**CSS EXPERIMENT 1**

input\_string = input("Enter a string: ").lower()

# Creating a dict

letter\_number\_map = {

    'a': 0, 'b': 1, 'c': 2, 'd': 3, 'e': 4, 'f': 5, 'g': 6, 'h': 7, 'i': 8, 'j': 9,

    'k': 10, 'l': 11, 'm': 12, 'n': 13, 'o': 14, 'p': 15, 'q': 16, 'r': 17, 's': 18,

    't': 19, 'u': 20, 'v': 21, 'w': 22, 'x': 23, 'y': 24, 'z': 25

}

key = int(input("Enter a key: "))

choice = int(input("1. Encrypt \n2. Decrypt: "))

if choice == 1:  # Encrypt

    encrypted\_result = ""

    for letter in input\_string:

        if letter.isalpha():

            number = letter\_number\_map[letter]

            encrypted\_number = (number + key) % 26  # Adding the key to the letter number

            encrypted\_letter = [k for k, v in letter\_number\_map.items() if v == encrypted\_number][0] #k represents the letter and v represents the number.

            encrypted\_result += encrypted\_letter

    print("Given String: ",input\_string)

    print("Encrypted result:", encrypted\_result)

elif choice == 2:  # Decrypt

    decrypted\_result = ""

    for letter in input\_string:

        if letter.isalpha():

            number = letter\_number\_map[letter]

            decrypted\_number = (number - key) % 26  # Subtracting the key from the letter number

            decrypted\_letter = [k for k, v in letter\_number\_map.items() if v == decrypted\_number][0]

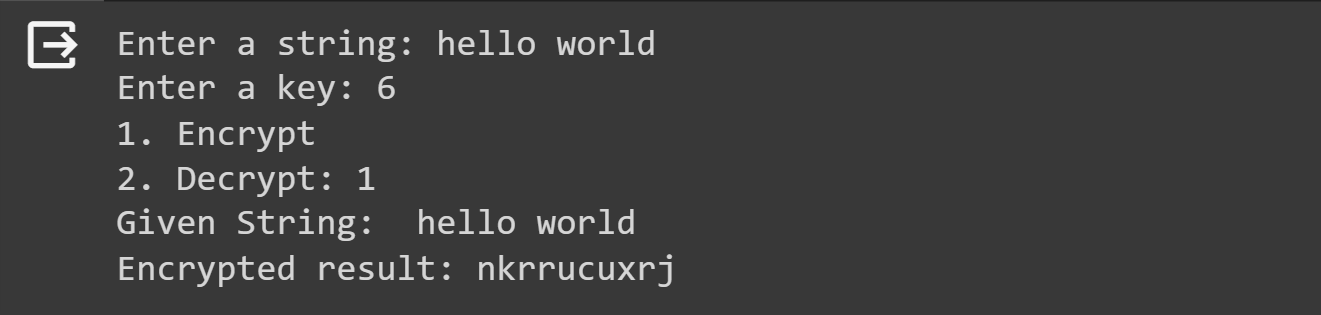
            decrypted\_result += decrypted\_letter

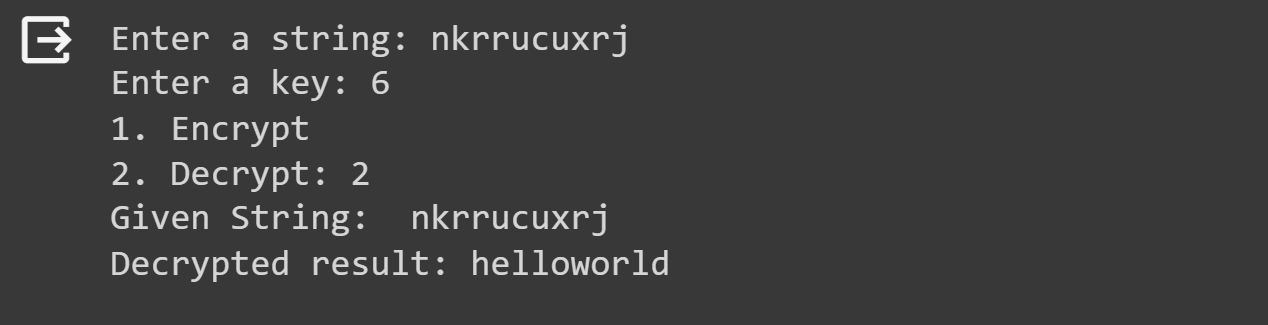
    print("Given String: ",input\_string)

    print("Decrypted result:", decrypted\_result)

else:

    print("Invalid choice. Please enter 1 to encrypt or 2 to decrypt.")



****

input\_string = input("Enter a string: ").lower()

# Creating a dict

letter\_number\_map = {

    'a': 0, 'b': 1, 'c': 2, 'd': 3, 'e': 4, 'f': 5, 'g': 6, 'h': 7, 'i': 8, 'j': 9,

    'k': 10, 'l': 11, 'm': 12, 'n': 13, 'o': 14, 'p': 15, 'q': 16, 'r': 17, 's': 18,

    't': 19, 'u': 20, 'v': 21, 'w': 22, 'x': 23, 'y': 24, 'z': 25

}

key = int(input("Enter a key: "))

choice = int(input("1. Encrypt \n2. Decrypt: "))

if choice == 1:  # Encrypt

    encrypted\_result = ""

    for letter in input\_string:

        if letter.isalpha():

            number = letter\_number\_map[letter]

            encrypted\_number = (number \* key) % 26  # Multiplying the key with the letter number

            for k, v in letter\_number\_map.items():  #k represents the letter and v represents the number

                if v == encrypted\_number:

                    encrypted\_letter = k

                    break

            encrypted\_result += encrypted\_letter

    print("Given String: ", input\_string)

    print("Encrypted result:", encrypted\_result)

elif choice == 2:  # Decrypt

    decrypted\_result = ""

    for letter in input\_string:

        if letter.isalpha():

            number = letter\_number\_map[letter]

            # Finding the multiplicative inverse of the key

            inverse\_key = 0

            for i in range(26):

                if (key \* i) % 26 == 1:

                    inverse\_key = i

                    break

            decrypted\_number = (number \* inverse\_key) % 26  # Multiplying the letter number with the inverse of the key

            for k, v in letter\_number\_map.items():

                if v == decrypted\_number:

                    decrypted\_letter = k

                    break

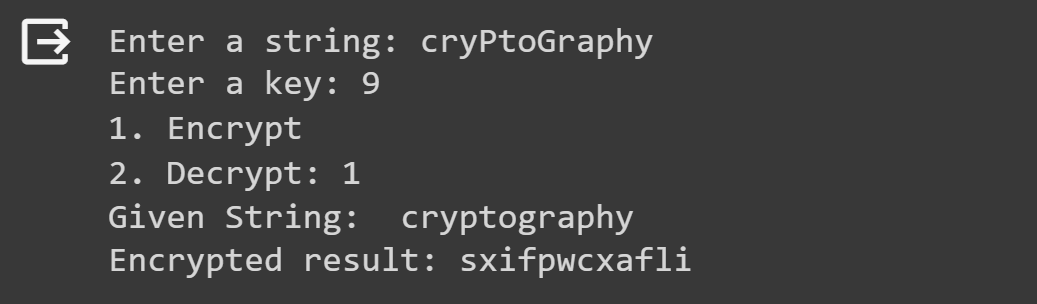
            decrypted\_result += decrypted\_letter

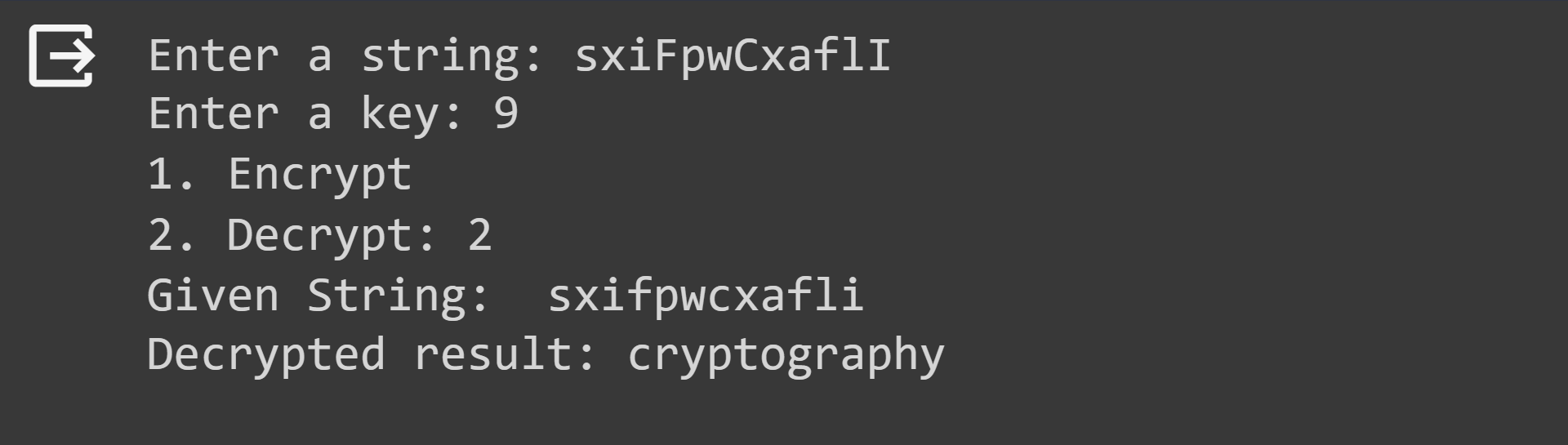
    print("Given String: ", input\_string)

    print("Decrypted result:", decrypted\_result)

else:

    print("Invalid choice. Please enter 1 to encrypt or 2 to decrypt.")





Row Transformation

input\_string = input("Enter a string: ").lower()

string\_without\_spaces = input\_string.replace(" ", "")

print("String without spaces:", string\_without\_spaces)

# Add 'x' at the end if the length is not even

if len(string\_without\_spaces) % 2 != 0:

    string\_without\_spaces += 'x'

print("Modified string:", string\_without\_spaces)

# Calculate the midpoint index

midpoint = len(string\_without\_spaces) // 2

# Divide the string into two parts

part1 = string\_without\_spaces[:midpoint]

part2 = string\_without\_spaces[midpoint:]

# Convert part1 and part2 into lists

list\_part1 = list(part1)

list\_part2 = list(part2)

result = ""

for i in range(0, len(list\_part1), 2):

    result += f"{list\_part1[i]} {list\_part2[i]} "

    if i + 1 < len(list\_part1):

        result += f"{list\_part1[i + 1]} {list\_part2[i + 1]} "

print("Encrypted string:", result)

# Create two strings s1 and s2 with even and odd indices from result

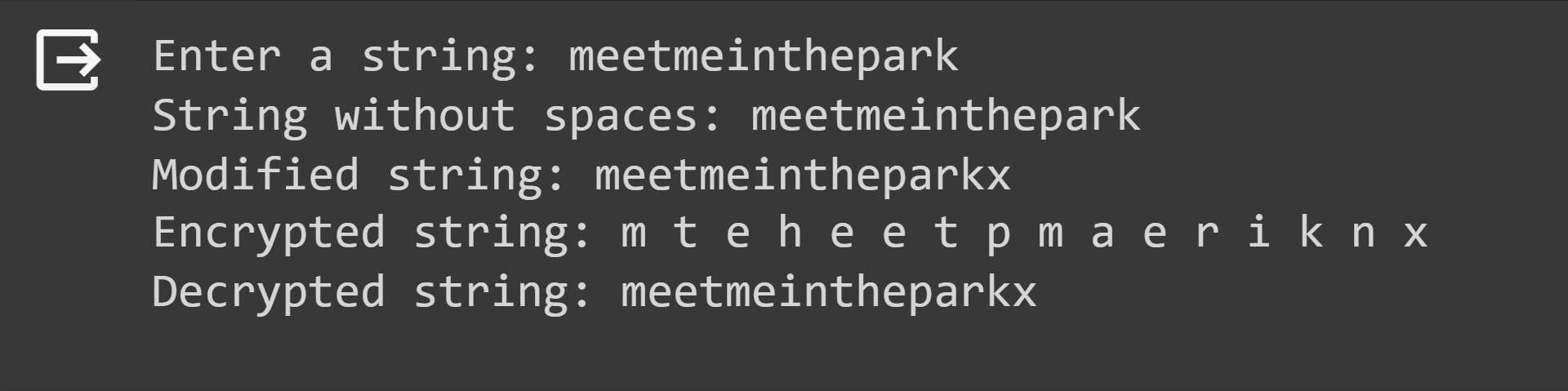
s1 = ''.join(result.split()[::2])

s2 = ''.join(result.split()[1::2])

# Create a variable decrypt by concatenating s1 and s2

decrypt = s1 + s2

print("Decrypted string:", decrypt)



Keyed

input\_string = input("Enter a string: ").lower()

string\_without\_spaces = input\_string.replace(" ", "")

print("String without spaces:", string\_without\_spaces)

# Add 'x' at the end if the length is not even

if len(string\_without\_spaces) % 2 != 0:

    string\_without\_spaces += 'x'

print("Modified string:", string\_without\_spaces)

# Calculate the midpoint index

midpoint = len(string\_without\_spaces) // 2

# Put even index characters in list\_part1 and odd index characters in list\_part2

list\_part1 = [part\_list[i] for part\_list in lists for i in range(0, midpoint, 2)]

list\_part2 = [part\_list[i] for part\_list in lists for i in range(1, midpoint, 2)]

print("List part1 (even index characters):", list\_part1)

print("List part2 (odd index characters):", list\_part2)

encrypt = ''.join(list\_part1 + list\_part2)

print("Encrypted 'encrypt':", encrypt)

# Calculate the midpoint index

midpoint = len(encrypt) // 2

# Separate the string into two parts

part1 = encrypted\_string[:midpoint]

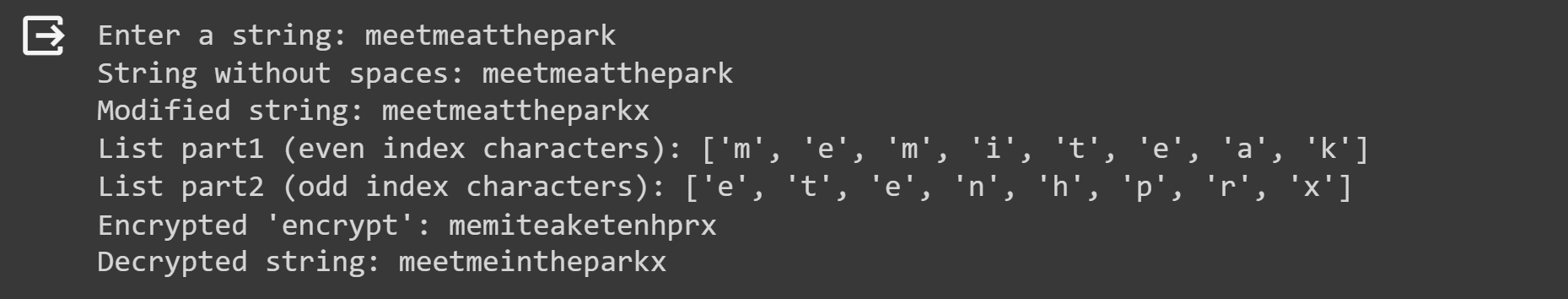
part2 = encrypted\_string[midpoint:]

# Interleave the characters from part1 and part2 to reconstruct the original string

decrypted\_string = ''.join([char1 + char2 for char1, char2 in zip(part1, part2)])

# Print the decrypted string

print("Decrypted string:", decrypted\_string)



input\_string = input("Enter a string: ").lower()

div = int(input("Enter a number: "))

# Get column order from the user

column\_order = input("Enter a list of numbers separated by space: ")

numbers = [int(x) for x in column\_order.split()]

string\_without\_spaces = input\_string.replace(" ", "")

print("String without spaces:", string\_without\_spaces)

# Add 'x' at the end if the length is not evenly divisible by div

remainder = len(string\_without\_spaces) % div

if remainder != 0:

    string\_without\_spaces += 'x' \* (div - remainder)

print("Modified string:", string\_without\_spaces)

# Calculate the number of rows in the matrix

num\_rows = len(string\_without\_spaces) // div

# Create a matrix with div columns and num\_rows rows

matrix = [[''] \* div for \_ in range(num\_rows)]

# Insert characters of string\_without\_spaces into the matrix

index = 0

for row in range(num\_rows):

    for col in range(div):

        if index < len(string\_without\_spaces):

            matrix[row][col] = string\_without\_spaces[index]

            index += 1

print("Matrix:")

for row in matrix:

    print(row)

# Use the user-provided column order

column\_order = [col % div for col in numbers]

new\_matrix = [[matrix[row][col] for col in column\_order] for row in range(len(matrix))]

# Print the new matrix

print("New Matrix:")

for row in new\_matrix:

    print(row)

encrypt = ''.join([''.join(row) for row in new\_matrix])

# Print the encrypted string

print("Encrypted String:", encrypt)

