PRACTICAL 6

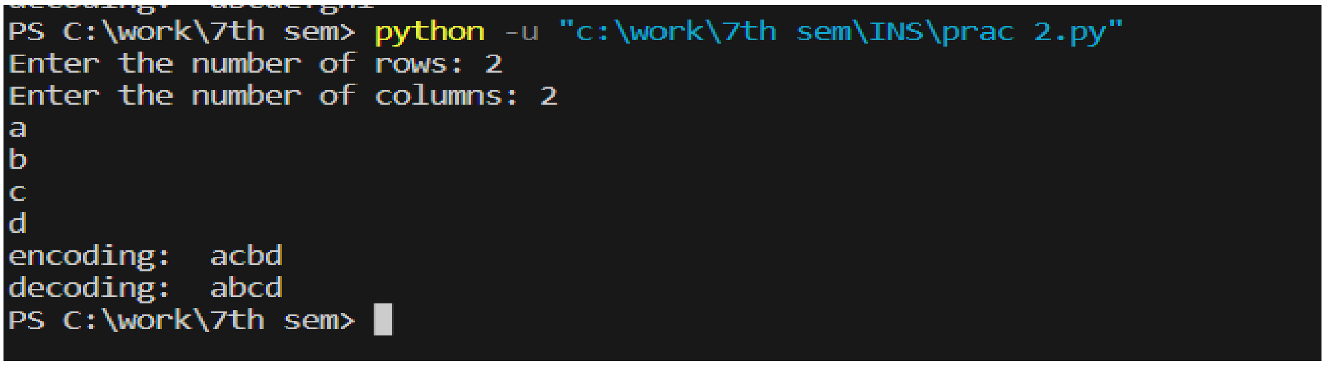
AIM: Implement Simple Transposition encryption-decryption.

Code:

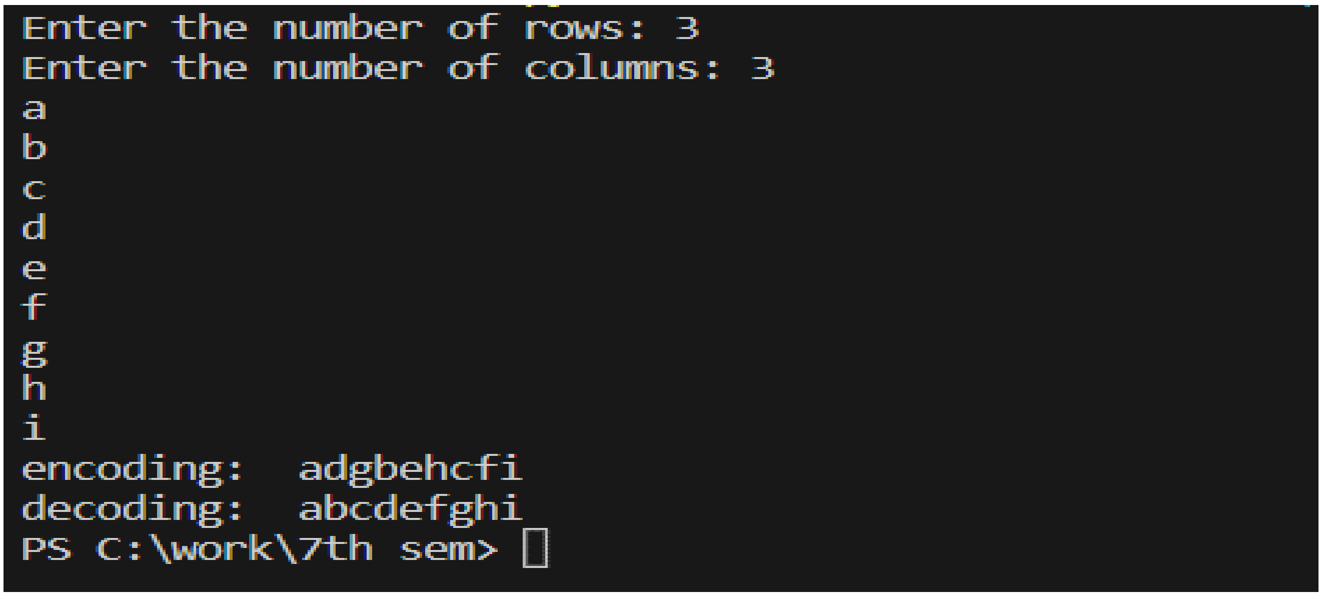
```
import numpy as np
rows = int(input("Enter the number of rows: "))
columns = int(input("Enter the number of columns: "))

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:



```
PS C:\work\7th sem> python -u "c:\work\7th sem\INS\prac 2.py"
Enter the number of rows: 2
Enter the number of columns: 2
a
b
c
d
encoding:  acbd
decoding:  abcd
PS C:\work\7th sem>
```



```
Enter the number of rows: 3
Enter the number of columns: 3
a
b
c
d
e
f
g
h
i
encoding:  adgbehcfi
decoding:  abcdefghi
PS C:\work\7th sem>
```

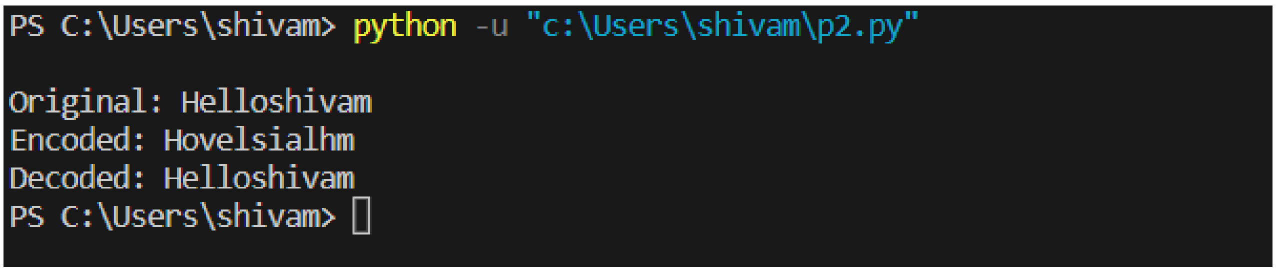
AIM: Implement rail fence encryption-decryption.

```
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:



```
PS C:\Users\shivam> python -u "c:\Users\shivam\p2.py"

Original: Helloshivam
Encoded: Hovelsialhm
Decoded: Helloshivam
PS C:\Users\shivam> 
```