Network Security

Project 0: Encryption Decryption using mono-alphabetic substitution

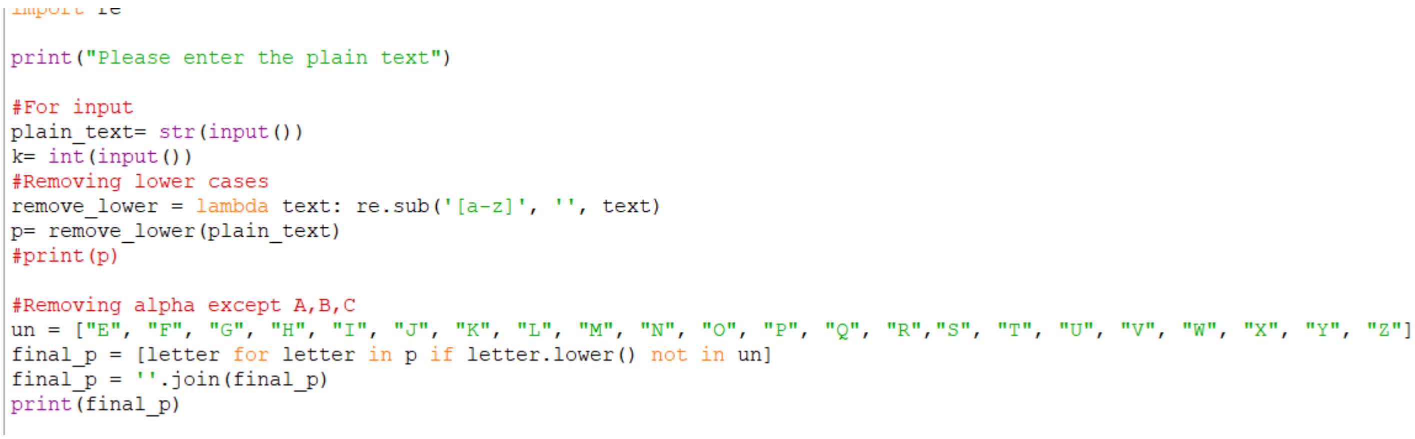
Parth Singh 2018356 Udbhav Gupta 2017319

Explaining Approach:

We are given that we have to do encryption and decryption using **monoalphabetic substitution** cipher which relies on a fixed replacement structure.

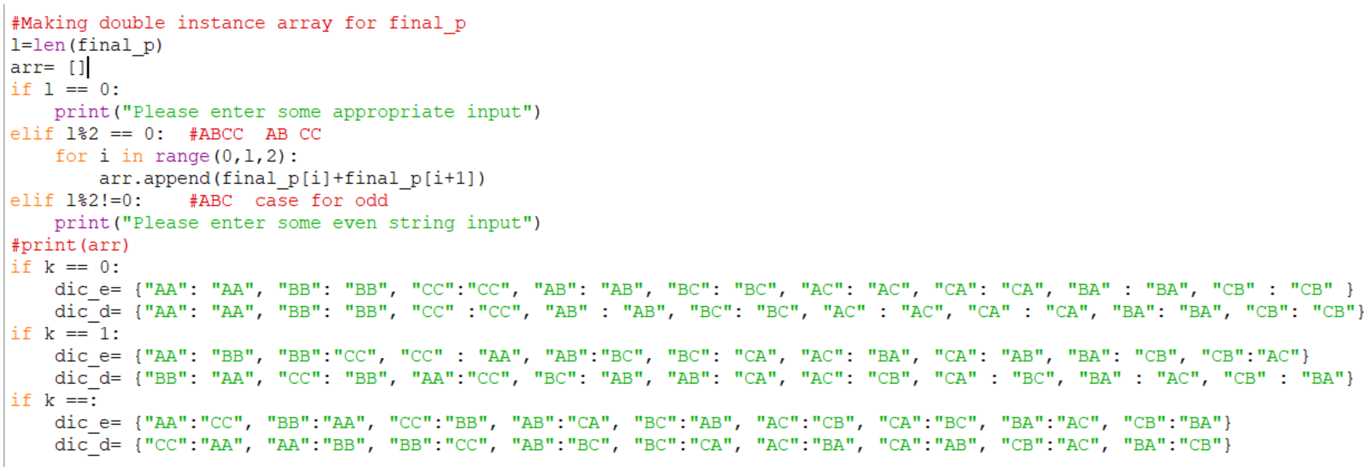
So, we took the input plain_text such that there it filters out:

1. All lower cases alphabets
2. All uppercase alphabets except A,B,C



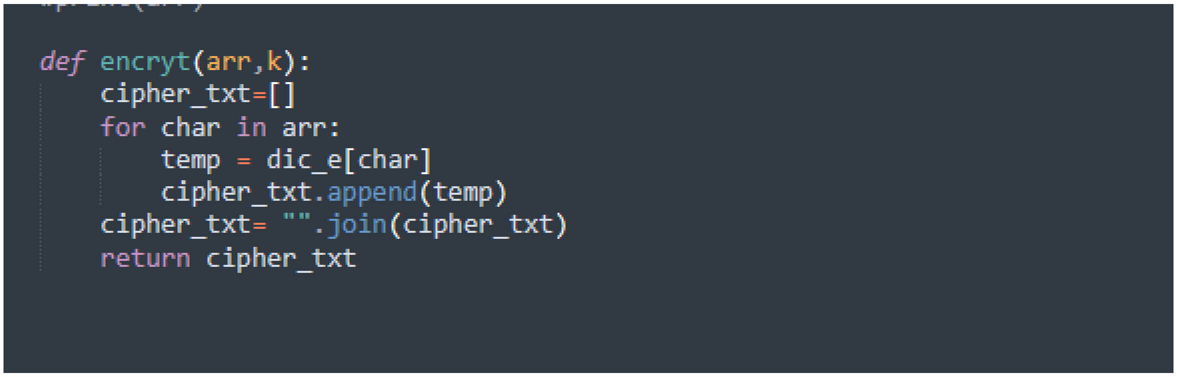
```
import re

print("Please enter the plain text")

#For input
plain_text= str(input())
k= int(input())
#Removing lower cases
remove_lower = lambda text: re.sub('[a-z]', '', text)
p= remove_lower(plain_text)
#print(p)

#Removing alpha except A,B,C
un = ["E", "F", "G", "H", "I", "J", "K", "L", "M", "N", "O", "P", "Q", "R","S", "T", "U", "V", "W", "X", "Y", "Z"]
final_p = [letter for letter in p if letter.lower() not in un]
final_p = ''.join(final_p)
print(final_p)
```

After this we started with dividing plane text into pairs of two. Also, making a dictionary to allot the elements like AB→ BC for key 1. But for Key 0, we will have AB→AB.



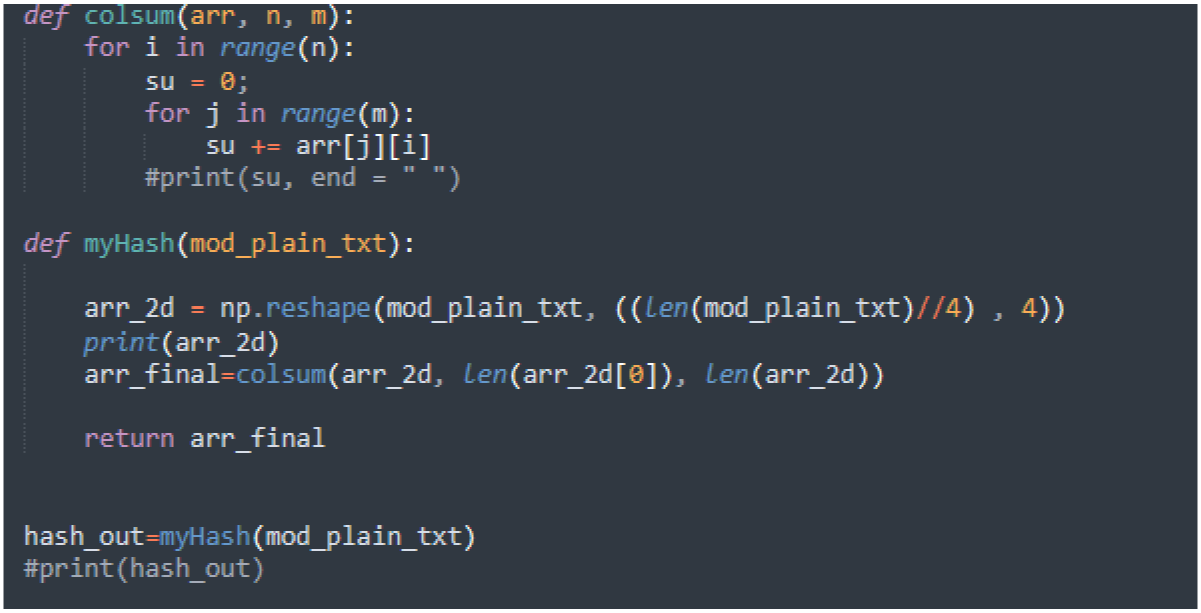
```
#Making double instance array for final_p
l=len(final_p)
arr= []
if l == 0:
    print("Please enter some appropriate input")
elif l%2 == 0:  #ABCC  AB CC
    for i in range(0,l,2):
        arr.append(final_p[i]+final_p[i+1])
elif l%2!=0:    #ABC  case for odd
    print("Please enter some even string input")
#print(arr)
if k == 0:
    dic_e= {"AA": "AA", "BB": "BB", "CC":"CC", "AB": "AB", "BC": "BC", "AC": "AC", "CA": "CA", "BA" : "BA", "CB" : "CB" }
    dic_d= {"AA": "AA", "BB": "BB", "CC" :"CC", "AB" : "AB", "BC": "BC", "AC" : "AC", "CA" : "CA", "BA": "BA", "CB": "CB"}
if k == 1:
    dic_e= {"AA": "BB", "BB":"CC", "CC" : "AA", "AB":"BC", "BC": "CA", "AC": "BA", "CA": "AB", "BA": "CB", "CB":"AC"}
    dic_d= {"BB": "AA", "CC": "BB", "AA":"CC", "BC": "AB", "AB": "CA", "AC": "CB", "CA" : "BC", "BA" : "AC", "CB" : "BA"}
if k ==:
    dic_e= {"AA":"CC", "BB":"AA", "CC":"BB", "AB":"CA", "BC":"AB", "AC":"CB", "CA":"BC", "BA":"AC", "CB":"BA"}
    dic_d= {"CC":"AA", "AA":"BB", "BB":"CC", "AB":"BC", "BC":"CA", "AC":"BA", "CA":"AB", "CB":"AC", "BA":"CB"}
```

Later starting the encryption process. Here we took the parameters arr of the split string and then processing it with the dictionary and then replacing the pairs with the pairs which corresponds their key associated pairs.



```
#print(arr)

def encryt(arr,k):
    cipher_txt=[]
    for char in arr:
        temp = dic_e[char]
        cipher_txt.append(temp)
    cipher_txt= "".join(cipher_txt)
    return cipher_txt
```

Next process is to do hashing. Where we are converting the plain text into the corresponding to ascii value. Input → Matrix(2D), in which the row size is l(len(plain text)) and the column is 4.



```
def colsum(arr, n, m):
    for i in range(n):
        su = 0;
        for j in range(m):
            su += arr[j][i]
        #print(su, end = " ")

def myHash(mod_plain_txt):

    arr_2d = np.reshape(mod_plain_txt, ((len(mod_plain_txt)//4) , 4))
    print(arr_2d)
    arr_final=colsum(arr_2d, len(arr_2d[0]), len(arr_2d))

    return arr_final


hash_out=myHash(mod_plain_txt)
#print(hash_out)
```

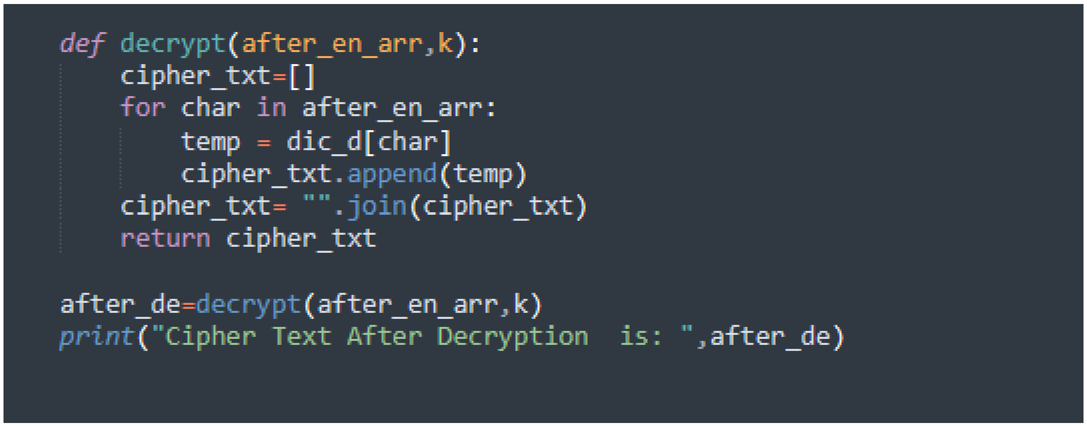
The value of the column of the hash array corresponds to the xor of all the elements of the column present at that column number.

Then converting the integer to the corresponding character where 0 represents A, 1→ B, 2→ C.

Here, we performed decryption.

// To be done → Concatenation of the hash function plus encrypted cipher text.

The encrypted text is being passed to the decrypted function from which the value of the pairs has been substituted via the decryption table to find the desired message.

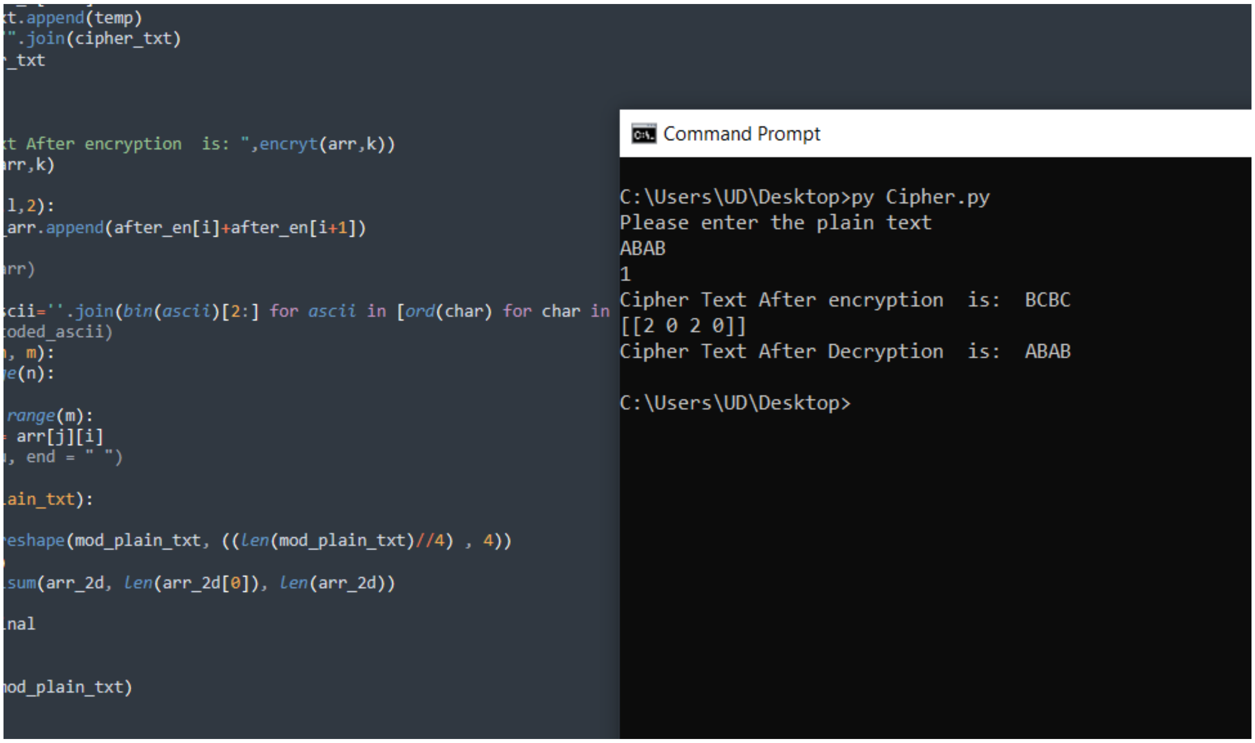


```
def decrypt(after_en_arr,k):
    cipher_txt=[]
    for char in after_en_arr:
        temp = dic_d[char]
        cipher_txt.append(temp)
    cipher_txt= "".join(cipher_txt)
    return cipher_txt

after_de=decrypt(after_en_arr,k)
print("Cipher Text After Decryption  is: ",after_de)
```

Example of working:

Input: ABAB Output: Encrypted: BCBC

Decrypted: ABAB



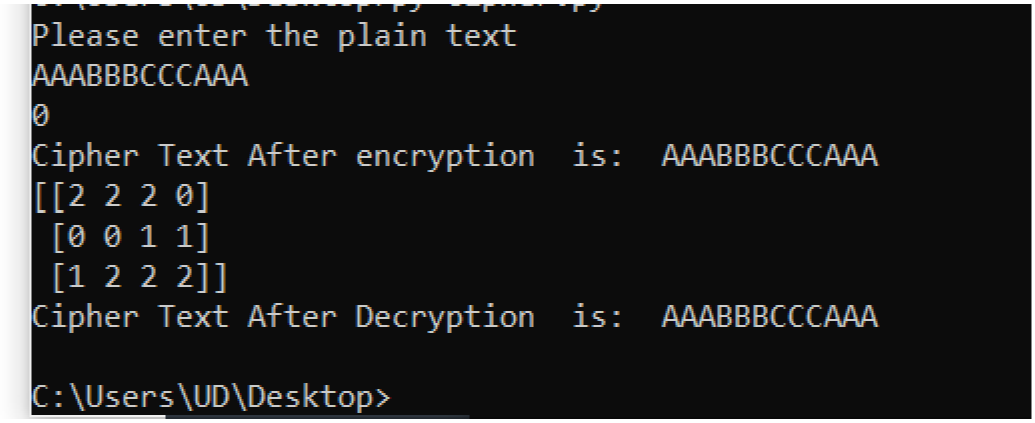
```
t.append(temp)
".join(cipher_txt)
_txt



t After encryption  is: ",encryt(arr,k))
rr,k)

1,2):
arr.append(after_en[i]+after_en[i+1])

rr)

cii=''.join(bin(ascii)[2:] for ascii in [ord(char) for char in
oded_ascii)
, m):
e(n):

range(m):
 arr[j][i]
, end = " ")

ain_txt):

eshape(mod_plain_txt, ((len(mod_plain_txt)//4) , 4))

sum(arr_2d, len(arr_2d[0]), len(arr_2d))

nal


od_plain_txt)
```

```
C:\Users\UD\Desktop>py Cipher.py
Please enter the plain text
ABAB
1
Cipher Text After encryption  is:  BCBC
[[2 0 2 0]]
Cipher Text After Decryption  is:  ABAB

C:\Users\UD\Desktop>
```

Another Example:

Input: AAABBBCCCAAA Output: Encrypted: AAABBBCCCAAA

Decrypted: AAABBBCCCAAA

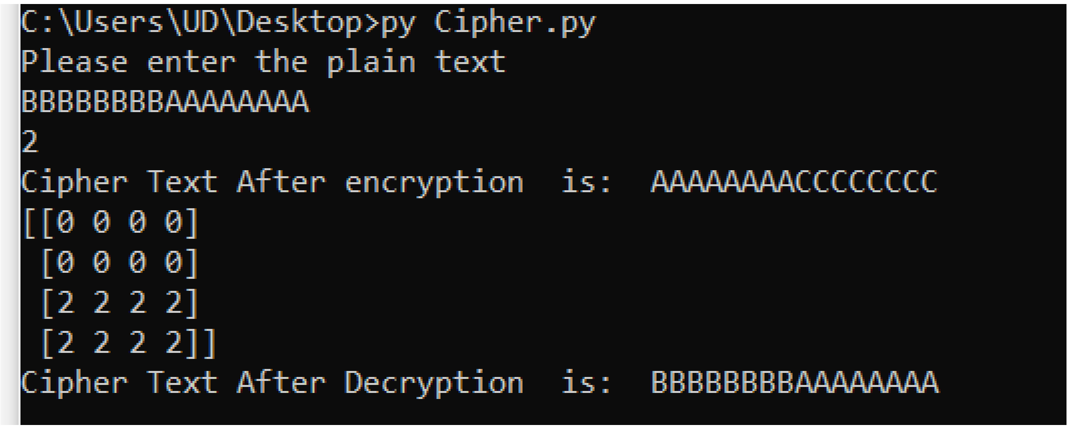


```
Please enter the plain text
AAABBBCCCAAA
0
Cipher Text After encryption  is:   AAABBBCCCAAA
[[2 2 2 0]
 [0 0 1 1]
 [1 2 2 2]]
Cipher Text After Decryption  is:   AAABBBCCCAAA

C:\Users\UD\Desktop>
```

Example:

Input: BBBBBBBBAAAAAAAA Output: AAAAAAAACCCCCCCC

Decrypted: BBBBBBBBAAAAAAAA



```
C:\Users\UD\Desktop>py Cipher.py
Please enter the plain text
BBBBBBBBAAAAAAAA
2
Cipher Text After encryption  is:   AAAAAAAACCCCCCCC
[[0 0 0 0]
 [0 0 0 0]
 [2 2 2 2]
 [2 2 2 2]]
Cipher Text After Decryption  is:   BBBBBBBBAAAAAAAA
```