Tugas Kecil 1 IF 4020 Kriptografi
Pengaplikasian Kriptografi Klasik

Oleh:
Mohammad Ali Rido - 13515101
Putra Hardi Ramadhan - 1316080

Kriptografi IF4020

INSTITUT TEKNOLOGI BANDUNG
2020

Deskripsi Tugas

Buatlah sebuah program Java/C++/Python/Ruby/Golang dengan antarmuka (GUI) yang mengimplementasikan:

a) *Vigenere Cipher* standard (26 huruf alfabet)
b) *Varian Vigenere Cipher* (26 huruf alfabet): Full Vigenere Cipher dan Auto-key Vigenere Cipher)
c) *Extended Vigenere Cipher* (256 karakter ASCII)
d) *Playfair Cipher* (26 huruf alfabet)
e) Super enkripsi: Vigenere Cipher standard + *cipher* transposisi (bebas). Jelaskan cipher transposisi yang dibuat.
f) Affine cipher (26 huruf alfabet)
g) Hill cipher (26 huruf alfabet)
h) Bonus: Enigma cipher (26 huruf alfabet)

dengan spesifikasi sebagai berikut:

1. Program dapat menerima pesan berupa *file* sembarang (file text maupun file biner) atau pesan yang diketikkan dari papan-ketik.
2. Program dapat mengenkripsi plainteks. Khusus untuk *Vigenere Cipher* dengan 26 huruf alfabet dan *Playfair Cipher* dengan 26 huruf alfabet, program hanya mengenkripsi karakter alfabet saja. Angka, spasi, dan tanda baca dibuang.
3. Program dapat mendekripsi cipherteks menjadi plainteks semula.
4. Untuk pesan berupa text, program dapat menampilkan plainteks dan cipherteks di layar.
5. Untuk plainteks berupa text, cipherteks dapat ditampilkan ke layar dalam bentuk:
   (a) tanpa spasi
   (b) dalam kelompok 5-huruf
6. Program dapat menyimpan cipherteks ke dalam *file*.
7. Kunci dimasukkan oleh pengguna. Panjang kunci bebas.
8. Untuk enkripsi plainteks sembarang file (khusus untuk extended Vigenere Cipher), setiap file diperlakukan sebagai *file of bytes*. Program membaca setiap *byte* di dalam file (termasuk *byte-byte header file*) dan mengenkripsinya. Hanya saja file yang sudah terenkripsi tidak bisa dibuka oleh program aplikasinya karena header file ikut terenkripsi. Namun dengan mendekripsinya kembali maka file tersebut dapat dibuka oleh aplikasinya.

Deskripsi Aplikasi

Aplikasi ini dibuat menggunakan bahasa pemrograman python, dengan total 8 file module cipher, dan sebuah program utama. Tampilan antarmuka menggunakan command prompt.

Source Code

1. Vigenere Cipher

```
import string

message = ""
keycode = ""
enc_message = ""

def remove(string):
    return string.replace(" ", "")


def vigenere_encryption(plaintext, key):
    cipher = ""
    index = 0
    for c in plaintext:
        if c in string.ascii_lowercase:
            offset = ord(key[index]) - ord('a')

            encrypted_c = chr((ord(c) - ord('a') + offset) % 26 +
ord('a'))
            cipher = cipher + encrypted_c

            index = (index + 1)% len(key)
        else:
            cipher = cipher + c

    return cipher

def vigenere_decryption(cipher, key):
    plaintext = ""
    index = 0
    for c in cipher:

        if c in string.ascii_lowercase:
            offset = ord(key[index]) - ord('a')
            decrypted_c_num = ord(c) - ord('a') - offset
            if decrypted_c_num < 0:
                decrypted_c_num = decrypted_c_num + 26

            decrypted_c = chr(decrypted_c_num + ord('a'))

            plaintext = plaintext + decrypted_c
            index = (index + 1) % len(key)

        else:
            plaintext = plaintext + c
```

```
    return plaintext

def main():
    mode = int(input("1. Type Text\n2. Input File\nChoose(1,2): "))
    if mode == 1:

        message = input("Enter Message: ").lower()
        message = remove(message)
        keycode = input("Enter Key: ").lower()

        choice = int(input("1. Encrypttion\n2.
Decryption\nChoose(1,2): "))
        if choice == 1:

            space = int(input("1. No space\n2. Space every 5
char\nChoose(1,2):"))
            if space == 1:
                print("--Encryption--")
                enc_message = vigenere_encryption(message, keycode)
                print(enc_message)
            elif space == 2:
                print("--Encryption--")
                enc_message = vigenere_encryption(message, keycode)
                enc_message = " ".join(enc_message[i:i + 5] for i in
range(0, len(enc_message), 5))
                print(enc_message)
            else:
                print("Invalid Input")

        elif choice == 2:

            space = int(input("1. No space\n2. Space every 5
char\nChoose(1,2):"))
            if space == 1:
                print("--Decryption--")
                message = vigenere_decryption(message, keycode)
                print(message)
            elif space == 2:
                print("--Decryption--")
                message = vigenere_decryption(message, keycode)
                message = " ".join(message[i:i + 5] for i in range(0,
len(message), 5))
                print(message)
            else:
                print("Invalid Input")
        else:
            print("Invalid Input")

    elif mode == 2:
```

```
        infile_name = input("Enter input file name: ")
        infile = open(infile_name, 'r+')

        message = infile.read()
        message = remove(message).lower()
        keycode = input("Enter Key: ").lower()

        choice = int(input("1. Encryption\n2. Decryption\nChoose(1,2):
"))
        if choice == 1:

            space = int(input("1. No space\n2. Space every 5
char\nChoose(1,2):"))
            if space == 1:
                print("--Encryption--")
                enc_message = vigenere_encryption(message, keycode)
                print(enc_message)

                save = int(input("1. Save message to file\n2. Dont
Save\nChoose(1,2):"))
                if save == 1:
                    infile.write(enc_message)
                else:
                    pass

            elif space == 2:
                print("--Encryption--")
                enc_message = vigenere_encryption(message, keycode)
                enc_message = " ".join(enc_message[i:i + 5] for i in
range(0, len(enc_message), 5))
                print(enc_message)

                save = int(input("1. Save message to file\n2. Dont
Save\nChoose(1,2):"))
                if save == 1:
                    infile.write(enc_message)
                else:
                    pass
            else:
                print("Invalid Input")


        elif choice == 2:

            space = int(input("1. No space\n2. Space every 5
char\nChoose(1,2):"))
            if space == 1:
                print("--Decryption--")
                message = vigenere_decryption(message, keycode)
                print(message)
```

```
            elif space == 2:
                print("--Decryption--")
                message = vigenere_decryption(message, keycode)
                message = " ".join(message[i:i + 5] for i in range(0,
len(message), 5))
                print(message)
            else:
                print("Invalid Input")
        else:
            print("Invalid Input")
    else:
        print("Invalid input")


main()
```

2. Full Vigenere Cipher

```
import random
import string

def key_generator():
    letters = string.ascii_lowercase
    key = ''.join(random.choice(letters) for i in range(26))
    return key

message = ""
keycode = ""
enc_message = ""

def remove(string):
    return string.replace(" ", "")


def vigenere_encryption(plaintext, key):
    cipher = ""
    index = 0
    for c in plaintext:
        if c in string.ascii_lowercase:
            offset = ord(key[index]) - ord('a')

            encrypted_c = chr((ord(c) - ord('a') + offset) % 26 +
ord('a'))
            cipher = cipher + encrypted_c

            index = (index + 1)% len(key)
```

```
        else:
            cipher = cipher + c

    return cipher

def vigenere_decryption(cipher, key):
    plaintext = ""
    index = 0
    for c in cipher:

        if c in string.ascii_lowercase:
            offset = ord(key[index]) - ord('a')
            decrypted_c_num = ord(c) - ord('a') - offset
            if decrypted_c_num < 0:
                decrypted_c_num = decrypted_c_num + 26

            decrypted_c = chr(decrypted_c_num + ord('a'))

            plaintext = plaintext + decrypted_c
            index = (index + 1) % len(key)

        else:
            plaintext = plaintext + c

    return plaintext

def main():
    mode = int(input("1. Type Text\n2. Input File\nChoose(1,2): "))
    if mode == 1:

        message = input("Enter Message: ").lower()
        message = remove(message)
        keycode = key_generator()

        choice = int(input("1. Encrypttion\n2.
Decryption\nChoose(1,2): "))
        if choice == 1:

            space = int(input("1. No space\n2. Space every 5
char\nChoose(1,2):"))
            if space == 1:
                print("--Encryption--")
                enc_message = vigenere_encryption(message, keycode)
                print(enc_message)
            elif space == 2:
                print("--Encryption--")
                enc_message = vigenere_encryption(message, keycode)
                enc_message = " ".join(enc_message[i:i + 5] for i in
range(0, len(enc_message), 5))
                print(enc_message)
```

```
            else:
                print("Invalid Input")

        elif choice == 2:

            space = int(input("1. No space\n2. Space every 5
char\nChoose(1,2):"))
            if space == 1:
                print("--Decryption--")
                message = vigenere_decryption(message, keycode)
                print(message)
            elif space == 2:
                print("--Decryption--")
                message = vigenere_decryption(message, keycode)
                message = " ".join(message[i:i + 5] for i in range(0,
len(message), 5))
                print(message)
            else:
                print("Invalid Input")
        else:
            print("Invalid Input")

    elif mode == 2:

        infile_name = input("Enter input file name: ")
        infile = open(infile_name, 'r+')

        message = infile.read()
        message = remove(message).lower()
        keycode = key_generator()

        choice = int(input("1. Encryption\n2. Decryption\nChoose(1,2):
"))
        if choice == 1:

            space = int(input("1. No space\n2. Space every 5
char\nChoose(1,2):"))
            if space == 1:
                print("--Encryption--")
                enc_message = vigenere_encryption(message, keycode)
                print(enc_message)

                save = int(input("1. Save message to file\n2. Dont
Save\nChoose(1,2):"))
                if save == 1:
                    infile.write(enc_message)
                else:
                    pass

            elif space == 2:
```

```
                print("--Encryption--")
                enc_message = vigenere_encryption(message, keycode)
                enc_message = " ".join(enc_message[i:i + 5] for i in
range(0, len(enc_message), 5))
                print(enc_message)

                save = int(input("1. Save message to file\n2. Dont
Save\nChoose(1,2):"))
                if save == 1:
                    infile.write(enc_message)
                else:
                    pass
            else:
                print("Invalid Input")


        elif choice == 2:

            space = int(input("1. No space\n2. Space every 5
char\nChoose(1,2):"))
            if space == 1:
                print("--Decryption--")
                message = vigenere_decryption(message, keycode)
                print(message)
            elif space == 2:
                print("--Decryption--")
                message = vigenere_decryption(message, keycode)
                message = " ".join(message[i:i + 5] for i in range(0,
len(message), 5))
                print(message)
            else:
                print("Invalid Input")
        else:
            print("Invalid Input")
    else:
        print("Invalid input")


main()
```

3. Auto-Key Vigenere Cipher

```
import string


def remove(string):
    return string.replace(" ", "")
```

```
def generate_key(message, key):
    i = 0
    while True:
        if len(key) == len(message):
            break
        if message[i] == ' ':
            i += 1
        else:
            key += message[i]
            i += 1
    print(key)
    return key

def vigenere_encryption(plaintext, key):
    cipher = ""
    index = 0
    for c in plaintext:
        if c in string.ascii_lowercase:
            offset = ord(key[index]) - ord('a')

            encrypted_c = chr((ord(c) - ord('a') + offset) % 26 +
ord('a'))
            cipher = cipher + encrypted_c

            index = (index + 1)% len(key)
        else:
            cipher = cipher + c

    return cipher

def vigenere_decryption(cipher, key):
    plaintext = ""
    index = 0
    for c in cipher:

        if c in string.ascii_lowercase:
            offset = ord(key[index]) - ord('a')
            decrypted_c_num = ord(c) - ord('a') - offset
            if decrypted_c_num < 0:
                decrypted_c_num = decrypted_c_num + 26

            decrypted_c = chr(decrypted_c_num + ord('a'))

            plaintext = plaintext + decrypted_c
            index = (index + 1) % len(key)

        else:
            plaintext = plaintext + c

    return plaintext
```

```
def main():
    mode = int(input("1. Type Text\n2. Input File\nChoose(1,2): "))
    if mode == 1:

        message = input("Enter Message: ").lower()
        message = remove(message)
        keycode = input("Enter Key: ").lower()
        keycode = generate_key(message, keycode)

        choice = int(input("1. Encrypttion\n2.
Decryption\nChoose(1,2): "))
        if choice == 1:

            space = int(input("1. No space\n2. Space every 5
char\nChoose(1,2):"))
            if space == 1:
                print("--Encryption--")
                enc_message = vigenere_encryption(message, keycode)
                print(enc_message)
            elif space == 2:
                print("--Encryption--")
                enc_message = vigenere_encryption(message, keycode)
                enc_message = " ".join(enc_message[i:i + 5] for i in
range(0, len(enc_message), 5))
                print(enc_message)
            else:
                print("Invalid Input")

        elif choice == 2:

            space = int(input("1. No space\n2. Space every 5
char\nChoose(1,2):"))
            if space == 1:
                print("--Decryption--")
                message = vigenere_decryption(message, keycode)
                print(message)
            elif space == 2:
                print("--Decryption--")
                message = vigenere_decryption(message, keycode)
                message = " ".join(message[i:i + 5] for i in range(0,
len(message), 5))
                print(message)
            else:
                print("Invalid Input")
        else:
            print("Invalid Input")

    elif mode == 2:
```

```
        infile_name = input("Enter input file name: ")
        infile = open(infile_name, 'r+')

        message = infile.read()
        message = remove(message).lower()
        keycode = input("Enter Key: ").lower()
        keycode = generate_key(message, keycode)

        choice = int(input("1. Encryption\n2. Decryption\nChoose(1,2):
"))
        if choice == 1:

            space = int(input("1. No space\n2. Space every 5
char\nChoose(1,2):"))
            if space == 1:
                print("--Encryption--")
                enc_message = vigenere_encryption(message, keycode)
                print(enc_message)

                save = int(input("1. Save message to file\n2. Dont
Save\nChoose(1,2):"))
                if save == 1:
                    infile.write(enc_message)
                else:
                    pass

            elif space == 2:
                print("--Encryption--")
                enc_message = vigenere_encryption(message, keycode)
                enc_message = " ".join(enc_message[i:i + 5] for i in
range(0, len(enc_message), 5))
                print(enc_message)

                save = int(input("1. Save message to file\n2. Dont
Save\nChoose(1,2):"))
                if save == 1:
                    infile.write(enc_message)
                else:
                    pass
            else:
                print("Invalid Input")


        elif choice == 2:

            space = int(input("1. No space\n2. Space every 5
char\nChoose(1,2):"))
            if space == 1:
                print("--Decryption--")
                message = vigenere_decryption(message, keycode)
```

```
                print(message)
            elif space == 2:
                print("--Decryption--")
                message = vigenere_decryption(message, keycode)
                message = " ".join(message[i:i + 5] for i in range(0,
len(message), 5))
                print(message)
            else:
                print("Invalid Input")
        else:
            print("Invalid Input")
    else:
        print("Invalid input")


main()
```

4. Extended Vigenere Cipher

```
import string

message = ""
keycode = ""
enc_message = ""

def remove(string):
    return string.replace(" ", "")


def vigenere_encryption(plaintext, key):
    cipher = ""
    index = 0
    for c in plaintext:
        if c in string.ascii_lowercase:
            offset = ord(key[index]) - ord('a')

            encrypted_c = chr((ord(c) - ord('a') + offset) % 256 +
ord('a'))
            cipher = cipher + encrypted_c

            index = (index + 1)% len(key)
        else:
            cipher = cipher + c

    return cipher

def vigenere_decryption(cipher, key):
    plaintext = ""
```

```
    index = 0
    for c in cipher:

        if c in string.ascii_lowercase:
            offset = ord(key[index]) - ord('a')
            decrypted_c_num = ord(c) - ord('a') - offset
            if decrypted_c_num < 0:
                decrypted_c_num = decrypted_c_num + 256

            decrypted_c = chr(decrypted_c_num + ord('a'))

            plaintext = plaintext + decrypted_c
            index = (index + 1) % len(key)

        else:
            plaintext = plaintext + c

    return plaintext

def main():
    mode = int(input("1. Type Text\n2. Input File\nChoose(1,2): "))
    if mode == 1:

        message = input("Enter Message: ")
        message = remove(message)
        keycode = input("Enter Key: ")

        choice = int(input("1. Encrypttion\n2.
Decryption\nChoose(1,2): "))
        if choice == 1:

            space = int(input("1. No space\n2. Space every 5
char\nChoose(1,2):"))
            if space == 1:
                print("--Encryption--")
                enc_message = vigenere_encryption(message, keycode)
                print(enc_message)
            elif space == 2:
                print("--Encryption--")
                enc_message = vigenere_encryption(message, keycode)
                enc_message = " ".join(enc_message[i:i + 5] for i in
range(0, len(enc_message), 5))
                print(enc_message)
            else:
                print("Invalid Input")

        elif choice == 2:

            space = int(input("1. No space\n2. Space every 5
char\nChoose(1,2):"))
```

```
            if space == 1:
                print("--Decryption--")
                message = vigenere_decryption(message, keycode)
                print(message)
            elif space == 2:
                print("--Decryption--")
                message = vigenere_decryption(message, keycode)
                message = " ".join(message[i:i + 5] for i in range(0,
len(message), 5))
                print(message)
            else:
                print("Invalid Input")
        else:
            print("Invalid Input")

    elif mode == 2:

        infile_name = input("Enter input file name: ")
        infile = open(infile_name, 'r+')

        message = infile.read()
        message = remove(message)
        keycode = input("Enter Key: ")

        choice = int(input("1. Encryption\n2. Decryption\nChoose(1,2):
"))
        if choice == 1:

            space = int(input("1. No space\n2. Space every 5
char\nChoose(1,2):"))
            if space == 1:
                print("--Encryption--")
                enc_message = vigenere_encryption(message, keycode)
                print(enc_message)

                save = int(input("1. Save message to file\n2. Dont
Save\nChoose(1,2):"))
                if save == 1:
                    infile.write(enc_message)
                else:
                    pass

            elif space == 2:
                print("--Encryption--")
                enc_message = vigenere_encryption(message, keycode)
                enc_message = " ".join(enc_message[i:i + 5] for i in
range(0, len(enc_message), 5))
                print(enc_message)

                save = int(input("1. Save message to file\n2. Dont
```

```
Save\nChoose(1,2):"))
                if save == 1:
                    infile.write(enc_message)
                else:
                    pass
            else:
                print("Invalid Input")


        elif choice == 2:

            space = int(input("1. No space\n2. Space every 5
char\nChoose(1,2):"))
            if space == 1:
                print("--Decryption--")
                message = vigenere_decryption(message, keycode)
                print(message)
            elif space == 2:
                print("--Decryption--")
                message = vigenere_decryption(message, keycode)
                message = " ".join(message[i:i + 5] for i in range(0,
len(message), 5))
                print(message)
            else:
                print("Invalid Input")
        else:
            print("Invalid Input")
    else:
        print("Invalid input")


main()
```

5. Super Enkripsi

```
# Python3 implementation of
# Columnar Transposition
import math
import string

def remove(string):
    return string.replace(" ", "")


def vigenere_encryption(plaintext, key):
    cipher = ""
    index = 0
    for c in plaintext:
```

```
        if c in string.ascii_lowercase:
            offset = ord(key[index]) - ord('a')

            encrypted_c = chr((ord(c) - ord('a') + offset) % 26 +
ord('a'))
            cipher = cipher + encrypted_c

            index = (index + 1)% len(key)
        else:
            cipher = cipher + c

    return cipher

def vigenere_decryption(cipher, key):
    plaintext = ""
    index = 0
    for c in cipher:

        if c in string.ascii_lowercase:
            offset = ord(key[index]) - ord('a')
            decrypted_c_num = ord(c) - ord('a') - offset
            if decrypted_c_num < 0:
                decrypted_c_num = decrypted_c_num + 26

            decrypted_c = chr(decrypted_c_num + ord('a'))

            plaintext = plaintext + decrypted_c
            index = (index + 1) % len(key)

        else:
            plaintext = plaintext + c

    return plaintext

# Encryption
def trans_encrypt(msg, key):
    cipher = ""

    # track key indices
    k_indx = 0

    msg_len = float(len(msg))
    msg_lst = list(msg)
    key_lst = sorted(list(key))

    # calculate column of the matrix
    col = len(key)

    # calculate maximum row of the matrix
    row = int(math.ceil(msg_len / col))
```

```
    # add the padding character '_' in empty
    # the empty cell of the matix
    fill_null = int((row * col) - msg_len)
    msg_lst.extend('_' * fill_null)

    # create Matrix and insert message and
    # padding characters row-wise
    matrix = [msg_lst[i: i + col]
        for i in range(0, len(msg_lst), col)]
    print(matrix)

    # read matrix column-wise using key
    for _ in range(col):
        curr_idx = key.index(key_lst[k_indx])
        cipher += ''.join([row[curr_idx]
            for row in matrix])
        k_indx += 1

    return cipher

# Decryption
def trans_decrypt(cipher, key):
    msg = ""

    # track key indices
    k_indx = 0

    # track msg indices
    msg_indx = 0
    msg_len = float(len(cipher))
    msg_lst = list(cipher)

    # calculate column of the matrix
    col = len(key)

    # calculate maximum row of the matrix
    row = int(math.ceil(msg_len / col))

    # convert key into list and sort
    # alphabetically so we can access
    # each character by its alphabetical position.
    key_lst = sorted(list(key))

    # create an empty matrix to
    # store deciphered message
    dec_cipher = []
    for _ in range(row):
        dec_cipher += [[None] * col]
```

```
    # Arrange the matrix column wise according
    # to permutation order by adding into new matrix
    for _ in range(col):
        curr_idx = key.index(key_lst[k_indx])

        for j in range(row):
                dec_cipher[j][curr_idx] = msg_lst[msg_indx]
                msg_indx += 1
        k_indx += 1

    # convert decrypted msg matrix into a string
    try:
        msg = ''.join(sum(dec_cipher, []))
    except TypeError:
        raise TypeError("This program cannot, handle repeating
words.")

    null_count = msg.count('_')

    if null_count > 0:
        return msg[: -null_count]

    return msg



def main():
    mode = int(input("1. Type Text\n2. Input File\nChoose(1,2): "))
    if mode == 1:

        message = input("Enter Message: ").lower()
        message = remove(message)
        keycode = input("Enter Key: ").lower()

        choice = int(input("1. Encrypttion\n2.
Decryption\nChoose(1,2): "))
        if choice == 1:

            space = int(input("1. No space\n2. Space every 5
char\nChoose(1,2):"))
            if space == 1:
                print("--Encryption--")
                enc_message = vigenere_encryption(message, keycode)
                enc_message = trans_encrypt(enc_message,keycode)
                print(enc_message)
            elif space == 2:
                print("--Encryption--")
                enc_message = vigenere_encryption(message, keycode)
                enc_message = trans_encrypt(enc_message,keycode)
                enc_message = " ".join(enc_message[i:i + 5] for i in
```

```
range(0, len(enc_message), 5))
                print(enc_message)
            else:
                print("Invalid Input")

        elif choice == 2:

            space = int(input("1. No space\n2. Space every 5
char\nChoose(1,2):"))
            if space == 1:
                print("--Decryption--")
                message = (trans_decrypt(message,keycode))
                message = vigenere_decryption(message, keycode)
                print(message)
            elif space == 2:
                print("--Decryption--")
                message = (trans_decrypt(message,keycode))
                message = vigenere_decryption(message, keycode)
                message = " ".join(message[i:i + 5] for i in range(0,
len(message), 5))
                print(message)
            else:
                print("Invalid Input")
        else:
            print("Invalid Input")

    elif mode == 2:

        infile_name = input("Enter input file name: ")
        infile = open(infile_name, 'r+')

        message = infile.read()
        message = remove(message).lower()
        keycode = input("Enter Key: ").lower()

        choice = int(input("1. Encryption\n2. Decryption\nChoose(1,2):
"))
        if choice == 1:

            space = int(input("1. No space\n2. Space every 5
char\nChoose(1,2):"))
            if space == 1:
                print("--Encryption--")
                enc_message = vigenere_encryption(message, keycode)
                enc_message = trans_encrypt(enc_message,keycode)
                print(enc_message)

                save = int(input("1. Save message to file\n2. Dont
Save\nChoose(1,2):"))
                if save == 1:
```

```
                    infile.write(enc_message)
                else:
                    pass

            elif space == 2:
                print("--Encryption--")
                enc_message = vigenere_encryption(message, keycode)
                enc_message = trans_encrypt(enc_message,keycode)
                enc_message = " ".join(enc_message[i:i + 5] for i in
range(0, len(enc_message), 5))
                print(enc_message)

                save = int(input("1. Save message to file\n2. Dont
Save\nChoose(1,2):"))
                if save == 1:
                    infile.write(enc_message)
                else:
                    pass
            else:
                print("Invalid Input")


        elif choice == 2:

            space = int(input("1. No space\n2. Space every 5
char\nChoose(1,2):"))
            if space == 1:
                print("--Decryption--")
                message = (trans_decrypt(message,keycode))
                message = vigenere_decryption(message, keycode)
                print(message)
            elif space == 2:
                print("--Decryption--")
                message = (trans_decrypt(message,keycode))
                message = vigenere_decryption(message, keycode)
                message = " ".join(message[i:i + 5] for i in range(0,
len(message), 5))
                print(message)
            else:
                print("Invalid Input")
        else:
            print("Invalid Input")
    else:
        print("Invalid input")


main()
```

6. Playfair Cipher

```
import string

def remove(string):
    return string.replace(" ", "")

#Making Key Matrix
def key_matrix_generation(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


#Encryption function
def playfair_encrypt(plaintext, plaintextpairs, key_matrix):
    i = 0
    ciphertext = ""
    ciphertextpairs = []

    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

    print(plaintextpairs)


    #RULE 2
    #Jika pasangan huruf muncul di row yang sama
    #Tukar dengan yang ada di sebelah kanan tiap huruf

    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])

                ciphertextpair = row[(j0 + 1)%5] + row[(j1 + 1)%5]
                ciphertextpairs.append(ciphertextpair)
                applied_rule = True

        if applied_rule:
            continue

        #RULE 3
        #Jika pasangan huruf muncul di column yang sama
        #Tukar dengan yang ada di bawah tiap huruf
        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])

                ciphertextpair = col[(i0 + 1)%5] + col[(i1 + 1)%5]
                ciphertextpairs.append(ciphertextpair)
                applied_rule = True

        if applied_rule:
            continue

        #RULE 4
        #Tukar pasangan huruf dengan huruf yang ada di sudut
        #Dari segiempat yang dibentuk kedua huruf pada matrix
        # Huruf pertama = baris huruf pertama, kolom huruf kedua
        # Huruf kedua = baris huruf kedua, kolom huruf pertama
        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])

        ciphertextpair = key_matrix[i0][j1] + key_matrix[i1][j0]
        ciphertextpairs.append(ciphertextpair)

    return ciphertextpairs

#Decryption function
def playfair_decrypt(ciphertext, ciphertextpairs, key_matrix):
    i = 0
    plaintext = ""
    plaintextpairs = []

    while i < len(ciphertext):
        a = ciphertext[i]
        b = ciphertext[i+1]

        ciphertextpairs.append(a + b)
        i += 2

    print(ciphertextpairs)


    #RULE 2
    #Jika pasangan huruf muncul di row yang sama
    #Tukar dengan yang ada di sebelah kanan tiap huruf

    for pair in ciphertextpairs:
        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])

                plaintextpair = row[(j0 + 4)%5] + row[(j1 + 4)%5]
                plaintextpairs.append(plaintextpair)
                applied_rule = True

        if applied_rule:
            continue
```

```
        #RULE 3
        #Jika pasangan huruf muncul di column yang sama
        #Tukar dengan yang ada di bawah tiap huruf
        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])

                plaintextpair = col[(i0 + 4)%5] + col[(i1 + 4)%5]
                plaintextpairs.append(plaintextpair)
                applied_rule = True

        if applied_rule:
            continue

        #RULE 4
        #Tukar pasangan huruf dengan huruf yang ada di sudut
        #Dari segiempat yang dibentuk kedua huruf pada matrix
        # Huruf pertama = baris huruf pertama, kolom huruf kedua
        # Huruf kedua = baris huruf kedua, kolom huruf pertama
        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])

        plaintextpair = key_matrix[i0][j1] + key_matrix[i1][j0]
        plaintextpairs.append(plaintextpair)

    return plaintextpairs


def main():
    mode = int(input("1. Type Text\n2. Input File\nChoose(1,2): "))
    if mode == 1:
        msg = input("Enter Message: ").lower()
        msg = remove(msg)
        keycode = input("Enter Key: ").lower()
        matrix = key_matrix_generation(str(keycode))

        choice = int(input("1. Encryption\n2. Decryption\nChoose(1,2):
```

```
"))
        if choice == 1:

            space = int(input("1. No space\n2. Space every 5
char\nChoose(1,2):"))
            if space == 1:
                print("--Encryption--")
                pairs = []
                encrypt_result = playfair_encrypt(msg, pairs, matrix)
                print("ciphertext: " + "".join(encrypt_result))
            elif space == 2:
                print("--Encryption--")
                pairs = []
                encrypt_result = playfair_encrypt(msg, pairs, matrix)
                str1 = ""
                encrypt_result = str1.join(encrypt_result)
                encrypt_result= " ".join(encrypt_result[i:i + 5] for i
in range(0, len(encrypt_result), 5))
                print("ciphertext: " + "".join(encrypt_result))
            else:
                print("Invalid Input")

        elif choice == 2:

            space = int(input("1. No space\n2. Space every 5
char\nChoose(1,2):"))
            if space == 1:
                print("--Decryption--")
                pairs = []
                decrypt_result = playfair_decrypt(msg, pairs, matrix)
                print("plaintext: " + "".join(decrypt_result))
            elif space == 2:
                print("--Decryption--")
                pairs = []
                decrypt_result = playfair_decrypt(msg, pairs, matrix)
                str1 = ""
                decrypt_result = str1.join(decrypt_result)
                decrypt_result= " ".join(decrypt_result[i:i + 5] for i
in range(0, len(decrypt_result), 5))
                print("plaintext: " + "".join(decrypt_result))
            else:
                print("Invalid Input")

        else:
            print("Invalid input")


    elif mode == 2:
        infile_name = input("Enter input file name: ")
        infile = open(infile_name, 'r+')
```

```
        msg = infile.read()
        msg = remove(msg).lower()
        keycode = input("Enter Key: ").lower()

        matrix = key_matrix_generation(str(keycode))

        choice = int(input("1. Encryption\n2. Decryption\nChoose(1,2):
"))
        if choice == 1:

            space = int(input("1. No space\n2. Space every 5
char\nChoose(1,2):"))
            if space == 1:
                print("--Encryption--")
                pairs = []
                encrypt_result = playfair_encrypt(msg, pairs, matrix)
                print("ciphertext: " + "".join(encrypt_result))

                save = int(input("1. Save message to file\n2. Dont
Save\nChoose(1,2):"))
                if save == 1:
                    infile.write(encrypt_result)
                else:
                    pass

            elif space == 2:
                print("--Encryption--")
                pairs = []
                encrypt_result = playfair_encrypt(msg, pairs, matrix)
                str1 = ""
                encrypt_result = str1.join(encrypt_result)
                encrypt_result= " ".join(encrypt_result[i:i + 5] for i
in range(0, len(encrypt_result), 5))
                print("ciphertext: " + "".join(encrypt_result))

                save = int(input("1. Save message to file\n2. Dont
Save\nChoose(1,2):"))
                if save == 1:
                    infile.write(encrypt_result)
                else:
                    pass
            else:
                print("Invalid Input")

        elif choice == 2:

            space = int(input("1. No space\n2. Space every 5
char\nChoose(1,2):"))
            if space == 1:
```

```
                print("--Decryption--")
                pairs = []
                decrypt_result = playfair_decrypt(msg, pairs, matrix)
                print("plaintext: " + "".join(decrypt_result))
            elif space == 2:
                print("--Decryption--")
                pairs = []
                decrypt_result = playfair_decrypt(msg, pairs, matrix)
                str1 = ""
                decrypt_result = str1.join(decrypt_result)
                decrypt_result= " ".join(decrypt_result[i:i + 5] for i
in range(0, len(decrypt_result), 5))
                print("plaintext: " + "".join(decrypt_result))
            else:
                print("Invalid Input")

        else:
            print("Invalid input")


    else:
        print("Invalid Input")

main()
```

7. Affine Cipher

```
# Dictionary
dict1 = {'a': 0, 'b': 1, 'c': 2, 'd': 3, 'e': 4,
         'f': 5, 'g': 6, 'h': 7, 'i': 8, 'j': 9,
         'k': 10, 'l': 11, 'm': 12, 'n': 13, 'o': 14,
         'p': 15, 'q': 16, 'r': 17, 's': 18, 't': 19,
         'u': 20, 'v': 21, 'w': 22, 'x': 23, 'y': 24, 'z': 25}

dict2 = {0: 'a', 1: 'b', 2: 'c', 3: 'd', 4: 'e',
         5: 'f', 6: 'g', 7: 'h', 8: 'i', 9: 'j',
         10: 'k', 11: 'l', 12: 'm', 13: 'n', 14: 'o',
         15: 'p', 16: 'q', 17: 'r', 18: 's', 19: 't',
         20: 'u', 21: 'v', 22: 'w', 23: 'x', 24: 'y', 25: 'z'}

def remove(string):
    return string.replace(" ", "")

#C = (a*P + b) % 26
def affine_encrypt(msg, a, b):
    cipher = ''
    for letter in msg:
        if letter == ' ':
```

```
            cipher += ' '
        else:
            z = (a*dict1[letter] + b) % 26
            cipher += dict2[z]

    return cipher


#P = (a^-1 * (C - b)) % 26
def affine_decrypt(cipher, a, b):
    message = ''
    a_inv = 0
    flag = 0
    for i in range(26):
        flag = (a*i) % 26
        if flag == 1:
            a_inv = i
            break

    for letter in cipher:
        if letter == ' ':
            message += ' '
        else:
            z = (a_inv*(dict1[letter]-b)) % 26
            message += dict2[z]

    return message


def main():
    mode = int(input("1. Type Text\n2. Input File\nChoose(1,2): "))
    if mode == 1:
        msg = input("Enter Message: ")
        msg = remove(msg).lower()
        keycode = []

        for i in range(0, 2):
            if (i == 0):
                print("Enter a key")
            else:
                print("Enter b key")
            ele = int(input())

            keycode.append(ele)
            if (i == 0):
                print()

        print(keycode)

        choice = int(input("1. Encryption\n2. Decryption\nChoose(1,2):
```

```
"))

        if choice == 1:

            space = int(input("1. No space\n2. Space every 5
char\nChoose(1,2):"))
            if space == 1:
                print("--Encryption--")
                encrypted_text = affine_encrypt(msg, keycode[0],
keycode[1])
                print('Encrypted Text: {}'.format( encrypted_text
).lower())
            elif space == 2:
                print("--Encryption--")
                encrypted_text = affine_encrypt(msg, keycode[0],
keycode[1])
                encrypted_text = " ".join(encrypted_text[i:i + 5] for
i in range(0, len(encrypted_text), 5))
                print('Encrypted Text: {}'.format( encrypted_text
).lower())
            else:
                print("Invalid Input")


        elif choice == 2:
            space = int(input("1. No space\n2. Space every 5
char\nChoose(1,2):"))
            if space == 1:
                print("--Decryption--")
                decrypted_text = affine_decrypt(msg, keycode[0],
keycode[1])
                print('Decrypted Text: {}'.format(
decrypted_text).lower())
            elif space == 2:
                print("--Decryption--")
                decrypted_text = affine_decrypt(msg, keycode[0],
keycode[1])
                decrypted_text = " ".join(decrypted_text[i:i + 5] for
i in range(0, len(decrypted_text), 5))
                print('Decrypted Text: {}'.format(
decrypted_text).lower())
            else:
                print("Invalid Input")


    elif mode == 2:
        infile_name = input("Enter input file name: ")
        infile = open(infile_name, 'r+')
        msg = infile.read()
```

```
        msg = remove(msg).lower()
        keycode = []

        for i in range(0, 2):
            if (i == 0):
                print("Enter a key")
            else:
                print("Enter b key")
            ele = int(input())

            keycode.append(ele)
            if (i == 0):
                print()

        print(keycode)

        choice = int(input("1. Encryption\n2. Decryption\nChoose(1,2):
"))

        if choice == 1:
            space = int(input("1. No space\n2. Space every 5
char\nChoose(1,2):"))
            if space == 1:
                print("--Encryption--")
                encrypted_text = affine_encrypt(msg, keycode[0],
keycode[1])
                print('Encrypted Text: {}'.format( encrypted_text
).lower())

                save = int(input("1. Save message to file\n2. Dont
Save\nChoose(1,2):"))
                if save == 1:
                    infile.write(encrypted_text)
                else:
                    pass

            elif space == 2:
                print("--Encryption--")
                encrypted_text = affine_encrypt(msg, keycode[0],
keycode[1])
                encrypted_text = " ".join(encrypted_text[i:i + 5] for
i in range(0, len(encrypted_text), 5))
                print('Encrypted Text: {}'.format( encrypted_text
).lower())

                save = int(input("1. Save message to file\n2. Dont
Save\nChoose(1,2):"))
                if save == 1:
                    infile.write(encrypted_text)
                else:
```

```
                    pass

            else:
                print("Invalid Input")

        elif choice == 2:
            space = int(input("1. No space\n2. Space every 5
char\nChoose(1,2):"))
            if space == 1:
                print("--Decryption--")
                decrypted_text = affine_decrypt(msg, keycode[0],
keycode[1])
                print('Decrypted Text: {}'.format(
decrypted_text).lower())
            elif space == 2:
                print("--Decryption--")
                decrypted_text = affine_decrypt(msg, keycode[0],
keycode[1])
                decrypted_text = " ".join(decrypted_text[i:i + 5] for
i in range(0, len(decrypted_text), 5))
                print('Decrypted Text: {}'.format(
decrypted_text).lower())
            else:
                print("Invalid Input")
        else:
            print("Invalid input")
    else:
        print("Invalid Input")
main()
```

8. Hill Cipher

```
import sys
import numpy as np
import string


def remove(string):
    return string.replace(" ", "")

def hill_encryption(msg, key):

    # Jika panjang message ganjil append 0
    len_chk = 0
    if len(msg) % 2 != 0:
        msg += "0"
        len_chk = 1
```

```
    # msg to matrix
    row = 2
    col = int(len(msg)/2)
    msg2d = np.zeros((row, col), dtype=int)

    itr1 = 0
    itr2 = 0
    for i in range(len(msg)):
        if i % 2 == 0:
            msg2d[0][itr1] = int(ord(msg[i])-65)
            itr1 += 1
        else:
            msg2d[1][itr2] = int(ord(msg[i])-65)
            itr2 += 1


    key2d = np.zeros((2, 2), dtype=int)
    itr3 = 0
    for i in range(2):
        for j in range(2):
            key2d[i][j] = ord(key[itr3])-65
            itr3 += 1

    # Mencari Determinan
    deter = key2d[0][0] * key2d[1][1] - key2d[0][1] * key2d[1][0]
    deter = deter % 26

    # Mencari Invers
    mul_inv = -1
    for i in range(26):
        temp_inv = deter * i
        if temp_inv % 26 == 1:
            mul_inv = i
            break
        else:
            continue

    if mul_inv == -1:
        print("Invalid key")
        sys.exit()

    encryp_text = ""
    itr_count = int(len(msg)/2)
    if len_chk == 0:
        for i in range(itr_count):
            temp1 = msg2d[0][i] * key2d[0][0] + msg2d[1][i] *
key2d[0][1]
            encryp_text += chr((temp1 % 26) + 65)
            temp2 = msg2d[0][i] * key2d[1][0] + msg2d[1][i] *
key2d[1][1]
```

```
            encryp_text += chr((temp2 % 26) + 65)

    else:
        for i in range(itr_count-1):
            temp1 = msg2d[0][i] * key2d[0][0] + msg2d[1][i] *
key2d[0][1]
            encryp_text += chr((temp1 % 26) + 65)
            temp2 = msg2d[0][i] * key2d[1][0] + msg2d[1][i] *
key2d[1][1]
            encryp_text += chr((temp2 % 26) + 65)


    return encryp_text


def hill_decryption(msg, key):
    # Jika panjang message ganjil append 0
    len_chk = 0
    if len(msg) % 2 != 0:
        msg += "0"
        len_chk = 1

    # msg to matrix
    row = 2
    col = int(len(msg) / 2)
    msg2d = np.zeros((row, col), dtype=int)

    itr1 = 0
    itr2 = 0
    for i in range(len(msg)):
        if i % 2 == 0:
            msg2d[0][itr1] = int(ord(msg[i]) - 65)
            itr1 += 1
        else:
            msg2d[1][itr2] = int(ord(msg[i]) - 65)
            itr2 += 1



    key2d = np.zeros((2, 2), dtype=int)
    itr3 = 0
    for i in range(2):
        for j in range(2):
            key2d[i][j] = ord(key[itr3]) - 65
            itr3 += 1


    # Mencari Determinan
    deter = key2d[0][0] * key2d[1][1] - key2d[0][1] * key2d[1][0]
    deter = deter % 26
```

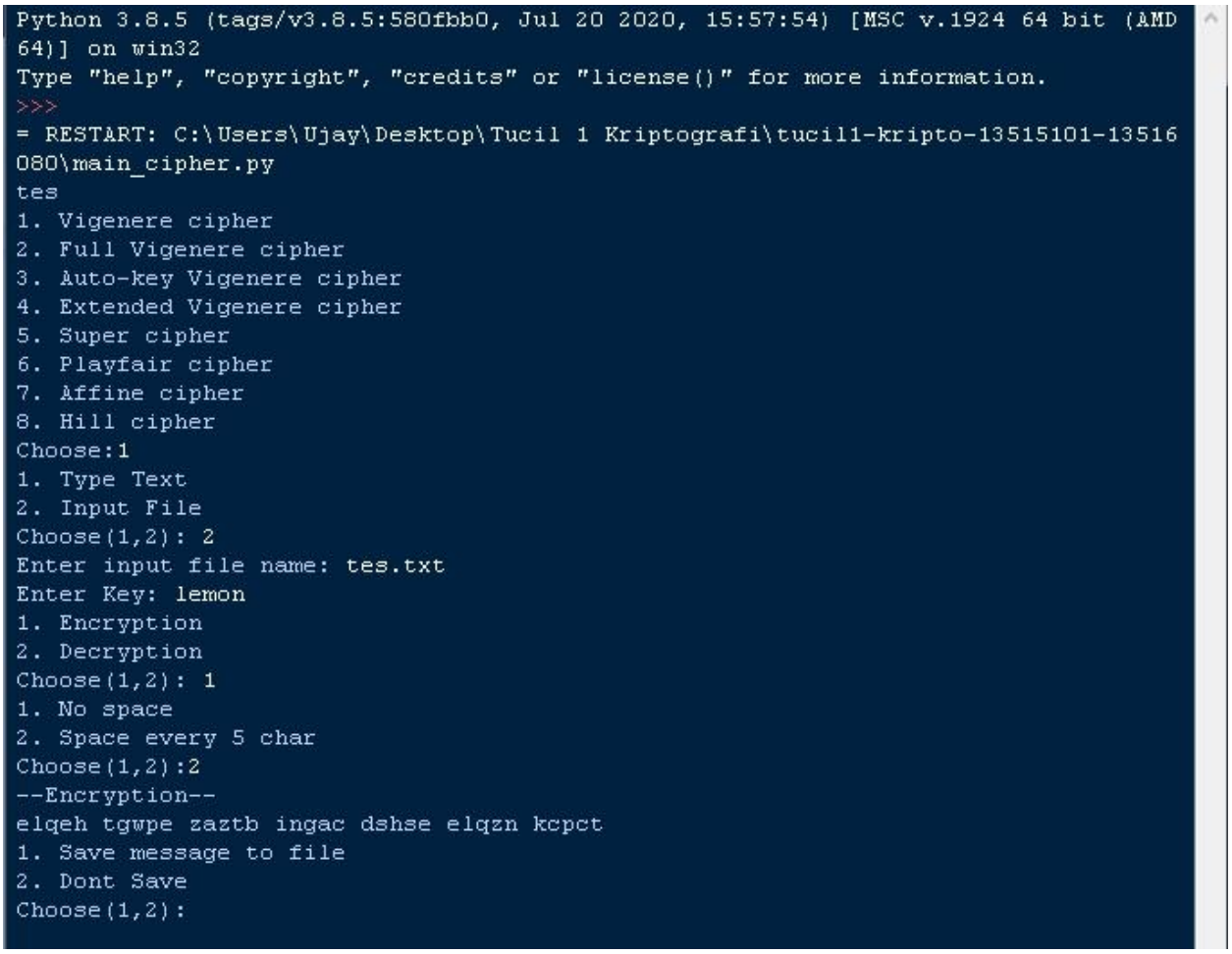
```
    # mencari invers
    mul_inv = -1
    for i in range(26):
        temp_inv = deter * i
        if temp_inv % 26 == 1:
            mul_inv = i
            break
        else:
            continue
    # for

    # adjoin
    # Transpos matrix
    key2d[0][0], key2d[1][1] = key2d[1][1], key2d[0][0]

    # Mengganti Tanda
    key2d[0][1] *= -1
    key2d[1][0] *= -1

    key2d[0][1] = key2d[0][1] % 26
    key2d[1][0] = key2d[1][0] % 26

    # invers dikali adjoin
    for i in range(2):
        for j in range(2):
            key2d[i][j] *= mul_inv

    # modulo
    for i in range(2):
        for j in range(2):
            key2d[i][j] = key2d[i][j] % 26

    # cipher to plain
    decryp_text = ""
    itr_count = int(len(msg) / 2)
    if len_chk == 0:
        for i in range(itr_count):
            temp1 = msg2d[0][i] * key2d[0][0] + msg2d[1][i] *
key2d[0][1]
            decryp_text += chr((temp1 % 26) + 65)
            temp2 = msg2d[0][i] * key2d[1][0] + msg2d[1][i] *
key2d[1][1]
            decryp_text += chr((temp2 % 26) + 65)

    else:
        for i in range(itr_count - 1):
            temp1 = msg2d[0][i] * key2d[0][0] + msg2d[1][i] *
key2d[0][1]
            decryp_text += chr((temp1 % 26) + 65)
```

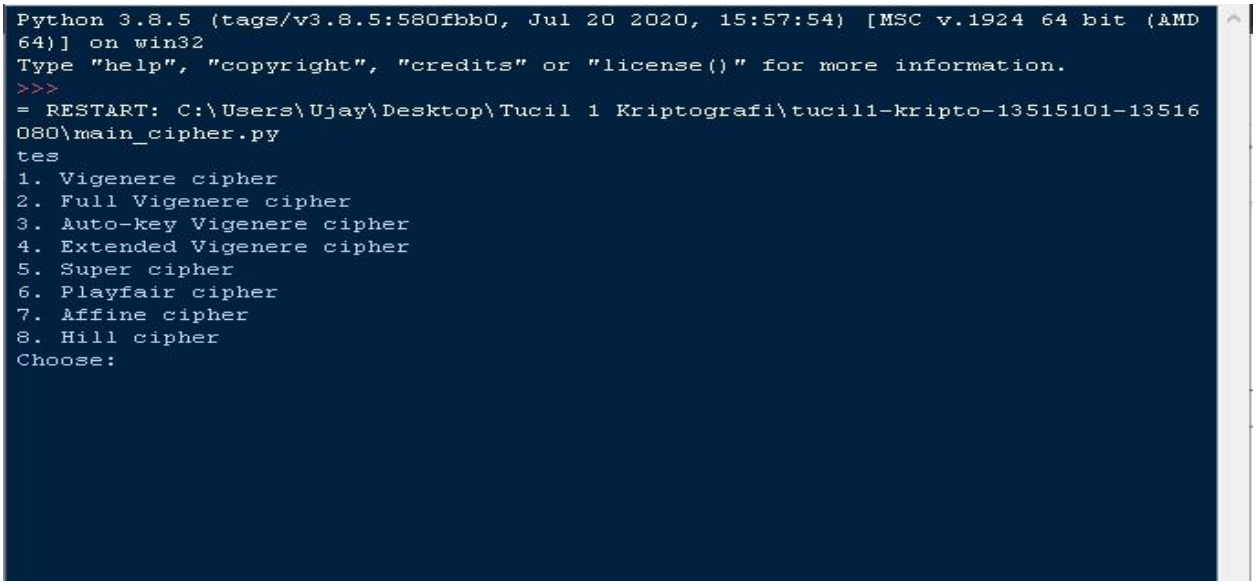
```
            temp2 = msg2d[0][i] * key2d[1][0] + msg2d[1][i] *
key2d[1][1]
            decryp_text += chr((temp2 % 26) + 65)

    return decryp_text


def main():
    mode = int(input("1. Type Text\n2. Input File\nChoose(1,2): "))
    if mode == 1:
        message = input("Enter message: ").upper()
        message = remove(message)
        keycode = input("Enter 4 letter Key String: ").upper()
        keycode = remove(keycode)

        choice = int(input("1. Encryption\n2. Decryption\nChoose(1,2):
"))
        if choice == 1:

            space = int(input("1. No space\n2. Space every 5
char\nChoose(1,2):"))
            if space == 1:
                print("---Encryption---")
                encrypted_text = hill_encryption(message, keycode)
                print("Encrypted Text:
{}".format(encrypted_text).lower())
            elif space == 2:
                print("---Encryption---")
                encrypted_text = hill_encryption(message, keycode)
                encrypted_text = " ".join(encrypted_text[i:i + 5] for
i in range(0, len(encrypted_text), 5))
                print("Encrypted Text:
{}".format(encrypted_text).lower())
            else:
                print("Invalid Input")



        elif choice == 2:
            space = int(input("1. No space\n2. Space every 5
char\nChoose(1,2):"))
            if space == 1:
                print("---Decryption---")
                decrypted_text = hill_decryption(message, keycode)
                print("Decrypted Text:
{}".format(encrypted_text).lower())
            elif space == 2:
                print("---Encryption---")
                decrypted_text = hill_decryption(message, keycode)
                decrypted_text = " ".join(decrypted_text[i:i + 5] for
```

```
i in range(0, len(decrypted_text), 5))
                print("Decrypted Text:
{}".format(decrypted_text).lower())
            else:
                print("Invalid Input")
        else:
            print("Invalid Input")

    elif mode == 2:
        infile_name = input("Enter input file name: ")
        infile = open(infile_name, 'r+')

        message = infile.read()
        message = remove(message).upper()

        keycode = input("Enter 4 letter Key String: ").upper()
        keycode = remove(keycode)

        choice = int(input("1. Encryption\n2. Decryption\nChoose(1,2):
"))
        if choice == 1:
            space = int(input("1. No space\n2. Space every 5
char\nChoose(1,2):"))
            if space == 1:
                print("---Encryption---")
                encrypted_text = hill_encryption(message, keycode)
                print("Encrypted Text:
{}".format(encrypted_text).lower())

                save = int(input("1. Save message to file\n2. Dont
Save\nChoose(1,2):"))
                if save == 1:
                    infile.write(encrypted_text)
                else:
                    pass

            elif space == 2:
                print("---Encryption---")
                encrypted_text = hill_encryption(message, keycode)
                encrypted_text = " ".join(encrypted_text[i:i + 5] for
i in range(0, len(encrypted_text), 5))
                print("Encrypted Text:
{}".format(encrypted_text).lower())

                save = int(input("1. Save message to file\n2. Dont
Save\nChoose(1,2):"))
                if save == 1:
                    infile.write(encrypted_text)
                else:
                    pass
```

```
            else:
                print("Invalid Input")


        elif choice == 2:
            space = int(input("1. No space\n2. Space every 5
char\nChoose(1,2):"))
            if space == 1:
                print("---Decryption---")
                decrypted_text = hill_decryption(message, keycode)
                print("Decrypted Text:
{}".format(encrypted_text).lower())
            elif space == 2:
                print("---Decryption---")
                decrypted_text = hill_decryption(message, keycode)
                decrypted_text = " ".join(decrypted_text[i:i + 5] for
i in range(0, len(decrypted_text), 5))
                print("Decrypted Text:
{}".format(decrypted_text).lower())
            else:
                print("Invalid Input")
        else:
            print("Invalid Input")
    else:
        print("Invalid input")
main()
```

Tampilan Antarmuka





Contoh plainteks dan cipherteks

1. **Vigenere Cipher**

Plain Teks = The quick brown fox jumps over the lazy dog
Key = Lemon
Cipherteks = elqeh tgwpe zaztb ingac dshse elqzn kcpct

2. **Full Vigenere Cipher**

Plain Teks = The quick brown fox jumps over the lazy dog
Key =
Cipherteks = scshn klmpq pybim ygolv sfsax lgzzr samqu

3. **Auto-Key Vigenere Cipher**

Plain Teks = The quick brown fox jumps over the lazy dog
Key = lemonthequickbrownfoxjumpsoverthela
Cipherteks = elqeh bjorl wyxgf lfhrd pxpqg lvzpr sfhzg

4. **Extended Vigenere Cipher**

Plain Teks = The quick brown fox jumps over the lazy dog
Key = Lemon
Cipherteks = Tsi} vnon |rr} uyy~ zzq sixo h{u

5. **Super Enkripsi**

Plain Teks = The quick brown fox jumps over the lazy dog
Key = Lemon
Cipherteks = lgans lcetz idekq wzghq phebc entep taszc

6. **Playfair Cipher**

Plain Teks = The quick brown fox jumps over the lazy dog
Key = Lemon
Cipherteks = rklrs pdihw eyfpm ynqni tmwls urbag vzkdi v

7. **Affine Cipher**

Plain Teks = The quick brown fox jumps over the lazy dog
Key = [17,20]
Cipherteks = fjkgw acilx yehby vrwqp oynkx fjkzu dmtys

8. **Hill Cipher**

Plain Teks = The quick brown fox jumps over the lazy dog
Key = bcej
Cipherteks = hjkek wwujb guxti dxiqb uqdqd fpmls vefi

Link Github :

https://github.com/Putrajayush/tucil1-kripto-13515101-13516080

No	Spek	Berhasil ( )	Kurang Berhasil()	Keterangan
1	Vigenere standard	V		
2	Full Vigenere Cipher	V		
3	Auto-Key Vigenere Cipher	V		
4	Extended Vigenere Cipher		V	Dekripsi masih salah
5	Playfair cipher	V		

6	Super enkripsi	V		
7	Affine cipher		V	Bug pada key
8	Hill cipher (matriks 3 x 3)		V	Matriks 2x2
9	Bonus: Enigma cipher		V	Tidak dibuat