|  |
| --- |
|  |
| Assignment 1 |
| Network security (UCS727) |

|  |
| --- |
|  |

*Name:*  Sachleen Singh Chani

*Roll No.:* 101506143

*Date:* April 2021

## Q1. Write a program on python to implement Shift Cipher (Caesar Cipher). Take the key and plain text as input from the user.

### Answer:

**Code** –

# Shift cipher | Caesar cipher

# User input plain text

MAX\_KEY\_SIZE **=** 26

# taking input plain text

**def** getText**():**

**print(**"\nEnter the plain text:"**)**

**return** **input()**

# taking input key

**def** getKey**():**

**print(**"\nEnter the Key (1-%s):" **%(**MAX\_KEY\_SIZE**))**

key **=** **int(input())**

#check if key in range

**if** **(**key **>=** 1 **and** key **<=** **(**MAX\_KEY\_SIZE**)):**

**return** key

**else:**

**print(**"Enter between 1-%s" **%(**MAX\_KEY\_SIZE**))**

**def** encText**(**text**,**k**):**

# function to encrypt the plain text

# plain text and the key as arguments

# empty string to add a character with each loop

cipher **=** ""

**for** char **in** text**:**

# looping through the length of the plain text taking each letter at a time

**if(**char **==** " "**):**

cipher **+=** " "

**elif(**char**.**isupper**()):**

# for upper case letters

cipher **+=** **chr(** **(((ord(**char**)** **-** 65**)** **+** k **)** **%** 26 **)** **+** 65 **)**

# shifting the letter by adding the key to the ASCII

# subtracting 65 to keep letters in the range of 0 -25

# mod26 to keep the sum in the A-Z ASCII

**elif(**char**.**islower**()):**

# for lower case letters

cipher **+=** **chr(** **(((ord(**char**)** **-** 97**)** **+** k **)** **%** 26 **)** **+** 97 **)**

**return** cipher

**def** decText**(**text**):**

# function to decrypt the cipher text

# brute forcing the key

**for** dkey **in** **range(**MAX\_KEY\_SIZE**):**

dcipher **=** ""

**for** char **in** text**:**

**if(**char **==** " "**):**

dcipher **+=** " "

**elif(**char**.**isupper**()):**

dcipher **+=** **chr((((ord(**char**)** **-** 65**)** **-** dkey**)** **%** 26 **)** **+** 65 **)**

**elif(**char**.**islower**()):**

dcipher **+=** **chr((((ord(**char**)** **-** 97**)** **-** dkey**)** **%** 26 **)** **+** 97 **)**

**print(**"Key#"**+str(**dkey**)+**" "**,**dcipher**)**

# checking to see if the decrypted text is the same as the plain text given

**if** **(**dcipher **==** plain**):**

**return** **(**dcipher**,** dkey**)**

# user input for plain text and the key | without spaces

plain **=** getText**()**

key **=** getKey**()**

# calling the encryption function

cipher **=** encText**(**plain**,**key**)**

**print(**"\nCipher text is:"**)**

**print(**cipher**,**"\n"**)**

# calling the decryption function

**(**dcipher**,** dkey**)** **=** decText**(**cipher**)**

**print(**"\nThe Original Plain text is:"**)**

**print(**dcipher**)**

**print(**"\nThe key used is:"**)**

**print(**dkey**)**

**Result –**

The Screenshot in fig. 1 shows the result for the above code. The decipher function, decText(), works on the method of brute force by checking all the possible keys from 0 to 25 and printing the output on the screen for the user to verify.

A quick way to verify the deciphered text which is used is to compare each string with the plain text string and the loop is stopped as soon as the match is found, and the value of the deciphered string and the key found is returned.

The ord() function converts the string into its ASCII value, from which 65 is subtracted (representing the upper case letters) to bring the value between 0 and 25 as the ASCII for upper case ‘A’ is 65. To this the shift cipher is applied by adding the key, and adding back the 65 value to change it back to ASCII.

cipher **+=** **chr(** **(((ord(**char**)** **-** 65**)** **+** k **)** **%** 26 **)** **+** 65 **)**

The modulo function is performed to keep the sum again in the 0 – 25 range. The same calculation is performed for the lower-case alphabets by using 97 instead of 65 as ASCII for lower case ‘a’ is 97.

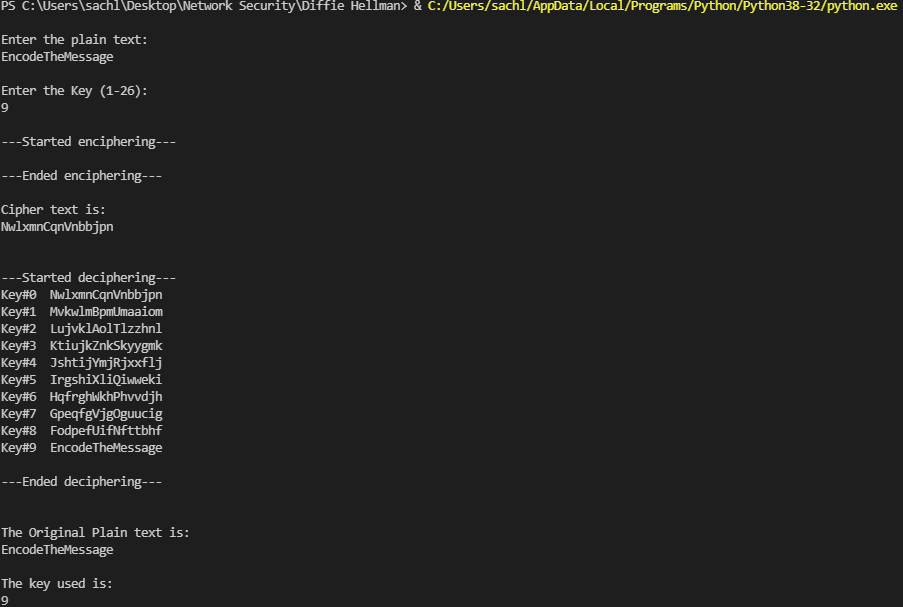


Figure User input Caesar Cipher

## Q2. Implement the Caesar Cipher by reading the plain text from a file.

### Answer:

The key is taken as an input from the user and the plain text is read from a file.

**Code** –

# Shift cipher | Caesar cipher

# plain text read from a file

MAX\_KEY\_SIZE **=** 26

# read in the key

**def** getKey**():**

**print(**"\nEnter the Key (1-%s):" **%(**MAX\_KEY\_SIZE**))**

key **=** **int(input())**

#check if key in range

**if** **(**key **>=** 1 **and** key **<=** **(**MAX\_KEY\_SIZE**)):**

**return** key

**else:**

**print(**"Enter between 1-%s" **%(**MAX\_KEY\_SIZE**))**

**def** encText**(**text**,**k**):**

# function to encrypt the plain text

# plain text and the key as arguments

# empty string to add a character with each loop

cipher **=** ""

**for** char **in** text**:**

# looping through the length of the plain text taking each letter at a time

**if(**char **==** " "**):**

cipher **+=** " "

**elif(**char**.**isupper**()):**

# for upper case letters

cipher **+=** **chr(** **(((ord(**char**)** **-** 65**)** **+** k **)** **%** 26 **)** **+** 65 **)**

# shifting the letter by adding the key to the ASCII

# subtracting 65 to keep letters in the range of 0 -25

# mod26 to keep the sum in the A-Z ASCII

**elif(**char**.**islower**()):**

# for lower case letters

cipher **+=** **chr(** **(((ord(**char**)** **-** 97**)** **+** k **)** **%** 26 **)** **+** 97 **)**

**return** cipher

**def** decText**(**text**):**

# function to decrypt the cipher text

# brute forcing the key

**for** dkey **in** **range(**MAX\_KEY\_SIZE**):**

dcipher **=** ""

**for** char **in** text**:**

**if(**char **==** " "**):**

dcipher **+=** " "

**elif(**char**.**isupper**()):**

dcipher **+=** **chr((((ord(**char**)** **-** 65**)** **-** dkey**)** **%** 26 **)** **+** 65 **)**

**elif(**char**.**islower**()):**

dcipher **+=** **chr((((ord(**char**)** **-** 97**)** **-** dkey**)** **%** 26 **)** **+** 97 **)**

**print(**"Key#"**+str(**dkey**)+**" "**,**dcipher**)**

# checking to see if the decrypted text is the same as the plain text given

**if** **(**dcipher **==** plain**):**

**return** **(**dcipher**,** dkey**)**

# open a file in read mode with the plain text

**while** **True:**

# repeat until the try statement succeeds

**try:**

file **=** **open(**"C:\\Users\\sachl\\Desktop\\plaintext\_assignment1.ssc"**,** "r"**)**

**break**

# exit the loop

**except** **IOError:**

**input(**"Could not open file!"**)**

# restart the loop

**print(**"\nPlain text read from a file:"**)**

# plain text read from a file

plain **=** file**.**read**()**

**print(**plain**)**

# user input for the key

key **=** getKey**()**

# calling the encryption function

cipher **=** encText**(**plain**,**key**)**

**print(**"\nCipher text is:"**)**

**print(**cipher**,**"\n"**)**

# calling the decryption function

**(**dcipher**,** dkey**)** **=** decText**(**cipher**)**

**print(**"\nThe Original Plain text is:"**)**

**print(**dcipher**)**

**print(**"\nThe key used is:"**)**

**print(**dkey**)**

#closed the file

file**.**close**()**

**Result –**

The Code is all similar to the previous question apart from removing the user input for the plain text, instead it is read from a text file. The result for the above code could be seen in the screenshot in fig. 2.

While opening the file, a check has been done to see if the file opened successfully. The text file contains a single line of text, “Caesar Cipher”, which is taken as the input plain text.

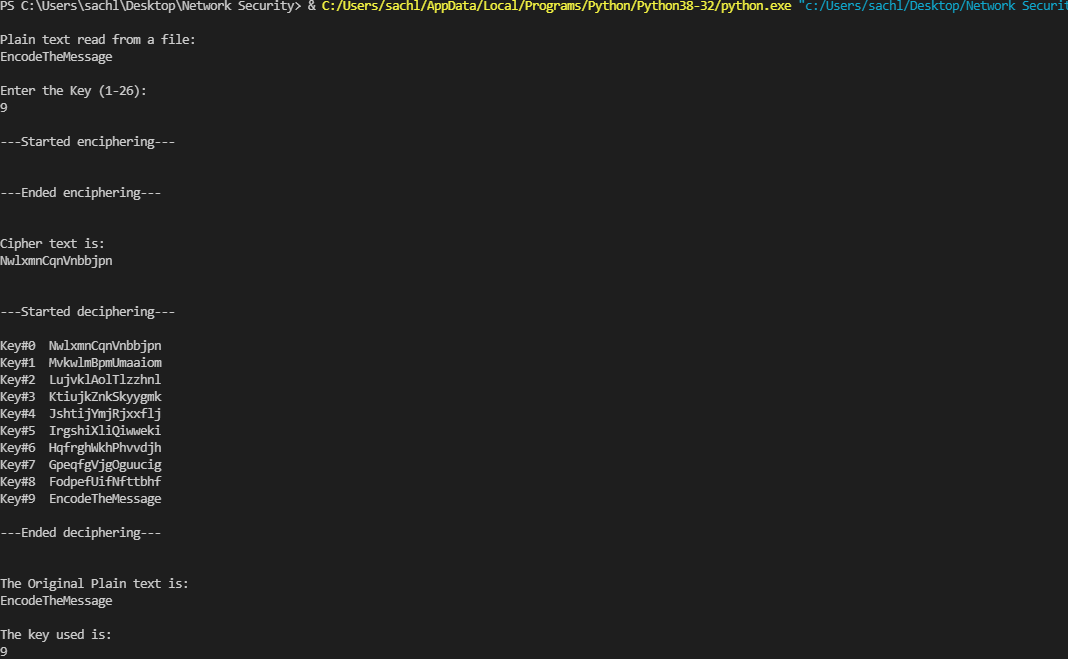


Figure Plain text read from a file for Caesar Cipher