

Screenshot Code:

```

C:\Users\irine> OneDrive\Deskto... python2.py > main
1  def encrypt(key, plaintext, matrix=None, ciphertext=''):
2      message = plaintext
3      while len(message) > 0:
4          chunk_size = min(max_col_length, len(message))
5          current_chunk = message[:chunk_size]
6          message = message[chunk_size:]
7          row = list(current_chunk) + ['_'] * (max_col_length - len(current_chunk))
8          matrix.append(row)
9
10         matrix = [[row[i] for row in matrix if row[i] != "_"] for i in range(len(matrix[0]))]
11
12         for col in matrix:
13             for char in col:
14                 ciphertext += char
15
16         return ciphertext.replace(" ", "_")
17
18 def decrypt(key, ciphertext, matrix=None, position=0, plaintext=''):
19     if matrix is None:
20         matrix = []
21
22     num_rows = len(ciphertext) // key
23     extra_chars = len(ciphertext) % key
24
25     for i in range(key):
26         col_length = num_rows + 1 if i < extra_chars else num_rows
27         current_col = list(ciphertext[position:position + col_length])
28         position += col_length
29         matrix.append(current_col)
30
31     matrix = [[row[i] for row in matrix if i < len(row) and row[i] != "_"]
32               for i in range(num_rows + 1)]
33
34     for row in matrix:
35         for char in row:
36             plaintext += char
37
38     return plaintext.replace("_", " ")
39
40 def main():
41     plaintext = input("Enter the plaintext message: ")
42     key = int(input("Enter a key: "))
43
44     encrypted_text = encrypt(key, plaintext)
45     print("Encrypted Text:", encrypted_text)
46
47     decrypted_text = decrypt(key, encrypted_text)
48     print("Decrypted Text:", decrypted_text)
49
50 if __name__ == "__main__":
51     main()

```

Written Code:

```
def encrypt(key, plaintext, matrix=None, ciphertext=''):
    if matrix is None:
        matrix = []

    max_col_length = len(plaintext[:key])

    message = plaintext
    while len(message) > 0:
        chunk_size = min(max_col_length, len(message))
        current_chunk = message[:chunk_size]
        message = message[chunk_size:]
        row = list(current_chunk) + ['_'] * (max_col_length - len(current_chunk))
        matrix.append(row)

    matrix = [[row[i] for row in matrix if row[i] != "_"] for i in range(len(matrix[0]))]

    for col in matrix:
        for char in col:
            ciphertext += char

    return ciphertext.replace(" ", "_")

def decrypt(key, ciphertext, matrix=None, position=0, plaintext=''):
    if matrix is None:
        matrix = []

    num_rows = len(ciphertext) // key
    extra_chars = len(ciphertext) % key

    for i in range(key):
        col_length = num_rows + 1 if i < extra_chars else num_rows
        current_col = list(ciphertext[position:position + col_length])
        position += col_length
        matrix.append(current_col)

    matrix = [[row[i] for row in matrix if i < len(row) and row[i] != "_"]
               for i in range(num_rows + 1)]

    for row in matrix:
        for char in row:
            plaintext += char

    return plaintext.replace("_", " ")

def main():
    plaintext = input("Enter the plaintext message: ")
    key = int(input("Enter a key: "))

    encrypted_text = encrypt(key, plaintext)
```

```
print("Encrypted Text:", encrypted_text)

decrypted_text = decrypt(key, encrypted_text)
print("Decrypted Text:", decrypted_text)

if __name__ == "__main__":
    main()
```

Output:

```
PS C:\xampp\htdocs\E-Portfolio> & C:/Users/irine/AppData/Local/Microsof
Enter the plaintext message: Love is not blind.
Enter a key: 8
Encrypted Text: Lndoo.vte__bils_i_n
Decrypted Text: Loveisnotblind.
PS C:\xampp\htdocs\E-Portfolio> █
```