

Day-1

1.write a c program for Caesar cipher involves replacing each letter of the alphabet with the 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
```

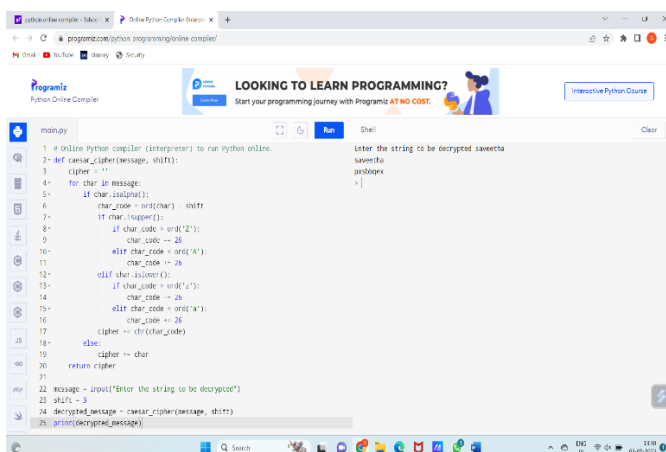
```
message = input("Enter the string to be decrypted")
```

```
shift = 3
```

```
decrypted_message = caesar_cipher(message, shift)
```

```
print(decrypted_message)
```

output:



The screenshot shows a web browser window with the URL `programiz.com/python-programming/online-compiler/`. The page features a banner for 'Programiz' with the text 'LOOKING TO LEARN PROGRAMMING?' and 'Start your programming journey with Programiz AT NO COST.' Below the banner is a code editor with the following Python code:

```
1 # Online Python compiler (interpreter) to run Python online.
2 def caesar_cipher(message, shift):
3     cipher = ""
4     for char in message:
5         if char.isalpha():
6             char_code = ord(char) - shift
7             if char.isupper():
8                 if char_code > ord('Z'):
9                     char_code -= 26
10                elif char_code < ord('A'):
11                    char_code += 26
12            elif char.islower():
13                if char_code > ord('z'):
14                    char_code -= 26
15                elif char_code < ord('a'):
16                    char_code += 26
17            cipher += chr(char_code)
18        else:
19            cipher += char
20    return cipher
21
22 message = input("Enter the string to be decrypted")
23 shift = 3
24 decrypted_message = caesar_cipher(message, shift)
25 print(decrypted_message)
```

On the right side of the code editor, there is a 'Shell' window with the prompt 'Enter the string to be decrypted' and the input 'sawetha'.

2. Write a C program for monoalphabetic substitution cipher maps a plain text alphabet to a cipher text so, that each letter of the plain text alphabets maps to a single unique letter of the cipher text 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 = "
```

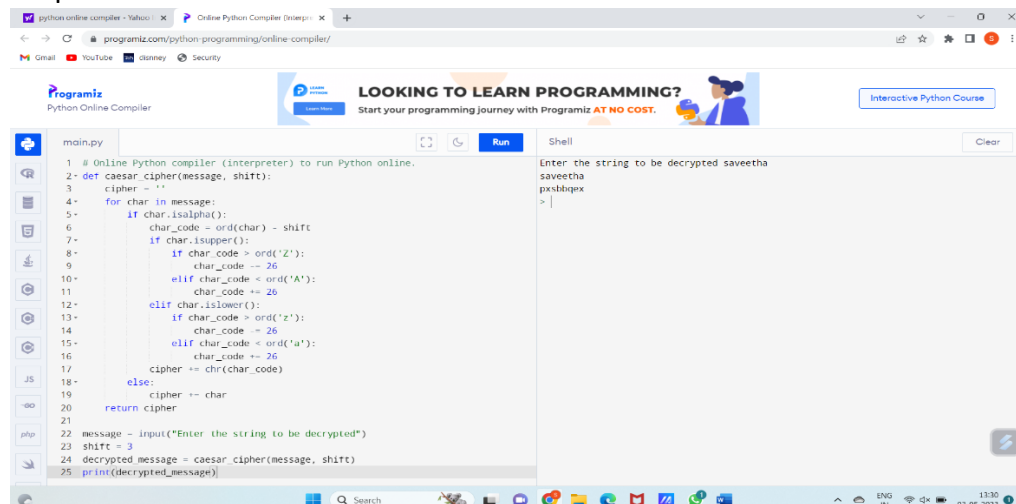
```
    for char in message:  
        if char in string.ascii_lowercase:  
            encrypted_char = cipher_map[char]  
        else:  
            encrypted_char = char  
        encrypted_message += encrypted_char  
    return encrypted_message
```

```
message = input("Enter the text:")
```

```
encrypted_message = encrypt(message)
```

```
print(encrypted_message)
```

output:



The screenshot shows a web browser window with the URL 'programiz.com/python-programming/online-compiler/'. The page features a banner for 'Programiz' with the text 'LOOKING TO LEARN PROGRAMMING? Start your programming journey with Programiz AT NO COST.' and a button for 'Interactive Python Course'. Below the banner is a code editor with a file named 'main.py'. The code defines a Caesar cipher function that shifts characters by a specified amount. The input string is 'saveetha peshiqax' and the shift is 3. The output is 'vayghu tyshludx'. The code is as follows:

```
1 # Online Python compiler (interpreter) to run Python online.  
2 def caesar_cipher(message, shift):  
3     cipher = ''  
4     for char in message:  
5         if char.isalpha():  
6             char_code = ord(char) - shift  
7             if char.isupper():  
8                 if char_code > ord('Z'):  
9                     char_code -= 26  
10                elif char_code < ord('A'):  
11                    char_code += 26  
12            elif char.islower():  
13                if char_code > ord('z'):  
14                    char_code -= 26  
15                elif char_code < ord('a'):  
16                    char_code += 26  
17            cipher += chr(char_code)  
18        else:  
19            cipher += char  
20    return cipher  
21  
22 message = input("Enter the string to be decrypted")  
23 shift = 3  
24 decrypted_message = caesar_cipher(message, shift)  
25 print(decrypted_message)
```

The output of the program is displayed in the shell window:

```
Enter the string to be decrypted saveetha  
peshiqax  
>  
vayghu  
tyshludx
```

3. 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)

            # Get 2 letter cipherText
        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 = 'instruments'
text_Plain = removeSpaces(toLowerCase(text_Plain))
PlainTextList = Diagraph(FillerLetter(text_Plain))
if len(PlainTextList[-1]) != 2:
    PlainTextList[-1] = PlainTextList[-1]+'z'
key = "Monarchy"
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)

```

output:

The screenshot shows the Programiz Python Online Compiler interface. The main editor displays a Python script named 'main.py' with the following code:

```

1 def toLowerCase(text):
2     return text.lower()
3 def removeSpaces(text):
4     newText = ""
5     for i in text:
6         if i == " ":
7             continue
8         else:
9             newText = newText + i
10    return newText
11 def Diagraph(text):
12     Diagraph = []
13     group = 0
14     for i in range(2, len(text), 2):
15         Diagraph.append(text[group:i])
16         group = i
17     Diagraph.append(text[group:])
18     return Diagraph
19 def FillerLetter(text):
20     k = len(text)
21     if k % 2 == 0:
22         for i in range(0, k, 2):
23             if text[i] == text[i+1]:
24                 new_word = text[0:i+1] + str('x') + text[i+1:]
25

```

The output window on the right shows the following results:

```

Key text: Monarchy
Plain Text: instruments
CipherText: gatlmzclrqtx
>

```

4. Write a Python program for the polyalphabetic substitution cypher 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):
```

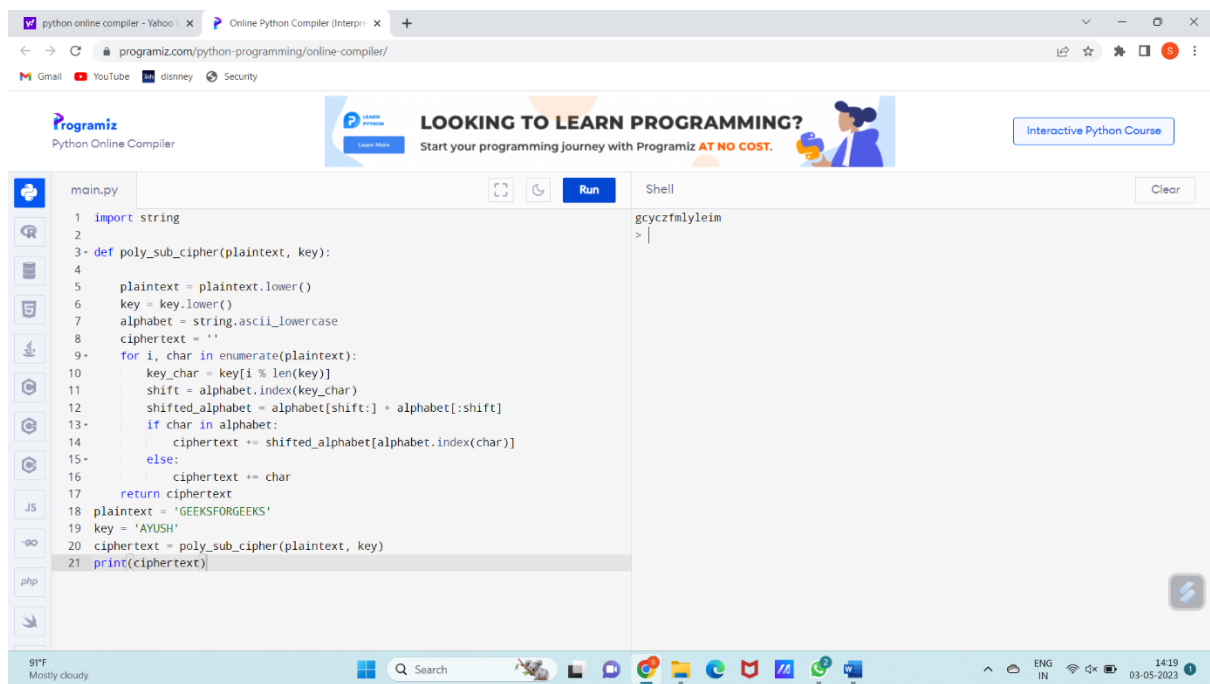
```

    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)
```

out put:



The screenshot displays the Programiz Online Python Compiler interface. The browser address bar shows the URL `programiz.com/python-programming/online-compiler/`. The page features a header with the Programiz logo, a promotional banner for learning programming, and a navigation menu on the left with icons for Python, JavaScript, and PHP. The main workspace is divided into two panels. The left panel, titled 'main.py', contains the following Python code:

```
1 import string
2
3 def poly_sub_cipher(plaintext, key):
4
5     plaintext = plaintext.lower()
6     key = key.lower()
7     alphabet = string.ascii_lowercase
8     ciphertext = ''
9     for i, char in enumerate(plaintext):
10         key_char = key[i % len(key)]
11         shift = alphabet.index(key_char)
12         shifted_alphabet = alphabet[shift:] + alphabet[:shift]
13         if char in alphabet:
14             ciphertext += shifted_alphabet[alphabet.index(char)]
15         else:
16             ciphertext += char
17     return ciphertext
18 plaintext = 'GEEKSFORGEEKS'
19 key = 'AYUSH'
20 ciphertext = poly_sub_cipher(plaintext, key)
21 print(ciphertext)
```

The right panel, titled 'Shell', shows the output of the program:

```
gcyzfmlyleim
> |
```

The bottom of the interface includes a status bar with weather information (91°F, Mostly cloudy), a search bar, and system icons for network, battery, and time (14:19, 03-05-2023).

