

Name - Yash Lakhtariya  
Enrollment number - 21162101012  
Branch - CBA      Batch - 61  
INS Practical 3

**Aim** : Alice wants to send some confidential information to Bob over a secure network, you have to create perform following task :

### 1) Provide Security using Caesar Cipher Algorithm

Code :

```
import YSL_io

class YSL:
    @staticmethod
    def encrypt(txt, shift: int):
        ncrptd = ""
        for char in txt:
            if char.isalpha():
                if char.isupper():
                    ncrptd += chr((ord(char) + shift - 65) % 26 + 65)
                else:
                    ncrptd += chr((ord(char) + shift - 97) % 26 + 97)
            else:
                ncrptd += char
        return ncrptd

    @staticmethod
    def decrypt(txt, shift):
        return YSL.encrypt(txt, -shift)

def perform_encryption():
    input_string = YSL_io.inputRED("\n\tEnter the string to encrypt : ")
    shift_value = int(YSL_io.inputBLU("\tEnter the key value : "))
    encrypted_string = YSL.encrypt(str(input_string), shift_value)
```

Name - Yash Lakhtariya  
Enrollment number - 21162101012  
Branch - CBA      Batch - 61  
INS Practical 3

```
YSL_io.printMGNTA("\n\tEncrypted string", end=" : ")
print(encrypted_string)

def perform_decryption():
    input_string = YSL_io.inputRED("\n\tEnter the string to decrypt : ")
    shift_value = int(YSL_io.inputBLU("\tEnter the key value : "))
    decrypted_string = YSL.decrypt(str(input_string), shift_value)
    YSL_io.printMGNTA("\n\tDecrypted string", end=" : ")
    print(decrypted_string)

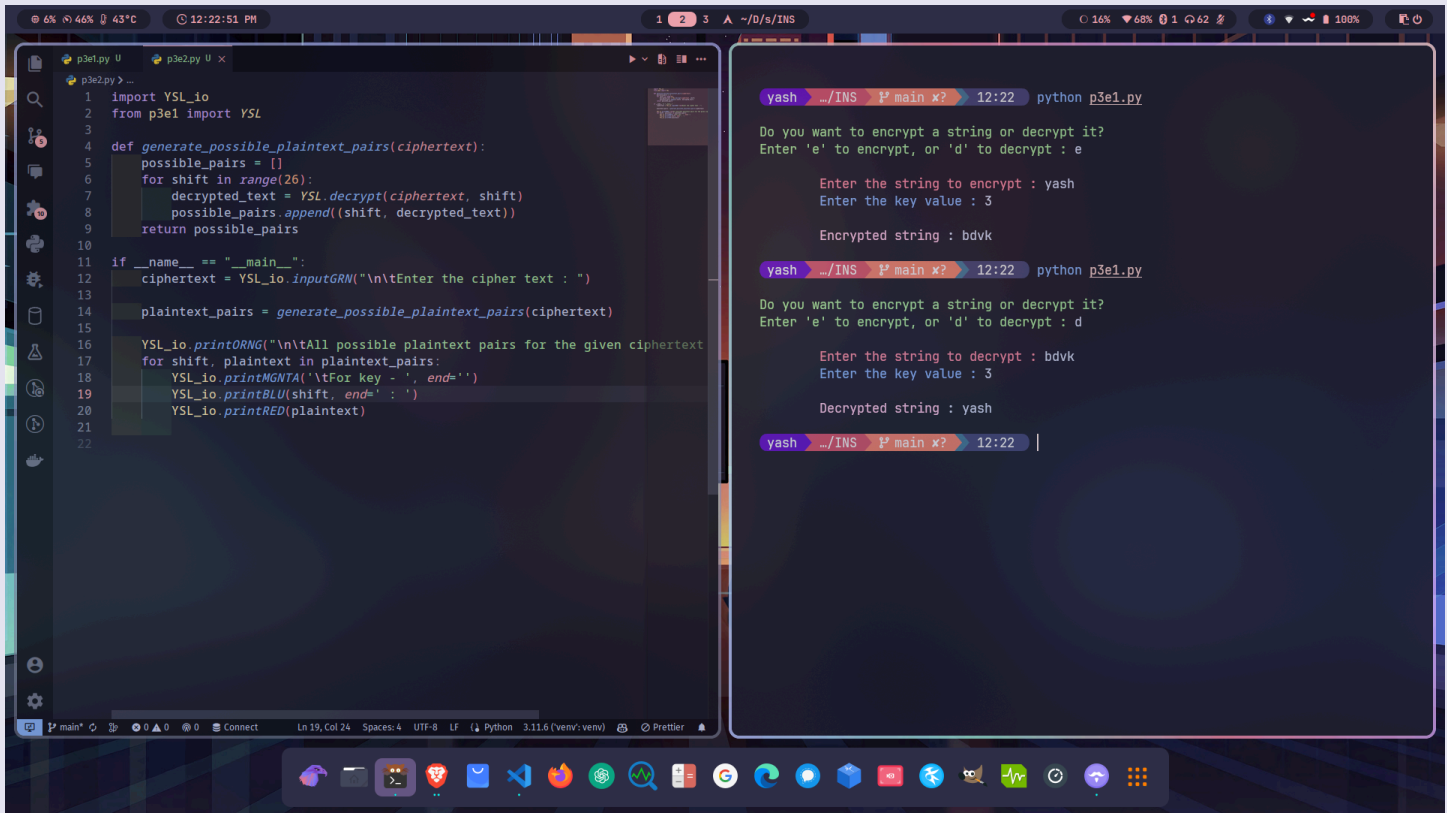
if __name__ == "__main__":
    nxt = YSL_io.inputGRN(
        "\nDo you want to encrypt a string or decrypt it?\nEnter 'e' to
encrypt, or 'd' to decrypt : "
    )
    while nxt.lower() not in ["e", "d"]:
        nxt = YSL_io.inputGRN(
            "\nInvalid input!\nEnter 'e' to encrypt, or 'd' to decrypt : "
        )

    if nxt.lower() == "e":
        perform_encryption()

    if nxt.lower() == "d":
        perform_decryption()
```

Name - Yash Lakhtariya  
Enrollment number - 21162101012  
Branch - CBA      Batch - 61  
INS Practical 3

Output :



The image shows a VS Code editor with a Python file named `p3e1.py` and a terminal window. The Python script implements a Caesar cipher that generates possible plaintext pairs for a given ciphertext by shifting each character by 26 possible values. The terminal shows the script being executed, with the user entering 'e' to encrypt, 'yash' as the string, and '3' as the key, resulting in the encrypted string 'bdvk'. In a second run, the user enters 'd' to decrypt, 'bdvk' as the string, and '3' as the key, resulting in the decrypted string 'yash'.

```
1 import YSL_io
2 from p3e1 import YSL
3
4 def generate_possible_plaintext_pairs(ciphertext):
5     possible_pairs = []
6     for shift in range(26):
7         decrypted_text = YSL.decrypt(ciphertext, shift)
8         possible_pairs.append((shift, decrypted_text))
9     return possible_pairs
10
11 if __name__ == "__main__":
12     ciphertext = YSL_io.inputGRN("\n\tEnter the cipher text : ")
13
14     plaintext_pairs = generate_possible_plaintext_pairs(ciphertext)
15
16     YSL_io.printORNG("\n\tAll possible plaintext pairs for the given ciphertext
17 for shift, plaintext in plaintext_pairs:
18     YSL_io.printMGNTA('\tFor key - ', end='')
19     YSL_io.printBLU(shift, end=' : ')
20     YSL_io.printRED(plaintext)
21
22
```

Terminal Output:

```
yash ~/INS $ python p3e1.py
Do you want to encrypt a string or decrypt it?
Enter 'e' to encrypt, or 'd' to decrypt : e

Enter the string to encrypt : yash
Enter the key value : 3

Encrypted string : bdvk

yash ~/INS $ python p3e1.py
Do you want to encrypt a string or decrypt it?
Enter 'e' to encrypt, or 'd' to decrypt : d

Enter the string to decrypt : bdvk
Enter the key value : 3

Decrypted string : yash

yash ~/INS $
```

Name - Yash Lakhtariya  
Enrollment number - 21162101012  
Branch - CBA      Batch - 61  
INS Practical 3

## 2) Find the all possible Cipher Text & Plaintext pairs

Code :

```
import YSL_io
from p3e1 import YSL

def generate_possible_plaintext_pairs(ciphertext):
    possible_pairs = []
    for shift in range(26):
        decrypted_text = YSL.decrypt(ciphertext, shift)
        possible_pairs.append((shift, decrypted_text))
    return possible_pairs

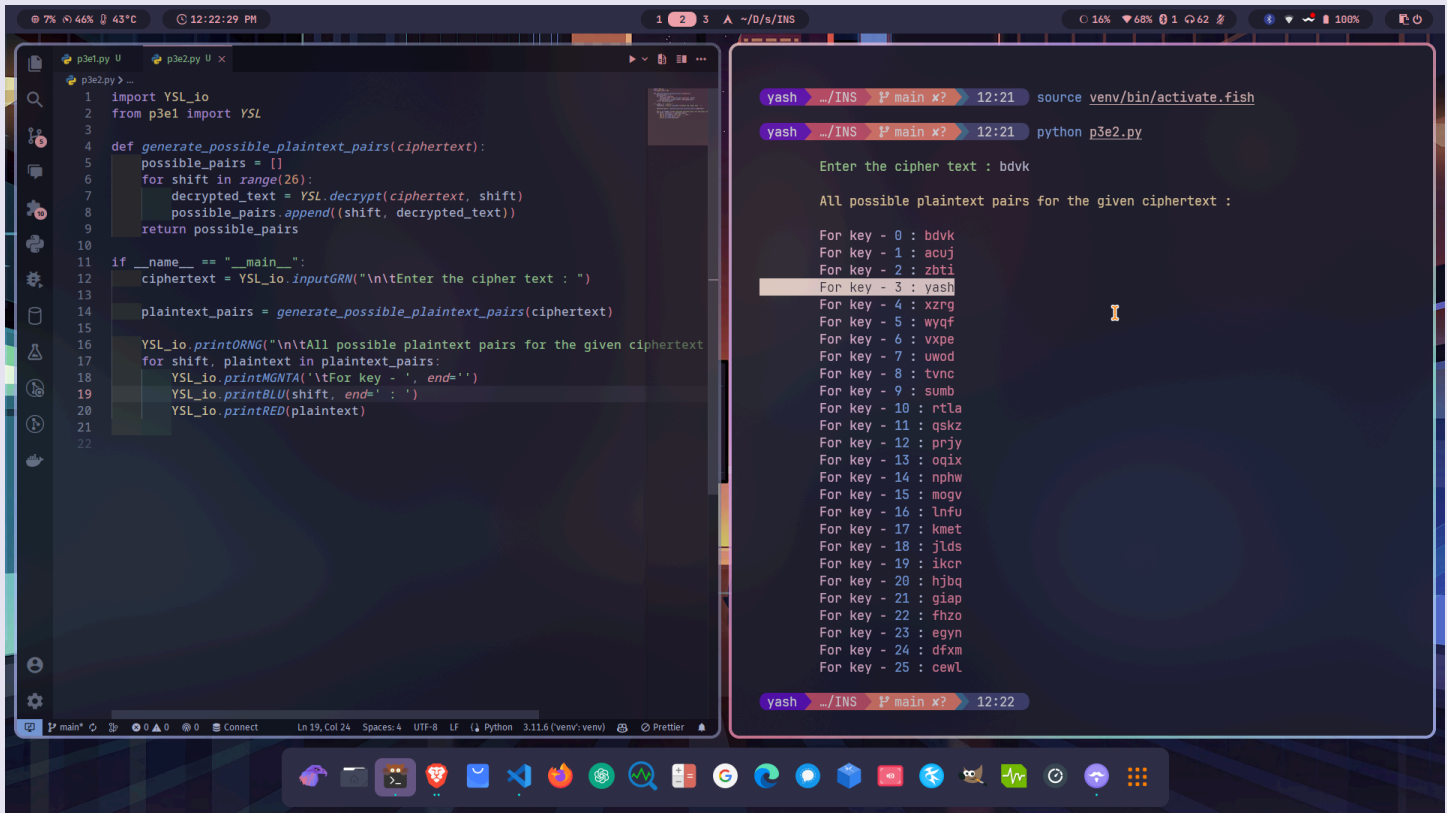
if __name__ == "__main__":
    ciphertext = YSL_io.inputGRN("\n\tEnter the cipher text : ")

    plaintext_pairs = generate_possible_plaintext_pairs(ciphertext)

    YSL_io.printORNG("\n\tAll possible plaintext pairs for the given
ciphertext : \n")
    for shift, plaintext in plaintext_pairs:
        YSL_io.printMGNTA('\tFor key - ', end='')
        YSL_io.printBLU(shift, end=' : ')
        YSL_io.printRED(plaintext)
```

Name - Yash Lakhtariya  
Enrollment number - 21162101012  
Branch - CBA      Batch - 61  
INS Practical 3

Output :



The screenshot shows a code editor on the left and a terminal on the right. The code editor displays a Python script named `p3e2.py` that implements a Caesar cipher decryption. The script defines a function `generate_possible_plaintext_pairs(ciphertext)` that iterates over all 26 possible shifts, decrypts the ciphertext for each shift, and returns a list of possible plaintext pairs. The main part of the script prompts the user to enter a ciphertext, calls the function, and prints all possible plaintext pairs for the given ciphertext.

```
1 import YSL_io
2 from p3e1 import YSL
3
4 def generate_possible_plaintext_pairs(ciphertext):
5     possible_pairs = []
6     for shift in range(26):
7         decrypted_text = YSL.decrypt(ciphertext, shift)
8         possible_pairs.append((shift, decrypted_text))
9     return possible_pairs
10
11 if __name__ == "__main__":
12     ciphertext = YSL_io.inputGRN("\nEnter the cipher text : ")
13
14     plaintext_pairs = generate_possible_plaintext_pairs(ciphertext)
15
16     YSL_io.printORNG("\nAll possible plaintext pairs for the given ciphertext")
17     for shift, plaintext in plaintext_pairs:
18         YSL_io.printMGNTA("\tFor key - ", end='')
19         YSL_io.printBLU(shift, end=' : ')
20         YSL_io.printRED(plaintext)
21
22
```

The terminal on the right shows the execution of the script. It starts with the command `source venv/bin/activate.fish` and `python p3e2.py`. The user enters the ciphertext `bdvk`. The program then displays all possible plaintext pairs for the given ciphertext:

```
Enter the cipher text : bdkv
All possible plaintext pairs for the given ciphertext :
For key - 0 : bdkv
For key - 1 : acuj
For key - 2 : zbti
For key - 3 : yash
For key - 4 : xzng
For key - 5 : wyqf
For key - 6 : vxpe
For key - 7 : uwod
For key - 8 : tvnc
For key - 9 : sumb
For key - 10 : rtla
For key - 11 : qskz
For key - 12 : prjy
For key - 13 : oqix
For key - 14 : nphw
For key - 15 : mogv
For key - 16 : lnfu
For key - 17 : kmet
For key - 18 : jlds
For key - 19 : ikcr
For key - 20 : hjbq
For key - 21 : giap
For key - 22 : fhzo
For key - 23 : egyn
For key - 24 : dfxm
For key - 25 : cewl
```

Name - Yash Lakhtariya  
Enrollment number - 21162101012  
Branch - CBA      Batch - 61  
INS Practical 3

### 3) Provide Security Mono-alphabetic Cipher Algorithm

Code :

```
import YSL_io
import random
import os

def load_key(key_filename):
    if os.path.exists(key_filename):
        with open(key_filename, "r") as file:
            key = file.read().strip().split()
            YSL_io.printORNG("\n\tKey loaded from file:", end=" ")
            print(" ".join(key))
            return key
    return None

def generate_key(alphabet, key_filename):
    random.shuffle(alphabet)
    with open(key_filename, "w") as file:
        file.write(" ".join(alphabet))
    YSL_io.printORNG("\n\tKey generated and saved to file:", end=" ")
    print(" ".join(alphabet))
    return alphabet

def encrypt(plaintext, key):
    encrypted_text = ""
    for char in plaintext:
        if char.isalpha():
            index = ord(char.upper()) - ord("A")
```

Name - Yash Lakhtariya  
Enrollment number - 21162101012  
Branch - CBA      Batch - 61  
INS Practical 3

```
        encrypted_char = key[index]
        if char.islower():
            encrypted_char = encrypted_char.lower()
        encrypted_text += encrypted_char
    else:
        encrypted_text += char
    return encrypted_text

def decrypt(ciphertext, key):
    decrypted_text = ""
    for char in ciphertext:
        if char.isalpha():
            index = key.index(char.upper())
            decrypted_char = chr(index + ord("A"))
            if char.islower():
                decrypted_char = decrypted_char.lower()
            decrypted_text += decrypted_char
        else:
            decrypted_text += char
    return decrypted_text

def save_ciphertext(ciphertext, ciphertext_filename):
    with open(ciphertext_filename, "w") as file:
        file.write(ciphertext)
    YSL_io.printRED("\n\tEncrypted text saved to file:", end=" ")
    print(ciphertext_filename)

def main():
    alphabet = list("ABCDEFGHIJKLMNOPQRSTUVWXYZ")
    key_filename = "monoalphabetic_key.txt"
```

Name - Yash Lakhtariya  
Enrollment number - 21162101012  
Branch - CBA      Batch - 61  
INS Practical 3

```
ciphertext_filename = "encrypted_text.txt"

operation = YSL_io.inputGRN(
    "\n\tDo you want to encrypt or decrypt? Enter 'e' or 'd' : "
)

if operation.lower() == "e":
    plaintext = YSL_io.inputRED("\n\tEnter the string to encrypt : ")

    generate_key_choice = YSL_io.inputBLU(
        "\n\tDo you want to generate a key? (y/n) : "
    )
    if generate_key_choice.lower() == "y":
        key = generate_key(alphabet.copy(), key_filename)
    else:
        key = load_key(key_filename)

    if key:
        encrypted_text = encrypt(plaintext, key)
        YSL_io.printMGNTA("\n\tEncrypted string:", end=" ")
        print(encrypted_text)

        save_ciphertext(encrypted_text, ciphertext_filename)

elif operation.lower() == "d":
    ciphertext = YSL_io.inputRED("\n\tEnter the string to decrypt : ")
    key_choice = YSL_io.inputBLU("\n\tDo you want to use an existing
key? (y/n) : ")

    if key_choice.lower() == "y":
        key = load_key(key_filename)
    else:
        key = YSL_io.inputRED(
```

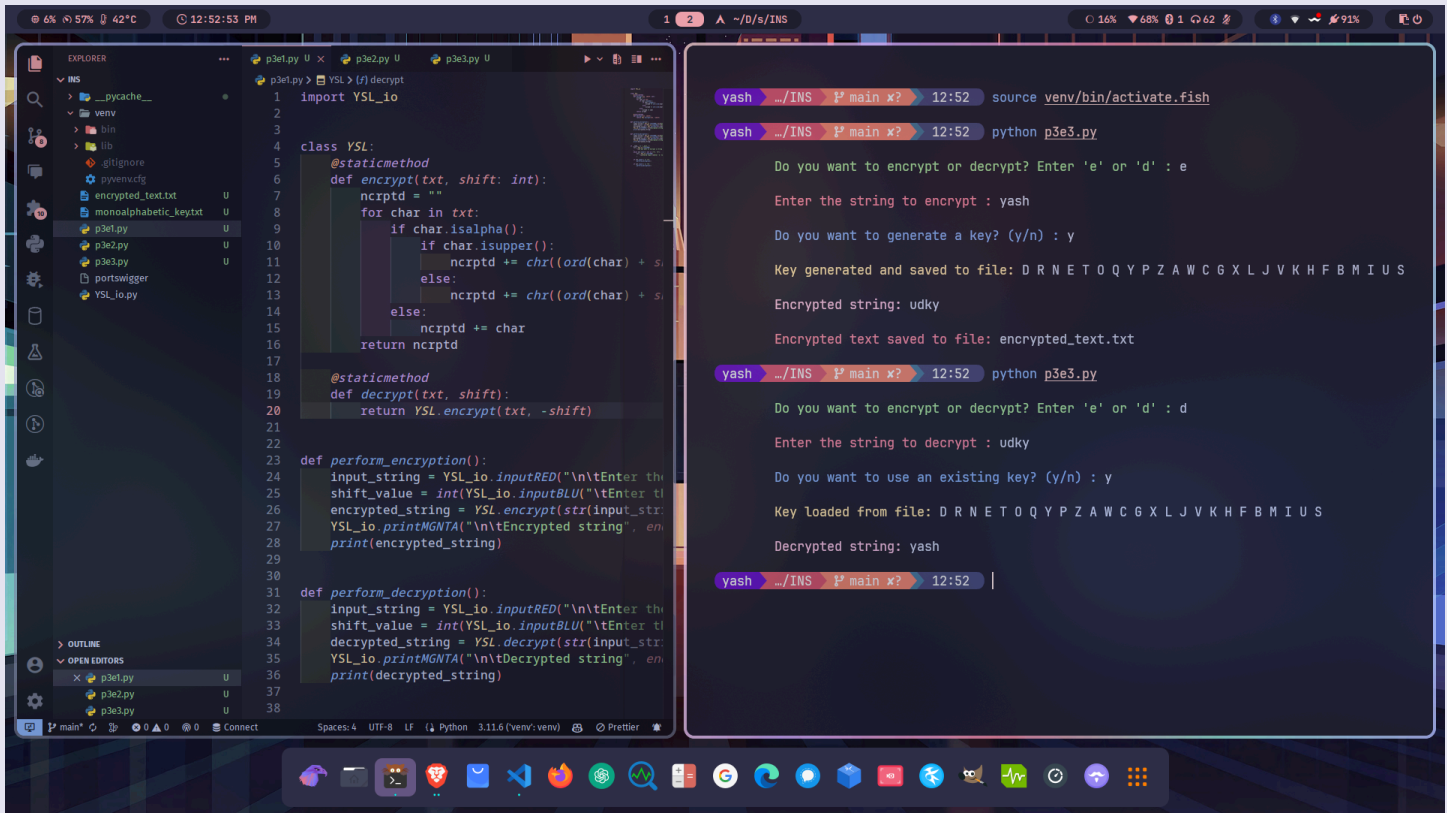


Name - Yash Lakhtariya  
Enrollment number - 21162101012  
Branch - CBA      Batch - 61  
INS Practical 3

```
        "\n\tEnter the key (space-separated characters) : "  
    ).split()  
  
    if key:  
        decrypted_text = decrypt(ciphertext, key)  
        YSL_io.printMGNTA("\n\tDecrypted string:", end=" ")  
        print(decrypted_text)  
  
    else:  
        YSL_io.printRED(  
            "\n\tInvalid operation. Please enter 'e' for encryption or 'd'  
for decryption."  
        )  
  
if __name__ == "__main__":  
    main()
```

Name - Yash Lakhtariya  
Enrollment number - 21162101012  
Branch - CBA Batch - 61  
INS Practical 3

Output :



The image shows a VS Code editor with a Python script named `p3e1.py` and its execution output in a terminal.

**Python Script (`p3e1.py`):**

```
1 import YSL_io
2
3
4 class YSL:
5     @staticmethod
6     def encrypt(txt, shift: int):
7         ncrypted = ""
8         for char in txt:
9             if char.isalpha():
10                 if char.isupper():
11                     ncrypted += chr((ord(char) + shift) % 26)
12                 else:
13                     ncrypted += chr((ord(char) + shift) % 26)
14             else:
15                 ncrypted += char
16         return ncrypted
17
18     @staticmethod
19     def decrypt(txt, shift):
20         return YSL.encrypt(txt, -shift)
21
22
23 def perform_encryption():
24     input_string = YSL_io.inputRED("\nEnter the string to encrypt: ")
25     shift_value = int(YSL_io.inputBLUE("\nEnter the shift value: "))
26     encrypted_string = YSL.encrypt(str(input_string), shift_value)
27     YSL_io.printMGNTA("\nEncrypted string: ")
28     print(encrypted_string)
29
30
31 def perform_decryption():
32     input_string = YSL_io.inputRED("\nEnter the string to decrypt: ")
33     shift_value = int(YSL_io.inputBLUE("\nEnter the shift value: "))
34     decrypted_string = YSL.decrypt(str(input_string), shift_value)
35     YSL_io.printMGNTA("\nDecrypted string: ")
36     print(decrypted_string)
37
38
```

**Terminal Output:**

```
yash ~/INS P main x? 12:52 source venv/bin/activate.fish
yash ~/INS P main x? 12:52 python p3e1.py
Do you want to encrypt or decrypt? Enter 'e' or 'd' : e
Enter the string to encrypt : yash
Do you want to generate a key? (y/n) : y
Key generated and saved to file: DRNETOQYPZAWCGXLJVKHFBMIUS
Encrypted string: udky
Encrypted text saved to file: encrypted_text.txt
yash ~/INS P main x? 12:52 python p3e1.py
Do you want to encrypt or decrypt? Enter 'e' or 'd' : d
Enter the string to decrypt : udky
Do you want to use an existing key? (y/n) : y
Key loaded from file: DRNETOQYPZAWCGXLJVKHFBMIUS
Decrypted string: yash
yash ~/INS P main x? 12:52
```