

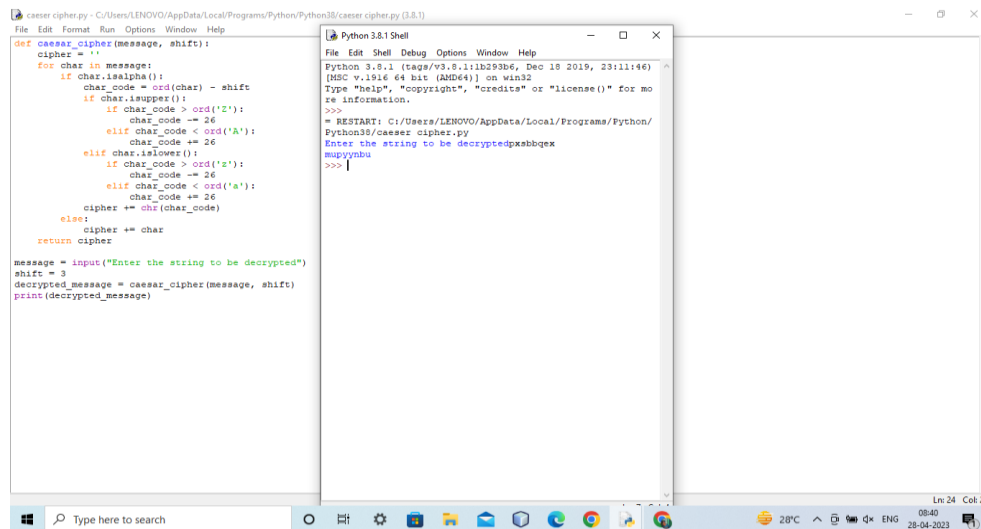
Day 1

Q1. Write a python program for Caesar cipher involves replacing each letter of the alphabet with a letter standing k places further down the alphabet for K in the range 1 through 25

PROGRAM:

```
def caesar_cipher(message, shift):  
    cipher = ""  
    for char in message:  
        if char.isalpha():  
            char_code = ord(char) - shift  
            if char.isupper():  
                if char_code > ord('Z'):  
                    char_code -= 26  
                elif char_code < ord('A'):  
                    char_code += 26  
            elif char.islower():  
                if char_code > ord('z'):  
                    char_code -= 26  
                elif char_code < ord('a'):  
                    char_code += 26  
            cipher += chr(char_code)  
        else:  
            cipher += char  
    return cipher  
message = input("Enter the string to be decrypted")  
shift = 3  
decrypted_message = caesar_cipher(message, shift)  
print(decrypted_message)
```

RESULT:



The screenshot shows a Python IDE with two windows. The left window, titled 'caesar_cipher.py', contains the following code:

```
def caesar_cipher(message, shift):
    cipher = ''
    for char in message:
        if char.isalpha():
            char_code = ord(char) - shift
            if char.isupper():
                if char_code > ord('Z'):
                    char_code -= 26
            elif char_code < ord('A'):
                char_code += 26
            elif char.islower():
                if char_code > ord('z'):
                    char_code -= 26
            elif char_code < ord('a'):
                char_code += 26
            cipher += chr(char_code)
        else:
            cipher += char
    return cipher

message = input("Enter the string to be decrypted")
shift = 3
decrypted_message = caesar_cipher(message, shift)
print(decrypted_message)
```

The right window, titled 'Python 3.8.1 Shell', shows the execution of the program. It displays the Python version, system information, and the user's input and output:

```
>>>
= RESTART: C:/Users/LENOVO/AppData/Local/Programs/Python/Python38/caesar_cipher.py
Enter the string to be decrypted:pxabbqex
mupymbu
>>>
```

Q2. Write a python program for monoalphabetic substitution cipher maps a plaintext alphabet to a ciphertext alphabet so that each letter of the plaintext alphabet maps to a single unique letter of the ciphertext alphabet.

PROGRAM:

```
import string
```

```
cipher_map = {'a': 'q', 'b': 'w', 'c': 'e', 'd': 'r', 'e': 't',
```

```
              'f': 'y', 'g': 'u', 'h': 'i', 'i': 'o', 'j': 'p',
```

```
              'k': 'a', 'l': 's', 'm': 'd', 'n': 'f', 'o': 'g',
```

```
              'p': 'h', 'q': 'j', 'r': 'k', 's': 'l', 't': 'z',
```

```
              'u': 'x', 'v': 'c', 'w': 'v', 'x': 'b', 'y': 'n', 'z': 'm'}
```

```
decipher_map = {v: k for k, v in cipher_map.items()}
```

```
def encrypt(message):
```

```
    """Encrypts the given message using the cipher map."""
```

```
    message = message.lower()
```

```
    encrypted_message = "
```

```
    # Encrypt each character in the message
```

```
    for char in message:
```

```

if char in string.ascii_lowercase:

    encrypted_char = cipher_map[char]

else:

    encrypted_char = char

encrypted_message += encrypted_char

return encrypted_message

message = input("Enter the text:")

encrypted_message = encrypt(message)

print(encrypted_message)

```

RESULT:

The screenshot shows a Windows desktop with two windows. The left window is a text editor titled 'mono cipher.py' containing a Python script for a Caesar cipher. The script defines a cipher map, a reverse map for decryption, an encrypt function, and a main execution block. The right window is a 'Python 3.8.1 Shell' showing the execution of the script. The user enters 'saveetha' at the prompt, and the output is 'lqetttzlg'.

```

mono cipher.py - C:/Users/LENOVO/AppData/Local/Programs/Python/Python38/mono cipher.py (3.8.1)
File Edit Format Run Options Window Help

import string

# Define the mapping for the cipher
cipher_map = {'a': 'q', 'b': 'w', 'c': 'e', 'd': 't', 'e': 'b',
              'f': 'y', 'g': 'u', 'h': 'i', 'i': 'o', 'j': 'p',
              'k': 's', 'l': 'g', 'm': 'd', 'n': 'r', 'o': 'g',
              'p': 'h', 'q': 'j', 'r': 'k', 's': 'l', 't': 'z',
              'u': 'x', 'v': 'c', 'w': 'v', 'x': 'b', 'y': 'n', 'z': 'm'}

# Define the reverse mapping for decryption
decipher_map = {v: k for k, v in cipher_map.items()}

def encrypt(message):
    """Encrypts the given message using the cipher map."""
    # Convert message to lowercase
    message = message.lower()
    # Initialize the encrypted message
    encrypted_message = ''
    # Encrypt each character in the message
    for char in message:
        if char in string.ascii_lowercase:
            encrypted_char = cipher_map[char]
        else:
            encrypted_char = char
        encrypted_message += encrypted_char
    return encrypted_message

message = input("Enter the text:")
encrypted_message = encrypt(message)
print(encrypted_message)

```

```

Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 23:11:46) [MSC v.1
516 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more inform
ation.
>>>
= RESTART: C:/Users/LENOVO/AppData/Local/Programs/Python/Python38/
mono cipher.py
Enter the text:saveetha
lqetttzlg
>>>

```

Q3. Write a Python program for the Playfair algorithm is based on the use of a 5 X 5 matrix of letters constructed on a keyword. Plaintext has encrypted two letters at a time using this matrix.

PROGRAM:

```

def toLowerCase(text):

    return text.lower()

def removeSpaces(text):

    newText = ""

```

```

    for i in text:
        if i == " ":
            continue

        else:
            newText = newText + i

    return newText

def Diagraph(text):
    Diagraph = []
    group = 0
    for i in range(2, len(text), 2):
        Diagraph.append(text[group:i])
        group = i
    Diagraph.append(text[group:])
    return Diagraph

def FillerLetter(text):
    k = len(text)
    if k % 2 == 0:
        for i in range(0, k, 2):
            if text[i] == text[i+1]:
                new_word = text[0:i+1] + str('x') + text[i+1:]
                new_word = FillerLetter(new_word)
                break
            else:
                new_word = text
    else:
        for i in range(0, k-1, 2):
            if text[i] == text[i+1]:
                new_word = text[0:i+1] + str('x') + text[i+1:]

```

```

        new_word = FillerLetter(new_word)

        break

    else:

        new_word = text

    return new_word

list1 = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'k', 'l', 'm',

        'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']

def generateKeyTable(word, list1):

    key_letters = []

    for i in word:

        if i not in key_letters:

            key_letters.append(i)

    compElements = []

    for i in key_letters:

        if i not in compElements:

            compElements.append(i)

    for i in list1:

        if i not in compElements:

            compElements.append(i)

    matrix = []

    while compElements != []:

        matrix.append(compElements[:5])

        compElements = compElements[5:]

    return matrix

def search(mat, element):

    for i in range(5):

        for j in range(5):

```

```

        if(mat[i][j] == element):

            return i, j

def encrypt_RowRule(matr, e1r, e1c, e2r, e2c):

    char1 = ""

    if e1c == 4:

        char1 = matr[e1r][0]

    else:

        char1 = matr[e1r][e1c+1]

    char2 = ""

    if e2c == 4:

        char2 = matr[e2r][0]

    else:

        char2 = matr[e2r][e2c+1]

    return char1, char2

def encrypt_ColumnRule(matr, e1r, e1c, e2r, e2c):

    char1 = ""

    if e1r == 4:

        char1 = matr[0][e1c]

    else:

        char1 = matr[e1r+1][e1c]

    char2 = ""

    if e2r == 4:

        char2 = matr[0][e2c]

    else:

        char2 = matr[e2r+1][e2c]

    return char1, char2

def encrypt_RectangleRule(matr, e1r, e1c, e2r, e2c):

    char1 = ""

```

```

        char1 = matr[e1r][e2c]

        char2 = ""

        char2 = matr[e2r][e1c]

        return char1, char2

def encryptByPlayfairCipher(Matrix, plainList):

    CipherText = []

    for i in range(0, len(plainList)):

        c1 = 0

        c2 = 0

        ele1_x, ele1_y = search(Matrix, plainList[i][0])

        ele2_x, ele2_y = search(Matrix, plainList[i][1])

        if ele1_x == ele2_x:

            c1, c2 = encrypt_RowRule(Matrix, ele1_x, ele1_y, ele2_x, ele2_y)

        elif ele1_y == ele2_y:

            c1, c2 = encrypt_ColumnRule(Matrix, ele1_x, ele1_y, ele2_x, ele2_y)

        else:

            c1, c2 = encrypt_RectangleRule(

                Matrix, ele1_x, ele1_y, ele2_x, ele2_y)

        cipher = c1 + c2

        CipherText.append(cipher)

    return CipherText

text_Plain = 'hello'

text_Plain = removeSpaces(toLowerCase(text_Plain))

PlainTextList = Diagraph(FillerLetter(text_Plain))

if len(PlainTextList[-1]) != 2:

    PlainTextList[-1] = PlainTextList[-1]+'z'

key = "network"

print("Key text:", key)

```

```

key = toLowerCase(key)

Matrix = generateKeyTable(key, list1)

print("Plain Text:", text_Plain)

CipherList = encryptByPlayfairCipher(Matrix, PlainTextList)

CipherText = ""

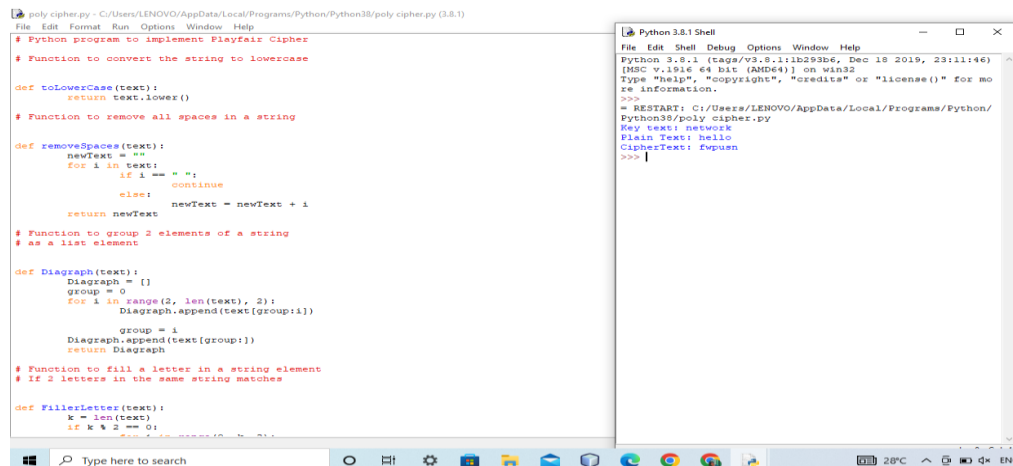
for i in CipherList:

    CipherText += i

print("CipherText:", CipherText)

```

RESULT:



```

poly cipher.py - C:/Users/LENOVO/AppData/Local/Programs/Python/Python38/poly cipher.py (3.8.1)
File Edit Format Run Options Window Help
# Python program to implement Playfair Cipher
# Function to convert the string to lowercase
def toLowerCase(text):
    return text.lower()

# Function to remove all spaces in a string
def removeSpaces(text):
    newText = ""
    for i in text:
        if i == " ":
            continue
        else:
            newText = newText + i
    return newText

# Function to group 2 elements of a string
# as a list element
def Diagraph(text):
    Diagraph = []
    group = 0
    for i in range(2, len(text), 2):
        Diagraph.append(text[group:i])
        group = i
    Diagraph.append(text[group:])
    return Diagraph

# Function to fill a letter in a string element
# If 2 letters in the same string matches
def FillerLetter(text):
    k = len(text)
    if k % 2 == 0:
        return text
    else:
        return text + "X"

key = "KEY"
text = "hello"
key = toLowerCase(key)
key = removeSpaces(key)
key = Diagraph(key)
key = FillerLetter(key)
key = generateKeyTable(key, list1)
text = removeSpaces(text)
text = Diagraph(text)
text = FillerLetter(text)
CipherList = encryptByPlayfairCipher(key, text)
CipherText = ""
for i in CipherList:
    CipherText += i
print("CipherText:", CipherText)

```

Q4. Write a Python program for the polyalphabetic substitution cipher uses a separate monoalphabetic substitution cypher for each successive letter of plaintext, depending on a key

PROGRAM:

```

import string

def poly_sub_cipher(plaintext, key):

    plaintext = plaintext.lower()

    key = key.lower()

    alphabet = string.ascii_lowercase

    ciphertext = ""

```



```

for i, char in enumerate(plaintext):

    key_char = key[i % len(key)]

    shift = alphabet.index(key_char)

    shifted_alphabet = alphabet[shift:] + alphabet[:shift]

    if char in alphabet:

        ciphertext += shifted_alphabet[alphabet.index(char)]

    else:

        ciphertext += char

return ciphertext

plaintext = 'GEEKSFORGEEKS'

key = 'AYUSH'

ciphertext = poly_sub_cipher(plaintext, key)

print(ciphertext)

```

RESULT:

The screenshot displays a Windows desktop environment. On the left, a text editor window titled 'polyalphabetic.py' shows the following Python code:

```

import string

def poly_sub_cipher(plaintext, key):
    plaintext = plaintext.lower()
    key = key.lower()
    alphabet = string.ascii_lowercase
    ciphertext = ''
    for i, char in enumerate(plaintext):
        key_char = key[i % len(key)]
        shift = alphabet.index(key_char)
        shifted_alphabet = alphabet[shift:] + alphabet[:shift]
        if char in alphabet:
            ciphertext += shifted_alphabet[alphabet.index(char)]
        else:
            ciphertext += char
    return ciphertext
plaintext = 'GEEKSFORGEEKS'
key = 'AYUSH'
ciphertext = poly_sub_cipher(plaintext, key)
print(ciphertext)

```

On the right, a 'Python 3.8.1 Shell' window shows the execution of the script. The output is:

```

Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 23:11:46) [MSC v.1916
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more informatio
n.
>>>
= RESTART: C:\Users\LENOVO\AppData\Local\Programs\Python\Python38\poly
alphabetic.py
gcyczfmlyleim
>>>

```

The taskbar at the bottom shows the Windows search bar and several application icons.