A close-up of a school of engineering

Description automatically generated

**LAB PROGRAMS CYCLE 1**

**ON**

**CSA5195 - Cryptography and Network Security for Cyber Security**

**SLOT A**

**Submitted by**

**192321093 – MURALI KRISHNAN. S**

**To**

**DR. Anbalagan E**

**Saveetha School of Engineering**

**SIMATS, Thandalam.**

## 1. Caesar Cipher

**Aim:** The Caesar cipher shifts each letter in the plaintext forward by k positions in the alphabet.

### ****Code:****

def caesar\_cipher(text, key):

result = ""

for char in text:

if char.isalpha():

shift = key % 26

base = ord('A') if char.isupper() else ord('a')

result += chr((ord(char) - base + shift) % 26 + base)

else:

result += char

return result

# Example usage

plaintext = "HELLO"

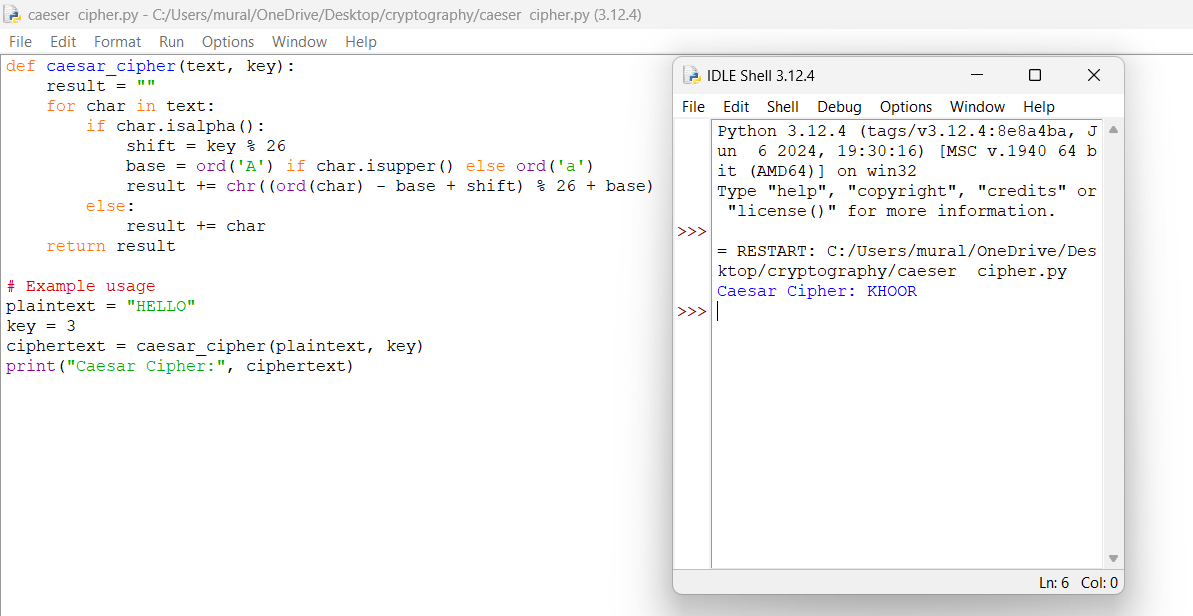
key = 3

ciphertext = caesar\_cipher(plaintext, key)

print("Caesar Cipher:", ciphertext)

### ****Sample Output:****

Caesar Cipher: KHOOR



## 2. Monoalphabetic Substitution Cipher

**Aim:** Each letter in the plaintext is mapped to a unique letter in the ciphertext using a predefined substitution table.

### ****Code:****

import random

import string

def generate\_key():

letters = list(string.ascii\_uppercase)

shuffled = letters[:]

random.shuffle(shuffled)

return dict(zip(letters, shuffled))

def monoalphabetic\_cipher(text, key):

return "".join(key.get(char, char) for char in text.upper())

# Example usage

key\_map = generate\_key()

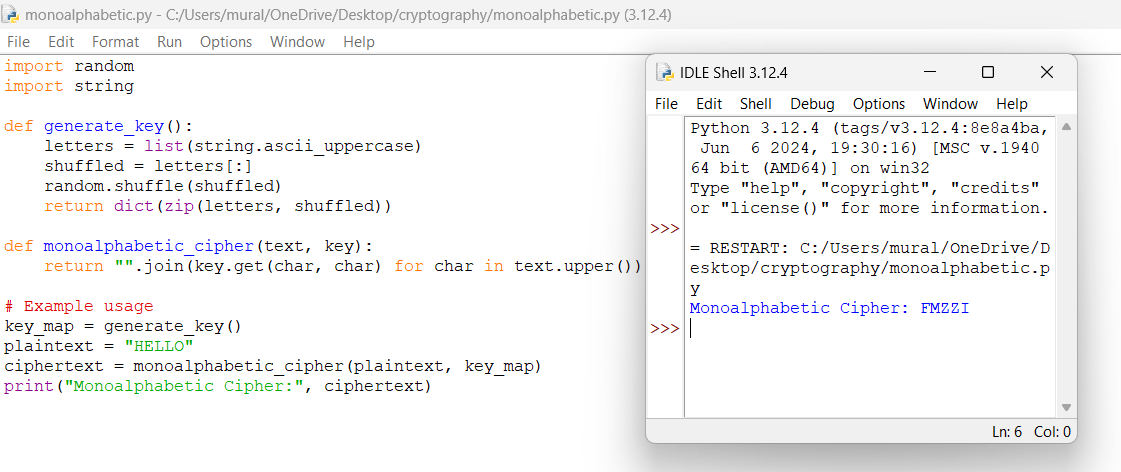
plaintext = "HELLO"

ciphertext = monoalphabetic\_cipher(plaintext, key\_map)

print("Monoalphabetic Cipher:", ciphertext)

### ****Sample Output:****

Monoalphabetic Cipher: XGWWO (varies due to random key)



## 3. Playfair Cipher

**Aim:** Uses a 5x5 matrix constructed from a keyword to encrypt digraphs (pairs of letters).

### ****Code:****

import numpy as np

def construct\_playfair\_matrix(keyword):

keyword = "".join(dict.fromkeys(keyword.upper().replace("J", "I")))

alphabet = "ABCDEFGHIKLMNOPQRSTUVWXYZ"

matrix = []

used\_chars = set(keyword)

for char in keyword + alphabet:

if char not in used\_chars:

matrix.append(char)

used\_chars.add(char)

return np.array(matrix).reshape(5, 5)

def find\_position(matrix, char):

for i in range(5):

for j in range(5):

if matrix[i][j] == char:

return i, j

def playfair\_encrypt(plaintext, matrix):

plaintext = plaintext.upper().replace("J", "I").replace(" ", "")

pairs = []

i = 0

while i < len(plaintext):

a = plaintext[i]

b = plaintext[i + 1] if i + 1 < len(plaintext) else "X"

if a == b:

b = "X"

i -= 1

pairs.append((a, b))

i += 2

ciphertext = ""

for a, b in pairs:

row1, col1 = find\_position(matrix, a)

row2, col2 = find\_position(matrix, b)

if row1 == row2:

ciphertext += matrix[row1][(col1 + 1) % 5] + matrix[row2][(col2 + 1) % 5]

elif col1 == col2:

ciphertext += matrix[(row1 + 1) % 5][col1] + matrix[(row2 + 1) % 5][col2]

else:

ciphertext += matrix[row1][col2] + matrix[row2][col1]

return ciphertext

# Example usage

keyword = "KEYWORD"

matrix = construct\_playfair\_matrix(keyword)

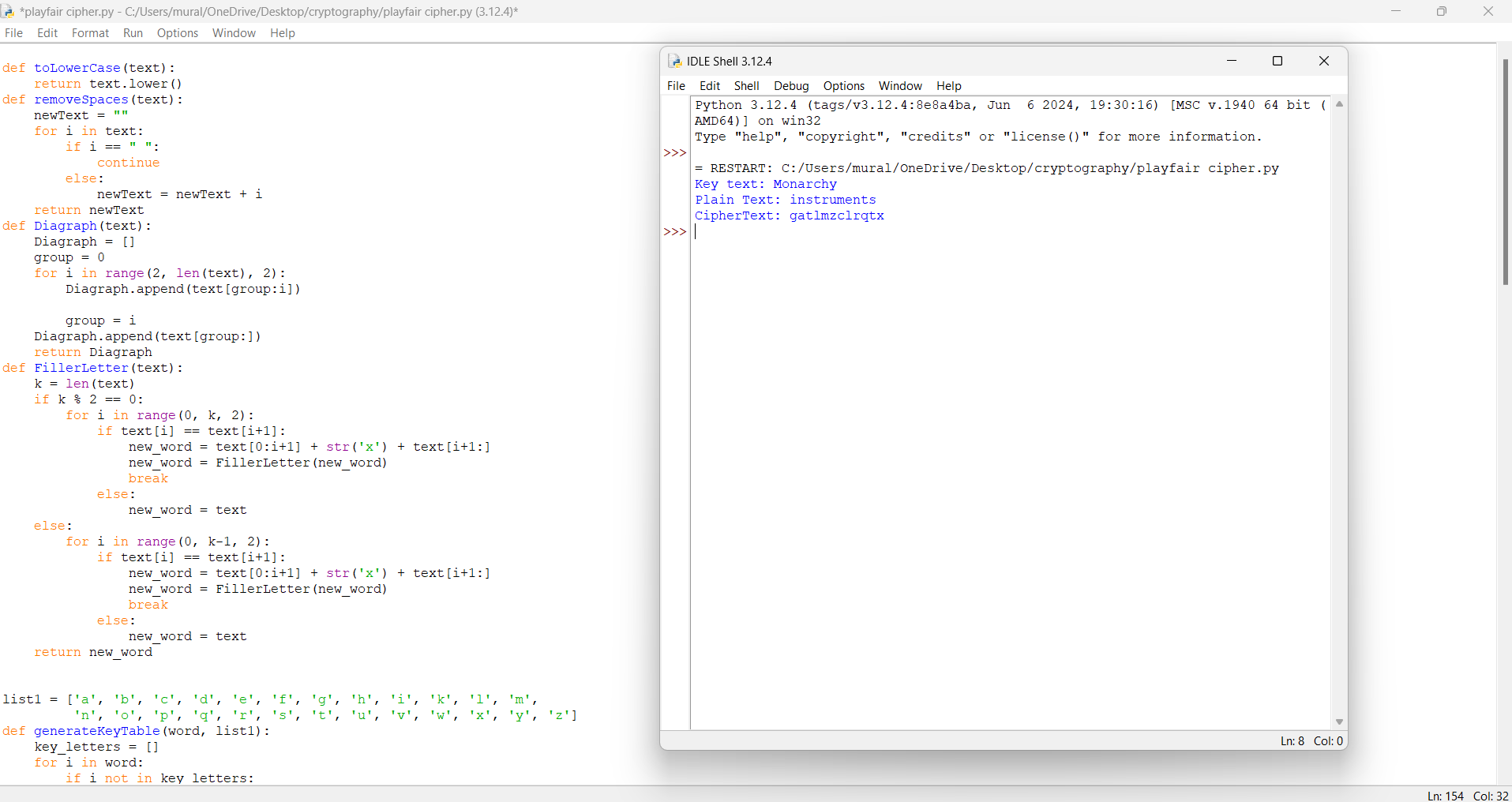
plaintext = "HELLO"

ciphertext = playfair\_encrypt(plaintext, matrix)

print("Playfair Cipher:", ciphertext)

### ****Sample Output:****

Playfair Cipher: ZYXXRM



## 4. Polyalphabetic Substitution Cipher (Vigenère Cipher)

**Aim:** Uses multiple Caesar ciphers determined by a repeating key.

### ****Code:****

def vigenere\_cipher(text, key):

key = key.upper()

result = ""

key\_index = 0

for char in text:

if char.isalpha():

shift = ord(key[key\_index % len(key)]) - ord('A')

base = ord('A') if char.isupper() else ord('a')

result += chr((ord(char) - base + shift) % 26 + base)

key\_index += 1

else:

result += char

return result

# Example usage

plaintext = "HELLO"

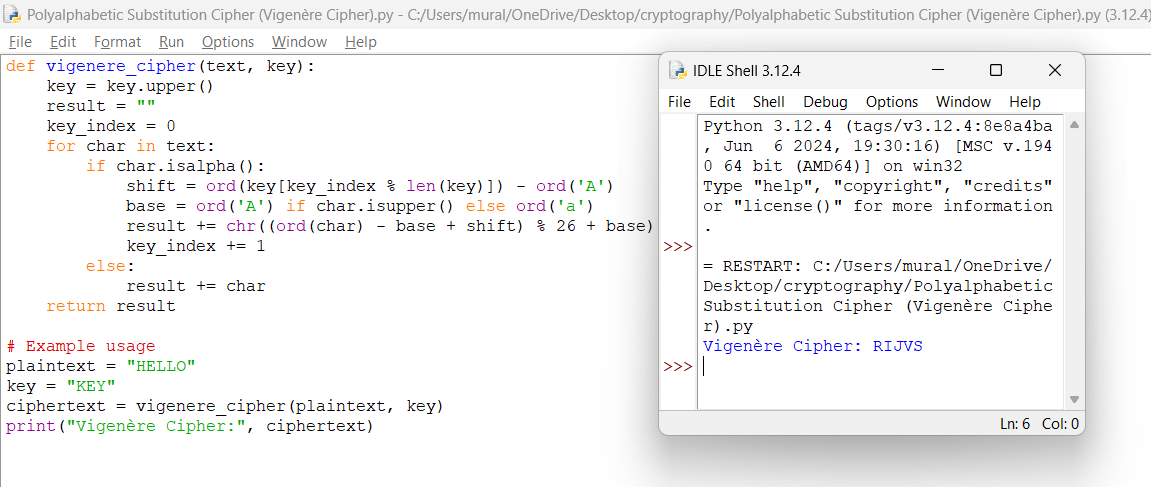
key = "KEY"

ciphertext = vigenere\_cipher(plaintext, key)

print("Vigenère Cipher:", ciphertext)

### ****Sample Output:****

Vigenère Cipher: RIJVS



5. Affine Caesar Cipher

**Aim:** Uses a mathematical function to substitute letters with (a \* p + b) mod 26.

### ****Code:****

import math

def affine\_encrypt(text, a, b):

if math.gcd(a, 26) != 1:

raise ValueError("Invalid value of 'a'. It must be coprime to 26.")

result = ""

for char in text:

if char.isalpha():

base = ord('A') if char.isupper() else ord('a')

result += chr(((a \* (ord(char) - base) + b) % 26) + base)

else:

result += char

return result

# Example usage

plaintext = "HELLO"

a, b = 5, 8 # Valid values of 'a' must be coprime with 26

ciphertext = affine\_encrypt(plaintext, a, b)

print("Affine Cipher:", ciphertext)

### ****Sample Output:****

Affine Cipher: AXEEH

