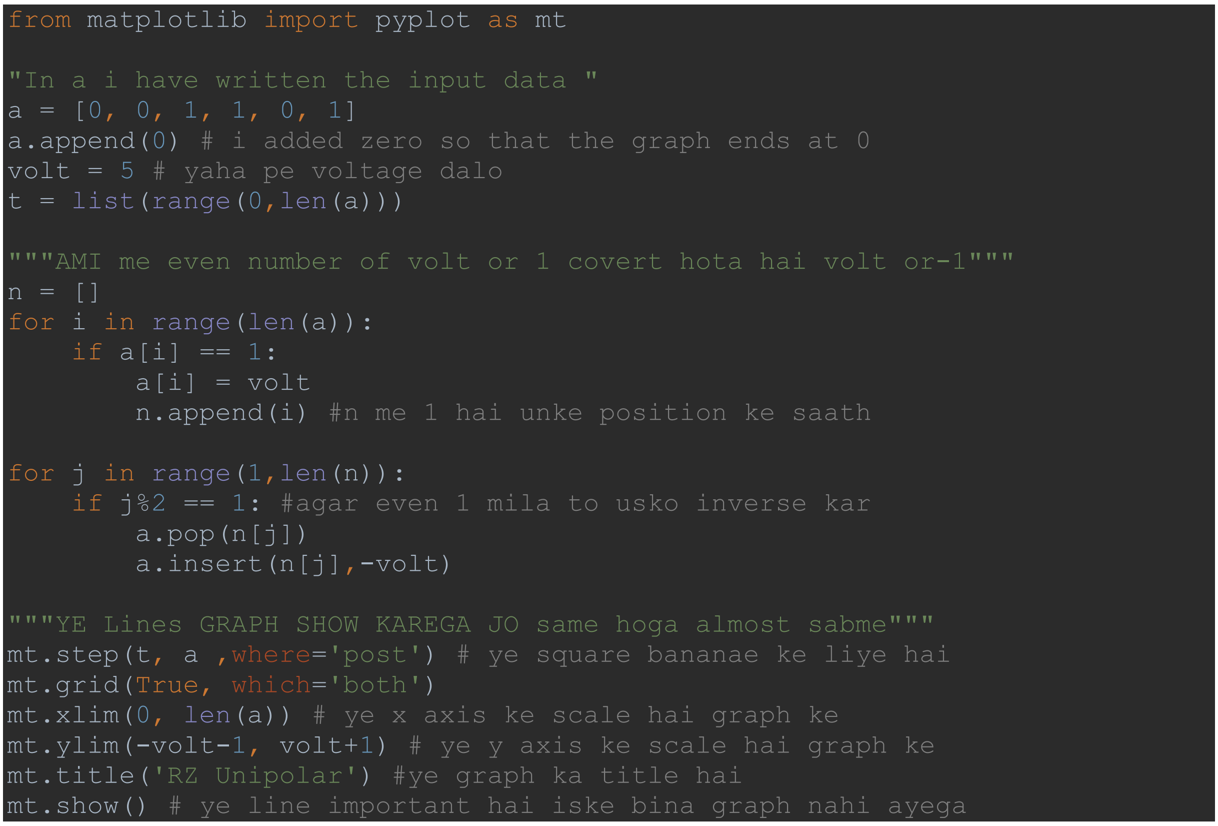
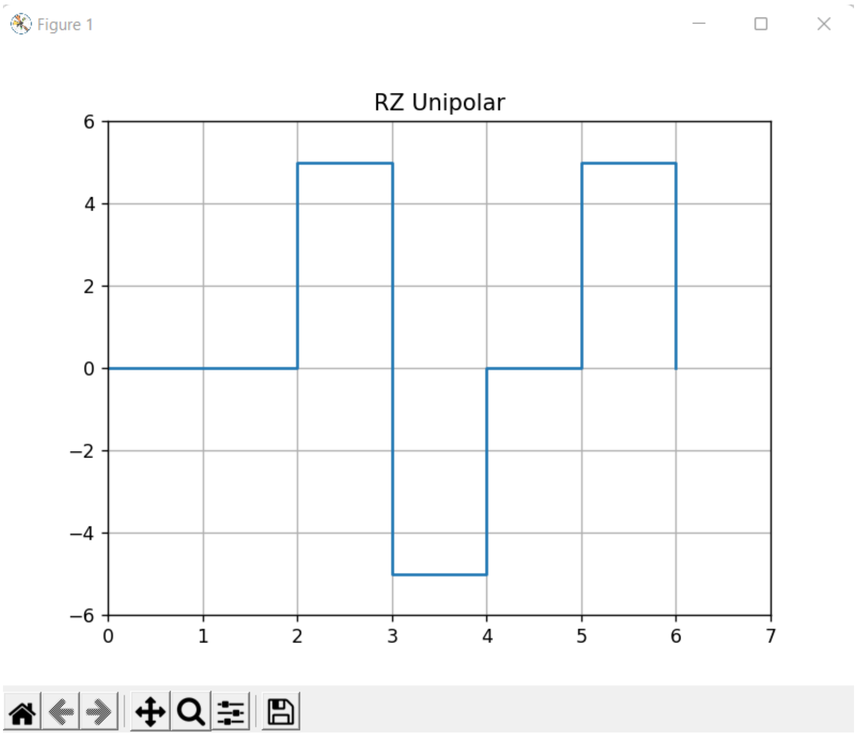


Figure 1
RZ Unipolar
6
4
2
0
−2
−4
−6
0
1
2
3
4
5
6

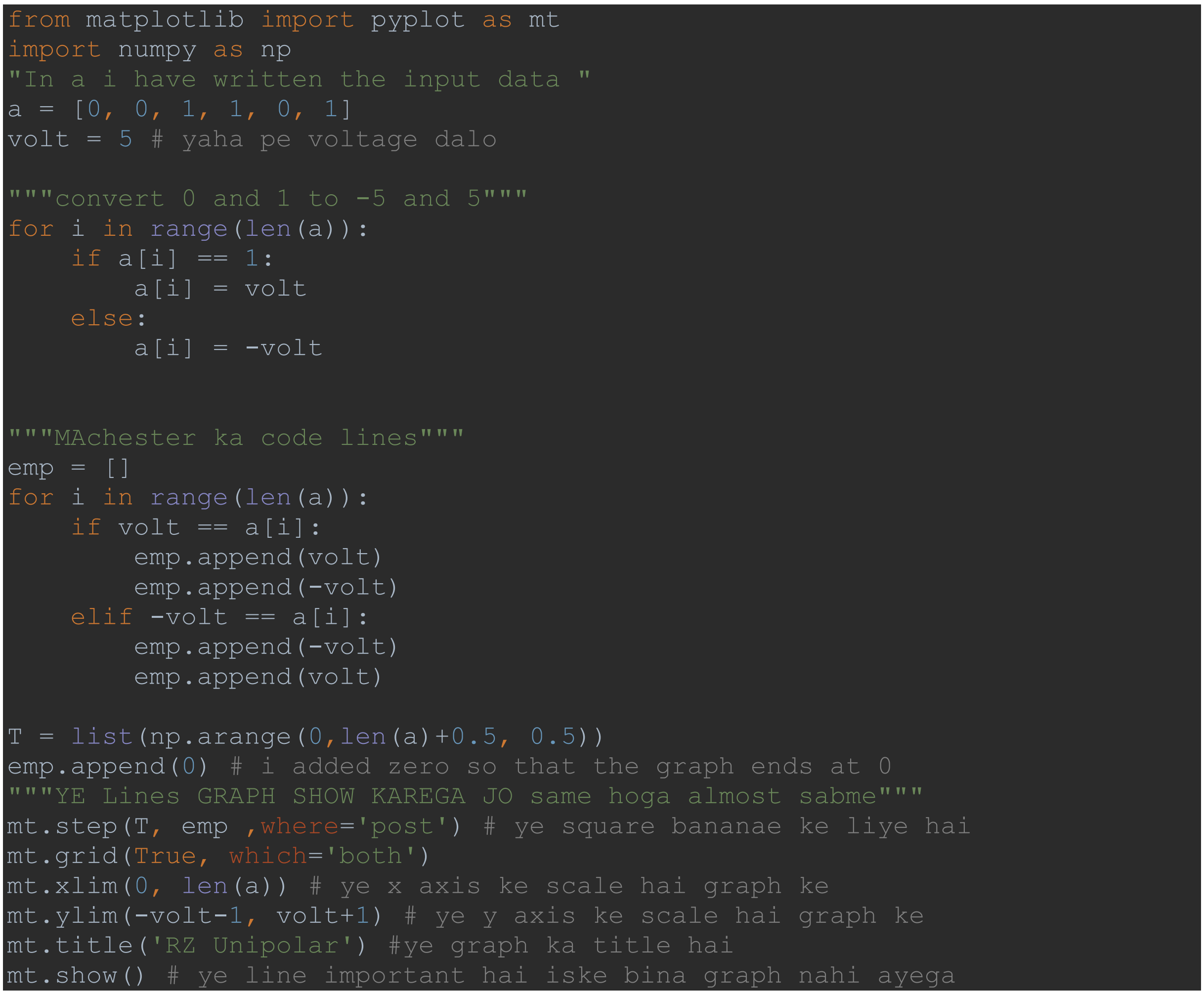
Q 12 Write a program for generating nrz AMI code for the data 001101



```
from matplotlib import pyplot as mt

"In a i have written the input data "
a = [0, 0, 1, 1, 0, 1]
a.append(0) # i added zero so that the graph ends at 0
volt = 5 # yaha pe voltage dalo
t = list(range(0,len(a)))

"""AMI me even number of volt or 1 covert hota hai volt or-1"""
n = []
for i in range(len(a)):
    if a[i] == 1:
        a[i] = volt
        n.append(i) #n me 1 hai unke position ke saath

for j in range(1,len(n)):
    if j%2 == 1: #agar even 1 mila to usko inverse kar
        a.pop(n[j])
        a.insert(n[j],-volt)

"""YE Lines GRAPH SHOW KAREGA JO same hoga almost sabme"""
mt.step(t, a ,where='post') # ye square bananae ke liye hai
mt.grid(True, which='both')
mt.xlim(0, len(a)) # ye x axis ke scale hai graph ke
mt.ylim(-volt-1, volt+1) # ye y axis ke scale hai graph ke
mt.title('RZ Unipolar') #ye graph ka title hai
mt.show() # ye line important hai iske bina graph nahi ayega
```



Q 13 Write a program for generating Manchester code for the data 001101



```
from matplotlib import pyplot as mt
import numpy as np
"In a i have written the input data "
a = [0, 0, 1, 1, 0, 1]
volt = 5 # yaha pe voltage dalo

"""convert 0 and 1 to -5 and 5"""
for i in range(len(a)):
    if a[i] == 1:
        a[i] = volt
    else:
        a[i] = -volt


"""MAchester ka code lines"""
emp = []
for i in range(len(a)):
    if volt == a[i]:
        emp.append(volt)
        emp.append(-volt)
    elif -volt == a[i]:
        emp.append(-volt)
        emp.append(volt)

T = list(np.arange(0,len(a)+0.5, 0.5))
emp.append(0) # i added zero so that the graph ends at 0
"""YE Lines GRAPH SHOW KAREGA JO same hoga almost sabme"""
mt.step(T, emp ,where='post') # ye square bananae ke liye hai
mt.grid(True, which='both')
mt.xlim(0, len(a)) # ye x axis ke scale hai graph ke
mt.ylim(-volt-1, volt+1) # ye y axis ke scale hai graph ke
mt.title('RZ Unipolar') #ye graph ka title hai
mt.show() # ye line important hai iske bina graph nahi ayega
```

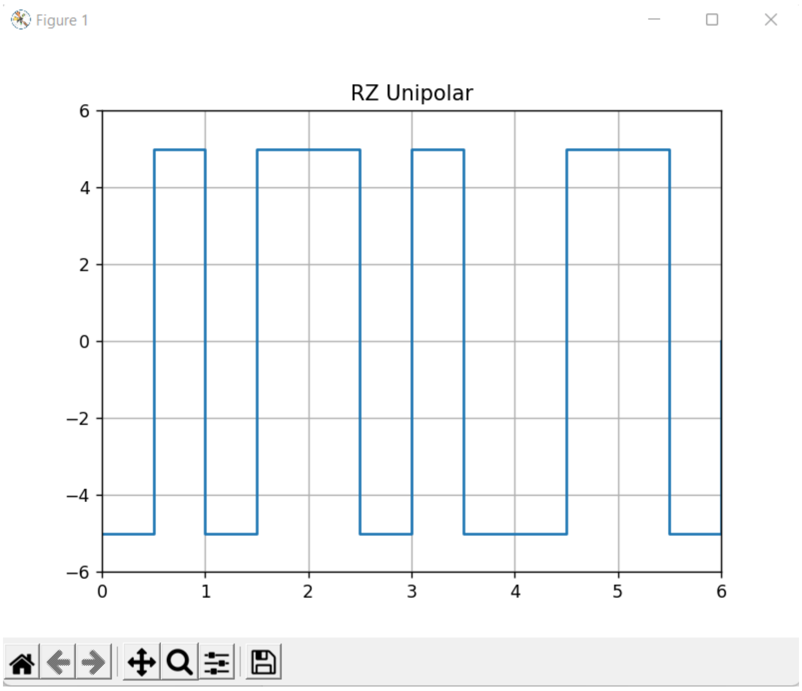


Figure 1
RZ Unipolar