**PRACTICAL 6**

**AIM**: Implement Simple Transposition encryption-decryption.

**Code**:

import numpy as np

rows = int(input("Enter the number of rows: "))

columns = int(input("Enter the number of columns: "))

l1 = np.empty((rows, columns),dtype=str)

l2 = np.empty((rows, columns),dtype=str)

l3 = np.empty((rows, columns),dtype=str)

for i in range(rows):

for j in range(columns):

l1[i][j] = input()

for i in range (rows):

for j in range(columns):

l2[i][j]=l1[j][i]

for i in range (rows):

for j in range(columns):

l3[i][j]=l2[j][i]

#n dimentional to single dimention

single\_dim\_list = [i for sublist in l2 for i in sublist]

single\_dim\_list1 = [i for sublist in l3 for i in sublist]

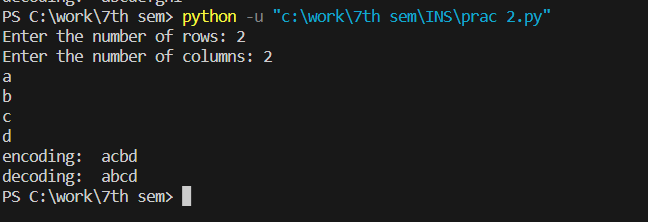
single\_dim\_string = ''.join(single\_dim\_list)

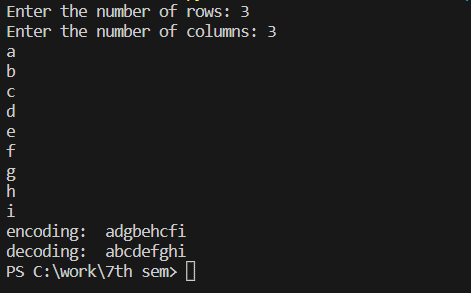
single\_dim\_string1 = ''.join(single\_dim\_list1)

print("encoding: ", single\_dim\_string)

print("decoding: ", single\_dim\_string1)

output:





**AIM**: Implement rail fence encryption-decryption.

def encode\_rail\_fence\_cipher(string, n):

rails = []

for i in range(n):

empty\_list = []

for j in range(len(string)):

empty\_list.append("")

rails.append(empty\_list)

row = 0

down = True

for j in range(len(string)):

rails[row][j] = string[j]

if down:

if row == n - 1:

down = False

row -= 1

else:

row += 1

else:

if row == 0:

down = True

row += 1

else:

row -= 1

encoded = ""

for i in range(n):

encoded = encoded + "".join(rails[i])

return encoded

original = "HelloHowAreYou"

print("\nOriginal:", original)

encoded = encode\_rail\_fence\_cipher(original, 3)

print("Encoded:", encoded)

def decode\_rail\_fence\_cipher(string, n):

rails = []

for i in range(n):

empty\_list = []

for j in range(len(string)):

empty\_list.append(" ")

rails.append(empty\_list)

row = 0

down = True

for j in range(len(string)):

rails[row][j] = "\_"

if down:

if row == n - 1:

down = False

row = row - 1

else:

row = row + 1

else:

if row == 0:

down = True

row = row + 1

else:

row = row - 1

count = 0

for i in range(n):

for j in range(len(string)):

if rails[i][j] == "\_":

rails[i][j] = string[count]

count = count + 1

decoded = ""

row = 0

down = True

for j in range(len(string)):

decoded = decoded + rails[row][j]

if down:

if row == n - 1:

down = False

row = row - 1

else:

row = row + 1

else:

if row == 0:

down = True

row = row + 1

else:

row = row - 1

return decoded

print("Decoded:", decode\_rail\_fence\_cipher(encoded, 3))

output:

