Pandit Deendayal Energy University

School of Technology

Information Security Lab

B.Tech-Computer Science & Engineering (Sem-V)

PATEL VEDANT H.

19BCP138

DIVISION – 2

Lab 6 Assignment

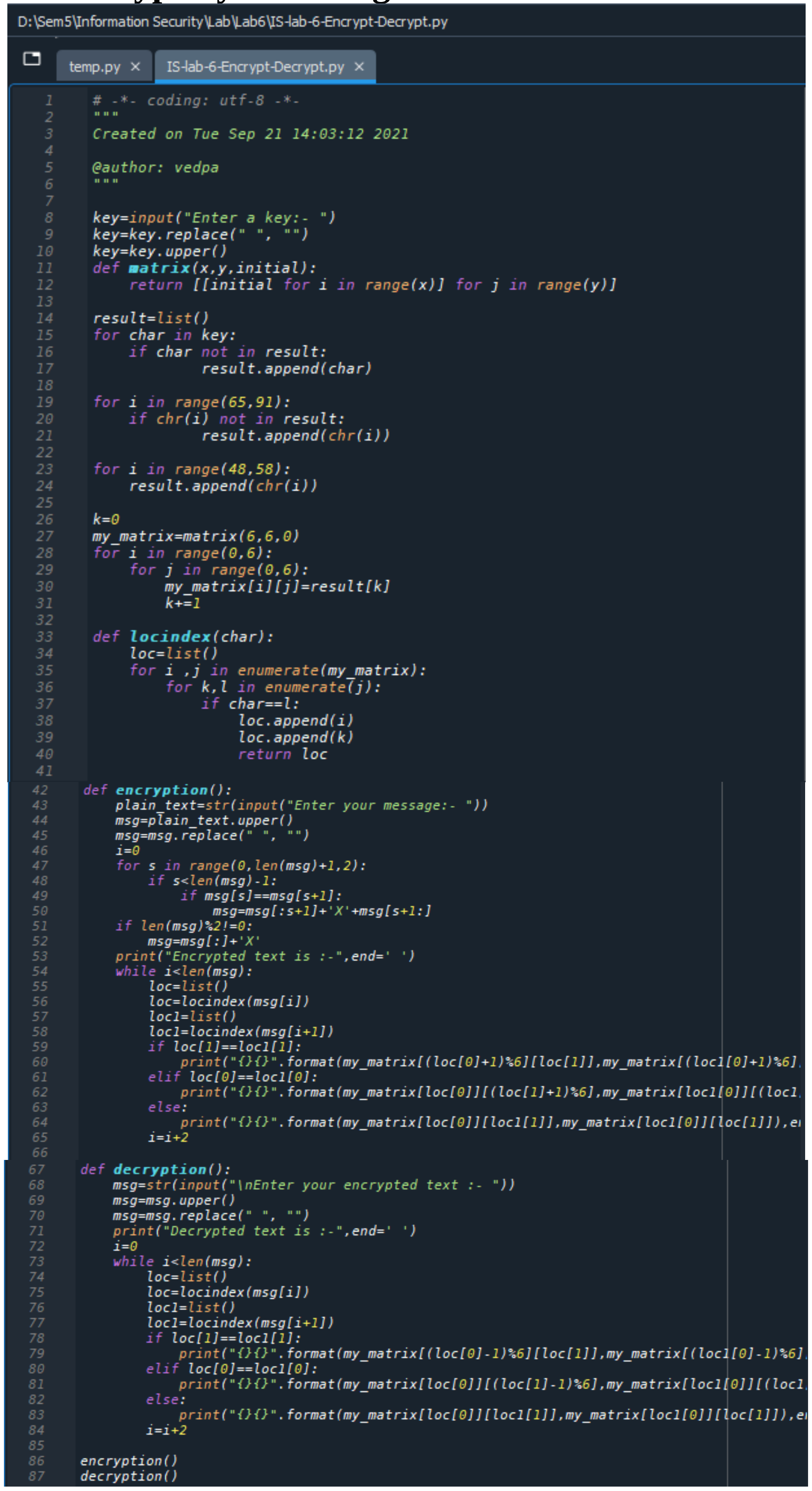
- **Aim:** Study and Implement program for 6X6 Playfair Cipher.
- **Introduction:**

The Playfair cipher is a classic form of polygraphic substitution cipher. It was the first practical polygraph substitution cipher in use. It was invented in 1854 by Charles Wheatstone, but named after lord Playfair after he promoted it heavily. Instead of encrypting single letters, the Playfair cipher encrypts pairs of letters (bigrams). The grid is formed by first taking a code word (with duplicate letters removed) and then adding any alphabet characters missing. A digraph is transformed by looking up the two characters in the grid. If they form a rectangle, pick letters from the same rows but other corners. If they form a column, pick the letters one row down. If they form a row, pick the letters one step to the right. If they are same letter, add a padding letter (for instance X) or pick the letters on row down and one step to the right.

The Playfair is harder to break since the frequency analysis used for simple substitution ciphers does not work with it. The frequency analysis of bigrams is possible, but considerably more difficult. With 600 possible bigrams rather than the 26 possible monograms (single symbols, usually letters in this context), a considerably larger cipher text is required in order to be useful.

❖ Program:

➢ Encrypt and Decrypt Python Program:

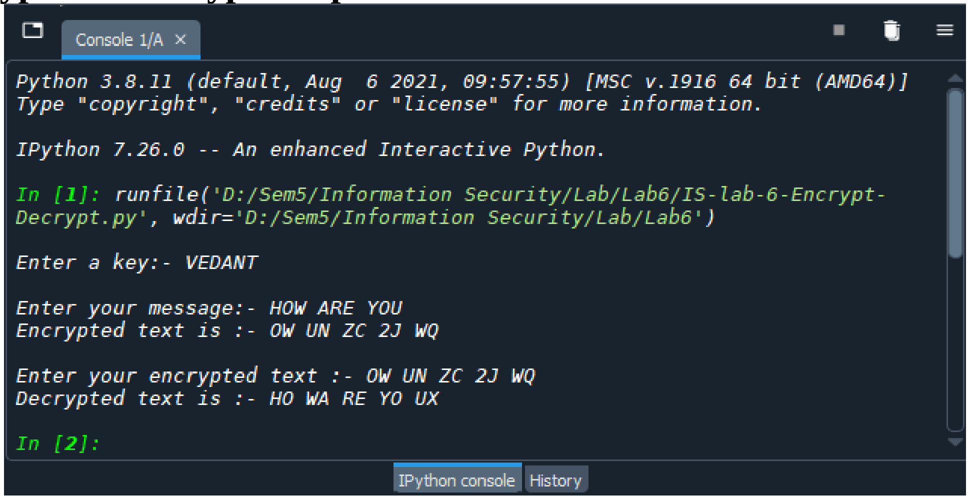


D:\Sem5\Information Security\Lab\Lab6\IS-lab-6-Encrypt-Decrypt.py

temp.py × IS-lab-6-Encrypt-Decrypt.py ×

```python
# -*- coding: utf-8 -*-
"""
Created on Tue Sep 21 14:03:12 2021

@author: vedpa
"""

key=input("Enter a key:- ")
key=key.replace(" ", "")
key=key.upper()
def matrix(x,y,initial):
    return [[initial for i in range(x)] for j in range(y)]

result=list()
for char in key:
    if char not in result:
            result.append(char)

for i in range(65,91):
    if chr(i) not in result:
            result.append(chr(i))

for i in range(48,58):
    result.append(chr(i))

k=0
my_matrix=matrix(6,6,0)
for i in range(0,6):
    for j in range(0,6):
        my_matrix[i][j]=result[k]
        k+=1

def locindex(char):
    loc=list()
    for i ,j in enumerate(my_matrix):
        for k,l in enumerate(j):
            if char==l:
                loc.append(i)
                loc.append(k)
                return loc

def encryption():
    plain_text=str(input("Enter your message:- "))
    msg=plain_text.upper()
    msg=msg.replace(" ", "")
    i=0
    for s in range(0,len(msg)+1,2):
        if s<len(msg)-1:
            if msg[s]==msg[s+1]:
                msg=msg[:s+1]+'X'+msg[s+1:]
    if len(msg)%2!=0:
        msg=msg[:]+'X'
    print("Encrypted text is :-",end=' ')
    while i<len(msg):
        loc=list()
        loc=locindex(msg[i])
        loc1=list()
        loc1=locindex(msg[i+1])
        if loc[1]==loc1[1]:
            print("{}{}".format(my_matrix[(loc[0]+1)%6][loc[1]],my_matrix[(loc1[0]+1)%6]
        elif loc[0]==loc1[0]:
            print("{}{}".format(my_matrix[loc[0]][(loc[1]+1)%6],my_matrix[loc1[0]][(loc1
        else:
            print("{}{}".format(my_matrix[loc[0]][loc1[1]],my_matrix[loc1[0]][loc[1]]),e
        i=i+2

def decryption():
    msg=str(input("\nEnter your encrypted text :- "))
    msg=msg.upper()
    msg=msg.replace(" ", "")
    print("Decrypted text is :-",end=' ')
    i=0
    while i<len(msg):
        loc=list()
        loc=locindex(msg[i])
        loc1=list()
        loc1=locindex(msg[i+1])
        if loc[1]==loc1[1]:
            print("{}{}".format(my_matrix[(loc[0]-1)%6][loc[1]],my_matrix[(loc1[0]-1)%6]
        elif loc[0]==loc1[0]:
            print("{}{}".format(my_matrix[loc[0]][(loc[1]-1)%6],my_matrix[loc1[0]][(loc1
        else:
            print("{}{}".format(my_matrix[loc[0]][loc1[1]],my_matrix[loc1[0]][loc[1]]),e

encryption()
decryption()
```

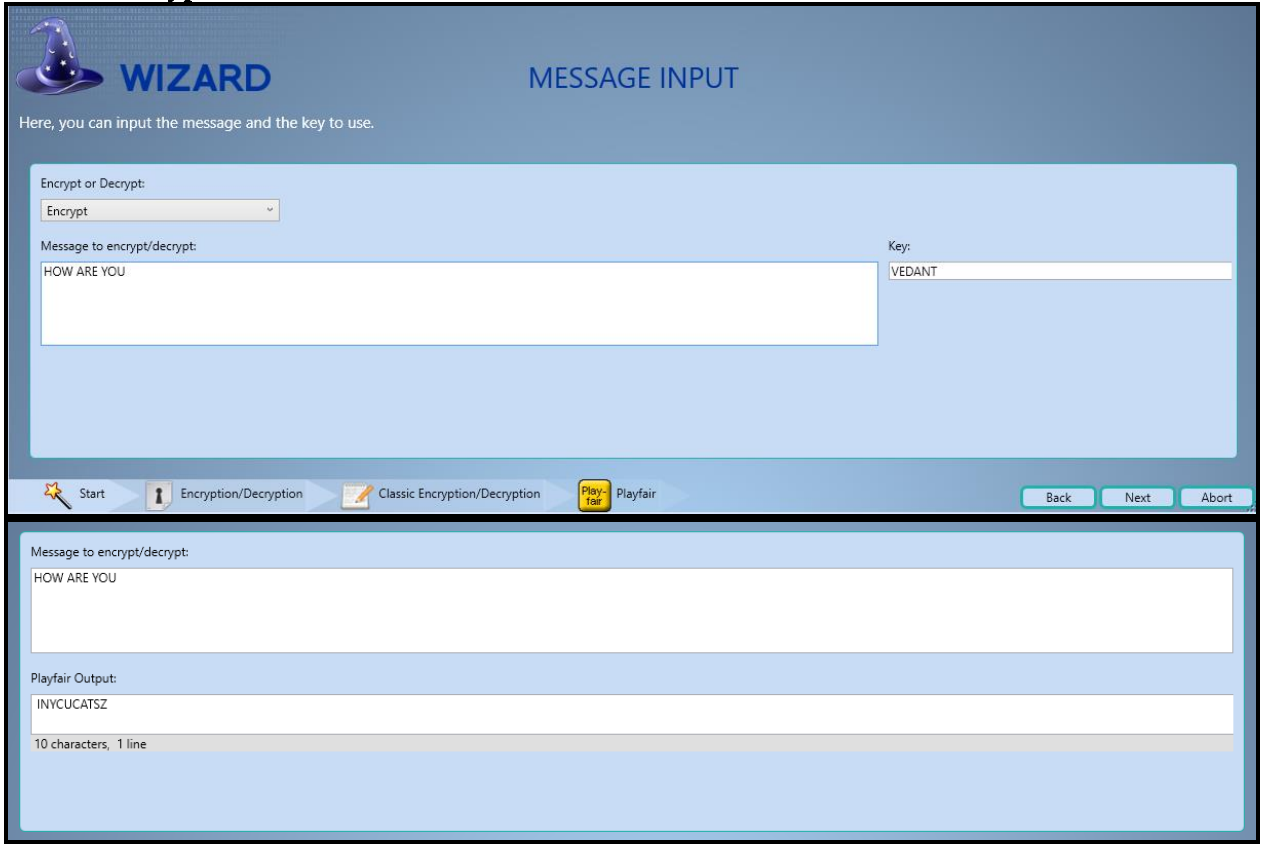
➢ Encrypt and Decrypt Output:



```
Python 3.8.11 (default, Aug  6 2021, 09:57:55) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.26.0 -- An enhanced Interactive Python.

In [1]: runfile('D:/Sem5/Information Security/Lab/Lab6/IS-lab-6-Encrypt-
Decrypt.py', wdir='D:/Sem5/Information Security/Lab/Lab6')

Enter a key:- VEDANT

Enter your message:- HOW ARE YOU
Encrypted text is :- OW UN ZC 2J WQ

Enter your encrypted text :- OW UN ZC 2J WQ
Decrypted text is :- HO WA RE YO UX

In [2]:
```

➢ CrypTool Online Encrypt and Decrypt Output:

- **Encryption:**



Here, you can input the message and the key to use.

Encrypt or Decrypt: Encrypt

Message to encrypt/decrypt: HOW ARE YOU

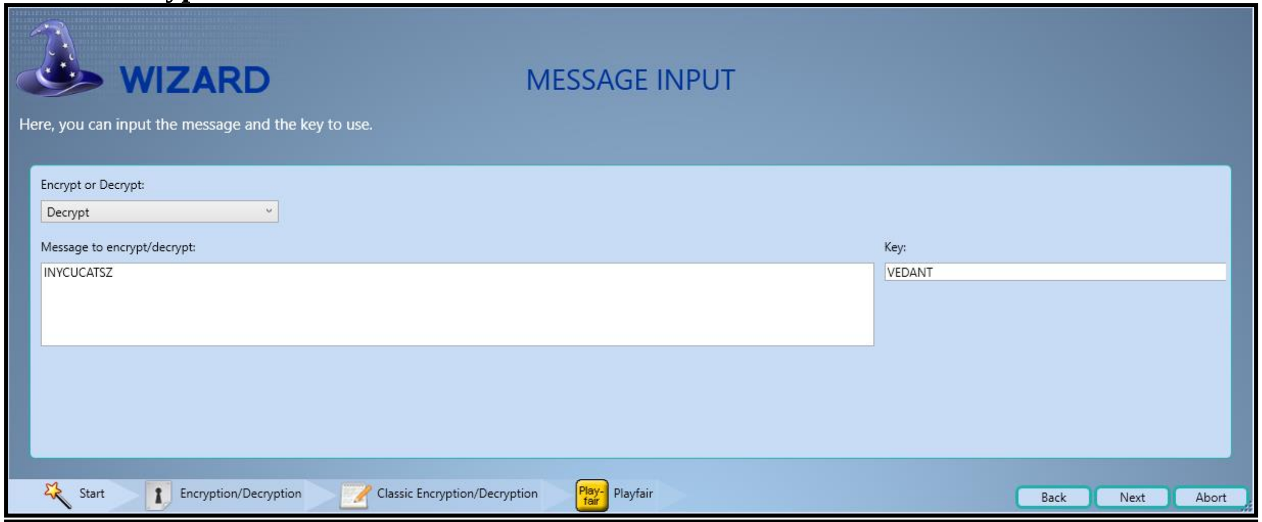
Key: VEDANT

Message to encrypt/decrypt: HOW ARE YOU

Playfair Output: INYCUCATSZ

10 characters, 1 line

- **Decryption:**

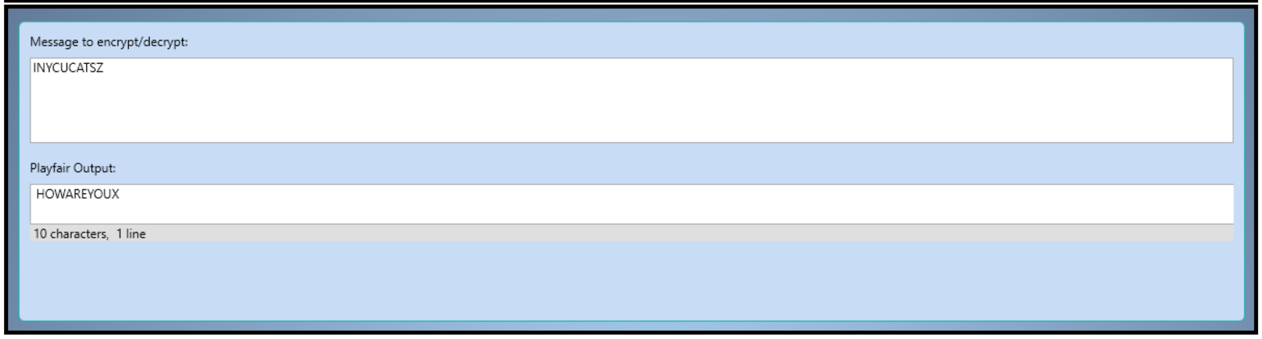


Here, you can input the message and the key to use.

Encrypt or Decrypt: Decrypt

Message to encrypt/decrypt: INYCUCATSZ

Key: VEDANT



Message to encrypt/decrypt: INYCUCATSZ

Playfair Output: HOWAREYOUX

10 characters, 1 line

❖ Cryptanalysis:

The playfair cipher is more complicated than a substitution cipher, but still easy to crack using automated approaches as it is hard for solving the cipher using pen and paper methods.

❖ Applications:

The Playfair cipher was used during World War I, but is no longer used by military forces since it can easily be broken by modern computers. Playfair ciphers, and variants of it, are occasionally used in CTFs, geocaching mystery caches, and logic puzzles. Brute force is used, but domain knowledge is usually applied to make cryptanalysis easier. Using Evolutionary Algorithm (used to solve difficult problems with less information than other methods).

❖ Reference:

➢ https://www.boxentriq.com/code-breaking/playfair-cipher
➢ https://rosettacode.org/wiki/Playfair_cipher
➢ https://en.wikipedia.org/wiki/Playfair_cipher