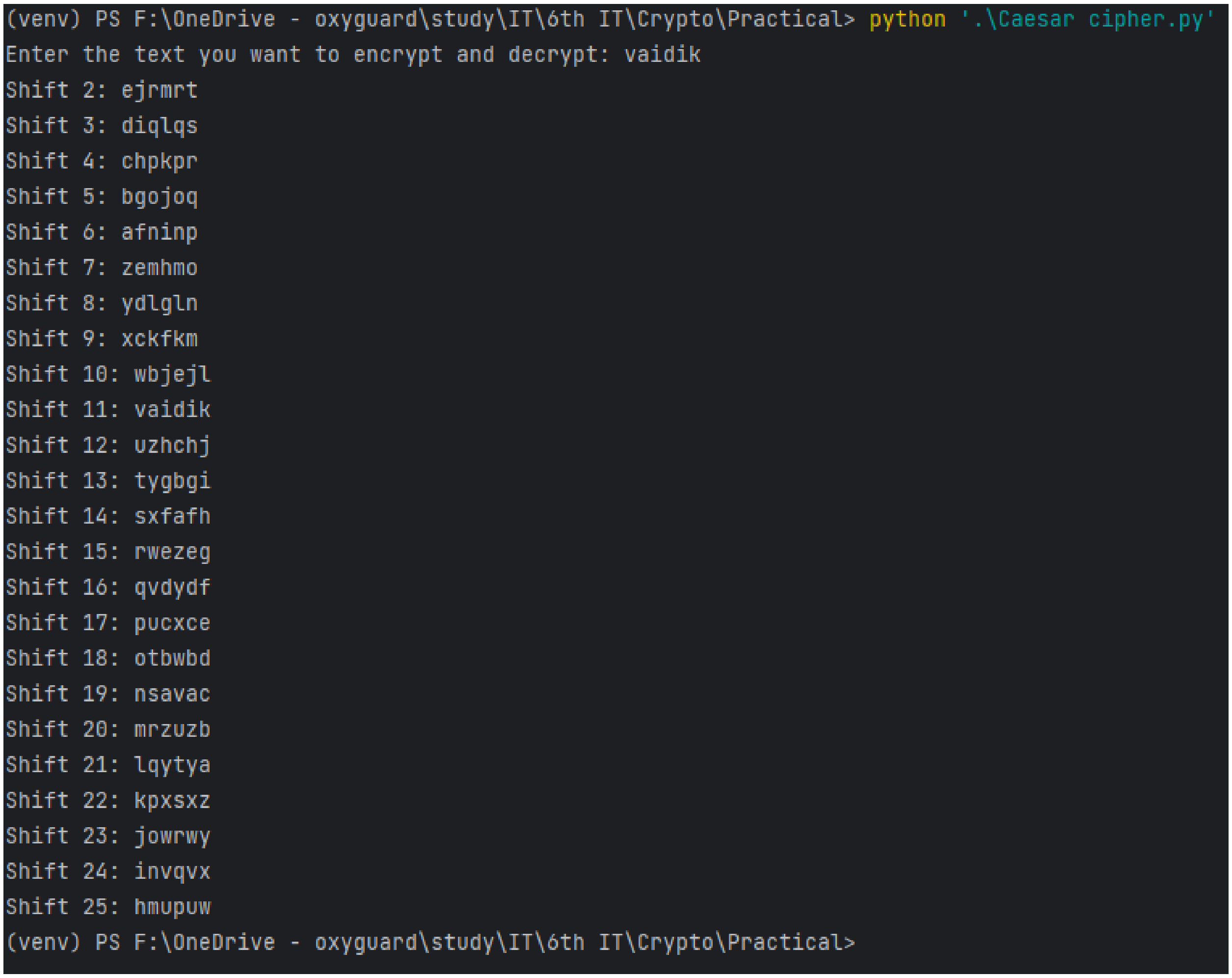
Practical-1

Aim: Perform encryption and decryption using Caesar substitution cipher. Perform Brute Force attack on the ciphertext to retrieve plaintext.

Code:

```
def caesar_encrypt(plaintext, shift):
    ciphertext = ''
    for char in plaintext:
        if char.isalpha():
            # For each character, it checks if
            # it is an alphabet letter (char.isalpha()). If it is, it calculates the shifted
            # character based on whether it's lowercase or uppercase and appends it to the
ciphertext.
            shift_amount = shift % 26
            if char.islower():
                shifted = chr(((ord(char) - ord('a') + shift_amount) % 26) + ord('a'))
            else:
                shifted = chr(((ord(char) - ord('A') + shift_amount) % 26) + ord('A'))
            ciphertext += shifted
        elif char.isdigit():
            shifted = str((int(char) + shift) % 10)
            ciphertext += shifted
        else:
            ciphertext += char
    return ciphertext

def caesar_decrypt(ciphertext, shift):
    plaintext = ''
    for char in ciphertext:
        if char.isalpha():
            shift_amount = shift % 26
            if char.islower():
                shifted = chr(((ord(char) - ord('a') - shift_amount) % 26) + ord('a'))
            else:
                shifted = chr(((ord(char) - ord('A') - shift_amount) % 26) + ord('A'))
            plaintext += shifted
        elif char.isdigit():
            shifted = str((int(char) - shift) % 10)
            plaintext += shifted
        else:
            plaintext += char
    return plaintext

def caesar_brute_force(ciphertext):
    decrypted_texts = []
    for shift in range(26):
        decrypted_text = ''
        for char in ciphertext:
            if char.isalpha():
                if char.islower():
                    decrypted_char = chr(((ord(char) - ord('a') - shift) % 26) + ord('a'))
                else:
                    decrypted_char = chr(((ord(char) - ord('A') - shift) % 26) + ord('A'))
                decrypted_text += decrypted_char
            else:
                decrypted_text += char
        decrypted_texts.append(decrypted_text)
    return decrypted_texts

def get_input():
    plaintext = input("Enter the text you want to encrypt and decrypt: ")
    shift_str = input("Enter the shift (a positive number for encryption, a negative number
for decryption): ")
    if shift_str.isdigit():
        shift = int(shift_str)
    else:
        print("Invalid input for shift. Please enter a valid number.")
        return None, None
    return plaintext, shift


plaintext, shift = get_input()

if plaintext is not None and shift is not None:
    encrypted_text = caesar_encrypt(plaintext, shift)
    print("Encrypted:", encrypted_text)
    decrypted_text = caesar_decrypt(encrypted_text, shift)
    print("Decrypted:", decrypted_text)

    decrypted_texts = caesar_brute_force(encrypted_text)
    for i, text in enumerate(decrypted_texts):
        print(f"Shift {i}: {text}")
```

OutPut:



```
(venv) PS F:\OneDrive - oxyguard\study\IT\6th IT\Crypto\Practical> python '.\Caesar cipher.py'
Enter the text you want to encrypt and decrypt: vaidik
Shift 2: ejrmrt
Shift 3: diqlqs
Shift 4: chpkpr
Shift 5: bgojoq
Shift 6: afninp
Shift 7: zemhmo
Shift 8: ydlgln
Shift 9: xckfkm
Shift 10: wbjejl
Shift 11: vaidik
Shift 12: uzhchj
Shift 13: tygbgi
Shift 14: sxfafh
Shift 15: rwezeg
Shift 16: qvdydf
Shift 17: pucxce
Shift 18: otbwbd
Shift 19: nsavac
Shift 20: mrzuzb
Shift 21: lqytya
Shift 22: kpxsxz
Shift 23: jowrwy
Shift 24: invqvx
Shift 25: hmupuw
(venv) PS F:\OneDrive - oxyguard\study\IT\6th IT\Crypto\Practical>
```

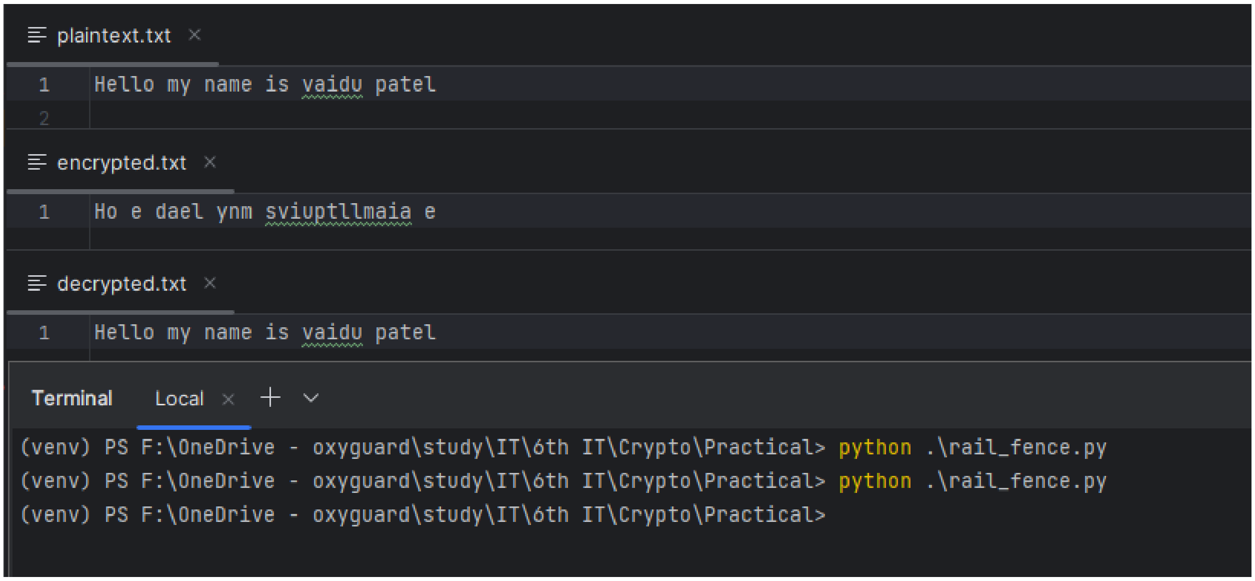
Practical-2

Aim: Perform encryption and decryption using Rail Fence transposition cipher. Perform encryption by fetching data from .txt file and decryption through file operations.

Code:

```
def encryptRailFence(plain_text, key):
    rail = [['\n' for i in range(len(plain_text))] for j in range(key)]
    dir_down = False
    row, col = 0, 0
    for i in range(len(plain_text)):
        if (row == 0) or (row == key - 1):
            dir_down = not dir_down
        rail[row][col] = plain_text[i]
        col += 1
        if dir_down:
            row += 1
        else:
            row -= 1
    result = []
    for i in range(key):
        for j in range(len(plain_text)):
            if rail[i][j] != '\n':
                result.append(rail[i][j])
    return "".join(result)

def decryptRailFence(cipher_text, key):
    rail = [['\n' for i in range(len(cipher_text))] for j in range(key)]
    dir_down = None
    row, col = 0, 0
    for i in range(len(cipher_text)):
        if row == 0:
            dir_down = True
        if row == key - 1:
            dir_down = False
        rail[row][col] = '*'
        col += 1
        if dir_down:
            row += 1
        else:
            row -= 1
    index = 0
    for i in range(key):
        for j in range(len(cipher_text)):
            if (rail[i][j] == '*') and (index < len(cipher_text)):
                rail[i][j] = cipher_text[index]
                index += 1
    result = []
    row, col = 0, 0
    for i in range(len(cipher_text)):
        if row == 0:
            dir_down = True
        if row == key - 1:
            dir_down = False
        if rail[row][col] != '*':
            result.append(rail[row][col])
            col += 1
        if dir_down:
            row += 1
        else:
            row -= 1
    return "".join(result)

key = 3
with open('plaintext.txt', 'r') as f:
    plain_text = f.read().replace('\n', '')
cipher_text = encryptRailFence(plain_text, key)
with open('encrypted.txt', 'w') as f:
    f.write(cipher_text)

with open('encrypted.txt', 'r') as f:
    cipher_text = f.read().replace('\n', '')
plain_text = decryptRailFence(cipher_text, key)
with open('decrypted.txt', 'w') as f:
    f.write(plain_text)
```

OutPut:



```
Hello my name is vaidu patel
```

```
Ho e dael ynm sviuptllmaia e
```

```
Hello my name is vaidu patel
```

```
(venv) PS F:\OneDrive - oxyguard\study\IT\6th IT\Crypto\Practical> python .\rail_fence.py
(venv) PS F:\OneDrive - oxyguard\study\IT\6th IT\Crypto\Practical> python .\rail_fence.py
(venv) PS F:\OneDrive - oxyguard\study\IT\6th IT\Crypto\Practical>
```

Practical-3

Aim: Perform encryption and decryption using Playfair substitution cipher. Perform encryption and decryption for both alphabetic and alphanumeric data types.

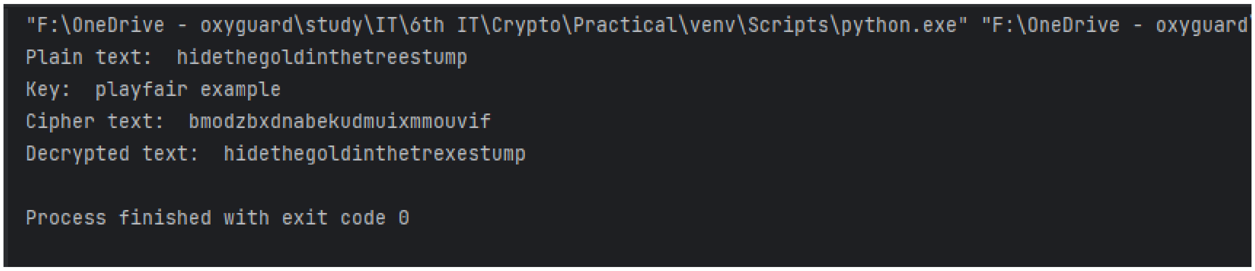
Code:

```
import string
def genKeyMat(key):
    atoz = string.ascii_lowercase.replace('j', '.')
    key_matrix = ['' for i in range(5)]
    i = 0
    j = 0
    for c in key:
        if c in atoz:
            key_matrix[i] += c
            atoz = atoz.replace(c, '.')

            j += 1
            if j > 4:
                i += 1
                j = 0
    for c in atoz:
        if c != '.':
            key_matrix[i] += c

            j += 1
            if j>4:
                i += 1
                j = 0
    return key_matrix



def encrypt(plainText):
    plaintextpairs = []
    ciphetextpairs = []
    # Rule 1: if both latter are same or only one left add "X" after first letter
    i = 0
    while i < len(plainText):
```

```
    a = plainText[i]
    b = ""
    if (i+1) == len(plainText):
      b = 'x'
    else:
      b = plainText[i+1]
    if a != b:
      plaintextpairs.append(a + b)
      i += 2
    else :
      plaintextpairs.append(a + 'x')
      i += 1
  # Rule 2: if letters are  in same row, replace with letters to their immediate right letter
  for pair in plaintextpairs:
    applied_rule = False
    for row in key_matrix:
      if pair[0] in row and pair[1] in row:
        j0 = row.find(pair[0])
        j1 = row.find(pair[1])
        ciphetextpair = row[(j0 + 1) % 5] + row[(j1 + 1) % 5]
        ciphetextpairs.append(ciphetextpair)
        applied_rule = True
    if applied_rule:
      continue
  # Rule 3 :If letter are in same column, replace them with immediate below letter
    for j in range(5):
      col = "".join([key_matrix[i][j] for i in range(5)])
      if pair[0] in col and pair[1] in col:
        i0 = col.find(pair[0])
        i1 = col.find(pair[1])
        ciphetextpair = col[(i0 + 1) % 5] + col[(i1 + 1) % 5]
        ciphetextpairs.append(ciphetextpair)
        applied_rule = True
    if applied_rule:
      continue
  # Rule 4: not in same column or row,replace them with the letters on same row
respectively but at
  # the other pair of corners of the rectangle define by the original pairs
    i0 = 0
    i1 = 0
```

```
        j0 = 0
        j1 = 0
        for i in range(5):
            row = key_matrix[i]
            if pair[0] in row:
                i0 = i
                j0 = row.find(pair[0])
            if pair[1] in row:
                i1 = i
                j1 = row.find(pair[1])
        ciphetextpair = key_matrix[i0][j1] + key_matrix[i1][j0]
        ciphetextpairs.append(ciphetextpair)
    return "".join(ciphetextpairs)

def decrypt(ciphetext):
    encryptedtextpairs =[]
    ciphetextpairs = []
    # Rule 1: if both latter are same or only one left add "X" after first letter
    i = 0
    while i<len(ciphetext):
        a = ciphetext[i]
        b = ciphetext[i+1]
        ciphetextpairs.append(a + b)
        i+=2
    # print(ciphetextpairs)

    for pair in ciphetextpairs:
        applied_rule = False
        for row in key_matrix:
            if pair[0] in row and pair[1] in row:
                j0 = row.find(pair[0])
                j1 = row.find(pair[1])
                encryptedtextpair = row[(j0 + 4) % 5] + row[(j1 + 4) % 5]
                encryptedtextpairs.append(encryptedtextpair)
                applied_rule = True
        if applied_rule:
            continue
            # Rule 3 :If letter are in same column, replace them with immediate below letter
        for j in range(5):
            col = "".join([key_matrix[i][j] for i in range(5)])
            if pair[0] in col and pair[1] in col:
                i0 = col.find(pair[0])
                i1 = col.find(pair[1])
                encryptedtextpair = col[(i0 + 4) % 5] + col[(i1 + 4) % 5]
                encryptedtextpairs.append(encryptedtextpair)
                applied_rule = True
        if applied_rule:
            continue
        # Rule 4: not in same column or row,replace them with the letters on same row
respectively but at
        # the other pair of corners of the rectangle define by the original pairs
        i0 = 0
        i1 = 0
        j0 = 0
        j1 = 0
        for i in range(5):
            row = key_matrix[i]
            if pair[0] in row:
                i0 = i
                j0 = row.find(pair[0])
            if pair[1] in row:
                i1 = i
                j1 = row.find(pair[1])
        encryptedtextpair = key_matrix[i0][j1] + key_matrix[i1][j0]
        encryptedtextpairs.append(encryptedtextpair)
    return "".join(encryptedtextpairs)


key = 'playfair example'
key_matrix = genKeyMat(key)
plainText = "hidethegoldinthetreestump"
ciphetext = encrypt(plainText)
print("Plain text: ",plainText)
print("Key: ",key)
print("Cipher text: ",ciphetext)
print("Decrypted text: ",decrypt(ciphetext))
```

OutPut:



```
"F:\OneDrive - oxyguard\study\IT\6th IT\Crypto\Practical\venv\Scripts\python.exe" "F:\OneDrive - oxyguard
Plain text:  hidethegoldinthetreestump
Key:  playfair example
Cipher text:  bmodzbxdnabekudmuixmmouvif
Decrypted text:  hidethegoldinthetrexestump

Process finished with exit code 0
```