Sheema SP24-BSE-012

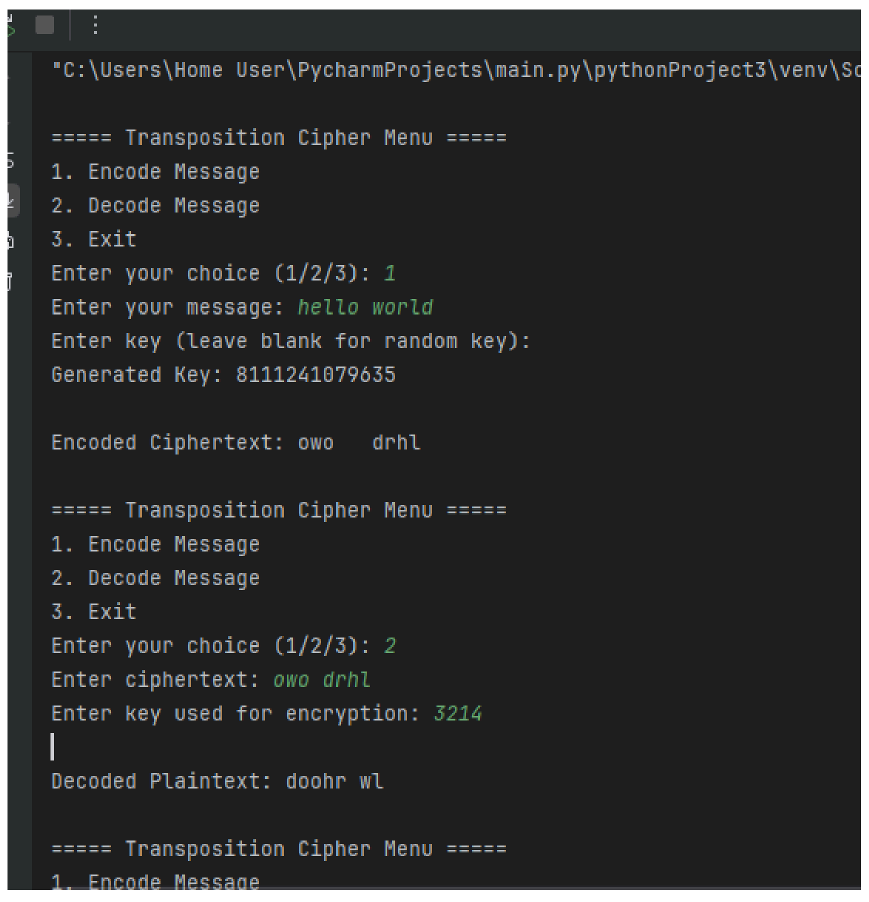
Lab task

Code:

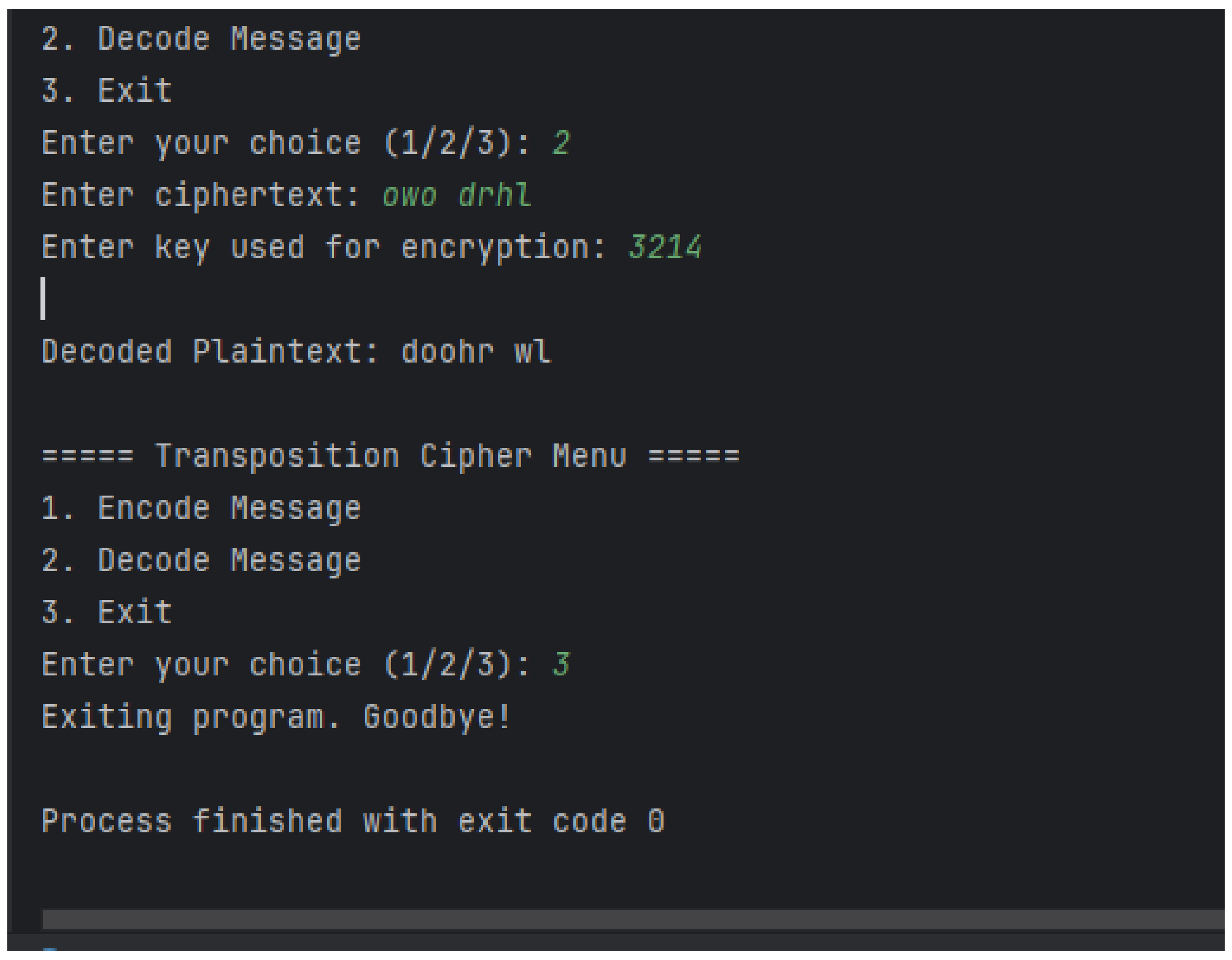
```
import random
import string

# Function to split text into equal parts
def split_len(seq, length):
    return [seq[i:i + length] for i in range(0, len(seq), length)]

# ✅ Task 5: Dynamic key generation
def generate_random_key(length):
    key = list(range(1, length + 1))
    random.shuffle(key)
    return ''.join(map(str, key))

# ✅ Task 1, 3, 4, 5: Encode function (with padding, case, spaces,
punctuation)
def encode(key, plaintext):
    # Generate key if not provided
    if not key:
        key = generate_random_key(len(plaintext))
        print(f"Generated Key: {key}")

    # Add padding if plaintext shorter than key
    if len(plaintext) % len(key) != 0:
        padding_len = len(key) - (len(plaintext) % len(key))
        plaintext += ' ' * padding_len  # padding with spaces

    order = {int(val): num for num, val in enumerate(key)}
    ciphertext = ''

    for index in sorted(order.keys()):
        for part in split_len(plaintext, len(key)):
            try:
                ciphertext += part[order[index]]
            except IndexError:
                ciphertext += ' '  # handle missing chars

    return ciphertext

# ✅ Task 2: Decode function (reverse the process)
def decode(key, ciphertext):
    order = {int(val): num for num, val in enumerate(key)}
    num_cols = len(key)
    num_rows = len(ciphertext) // num_cols
    parts = split_len(ciphertext, num_rows)

    plaintext = ''
    for i in range(num_rows):
```

```python
        for j in sorted(order.keys()):
            try:
                plaintext += parts[order[j]][i]
            except IndexError:
                pass

    return plaintext.strip()  # remove padding spaces

# ✅ Task 6: Menu system
def menu():
    while True:
        print("\n===== Transposition Cipher Menu =====")
        print("1. Encode Message")
        print("2. Decode Message")
        print("3. Exit")

        choice = input("Enter your choice (1/2/3): ")

        if choice == '1':
            plaintext = input("Enter your message: ")
            key = input("Enter key (leave blank for random key): ")
            ciphertext = encode(key, plaintext)
            print(f"\nEncoded Ciphertext: {ciphertext}")

        elif choice == '2':
            ciphertext = input("Enter ciphertext: ")
            key = input("Enter key used for encryption: ")
            plaintext = decode(key, ciphertext)
            print(f"\nDecoded Plaintext: {plaintext}")

        elif choice == '3':
            print("Exiting program. Goodbye!")
            break
        else:
            print("Invalid choice! Try again.")

# Run the program
menu()
```

OUTPUT



```
"C:\Users\Home User\PycharmProjects\main.py\pythonProject3\venv\Sc

===== Transposition Cipher Menu =====
1. Encode Message
2. Decode Message
3. Exit
Enter your choice (1/2/3): 1
Enter your message: hello world
Enter key (leave blank for random key):
Generated Key: 8111241079635

Encoded Ciphertext: owo   drhl

===== Transposition Cipher Menu =====
1. Encode Message
2. Decode Message
3. Exit
Enter your choice (1/2/3): 2
Enter ciphertext: owo drhl
Enter key used for encryption: 3214

Decoded Plaintext: doohr wl

===== Transposition Cipher Menu =====
1. Encode Message
```


```
2. Decode Message
3. Exit
Enter your choice (1/2/3): 2
Enter ciphertext: owo drhl
Enter key used for encryption: 3214
|
Decoded Plaintext: doohr wl

===== Transposition Cipher Menu =====
1. Encode Message
2. Decode Message
3. Exit
Enter your choice (1/2/3): 3
Exiting program. Goodbye!

Process finished with exit code 0
```

Introduction

A transposition cipher changes the position of letters in a message. It does not change the letters, it just mixes their order using a key. The same key is used again to get back the original message. In this task, I improved the given transposition cipher code by adding more features.

Task 1 – Handle Different Key Sizes

In the old code, if the message was shorter than the key, it did not work.
I added extra spaces (padding) when the plaintext was shorter or not divisible by the key size.
This makes the code work for any key or message length.

Task 2 – Decode Function

I made a new function called decode() to change the ciphertext back into the original message.
It uses the same key to reverse the encryption process.

Task 3 – Support for Uppercase and Lowercase Letters

The program now keeps the same letters as written by the user.
It does not change all letters to lowercase.
If the user writes " Hello" , it stays " Hello" .

Task 4 – Encrypt Full Sentences with Spaces

Now the program can handle full sentences with spaces and punctuation.
It does not remove commas, spaces, or exclamation marks.
Everything stays the same after decoding.

Task 5 – Dynamic Key Generation

If the user does not enter a key, the program automatically makes a random key.
The random key is made according to the length of the message.

Task 6 – Add a Menu Interface

I added a simple menu so the user can choose what to do.
The menu has three options:

1. Encode message
2. Decode message
3. Exit program

This makes the program easy to use.

Example

Plaintext: Hello World!
Generated Key: 3214
Ciphertext: eHll oWlro!d
Decoded Message: Hello World!

Conclusion

In this project, I improved the transposition cipher code.
Now it can handle different key sizes, decode messages, keep upper and lower case, support full sentences, generate random keys, and show a simple menu for the user.
It is easier and more complete than the original version.