Network Security and Forensics

Lab Session 5

Submitted To:-

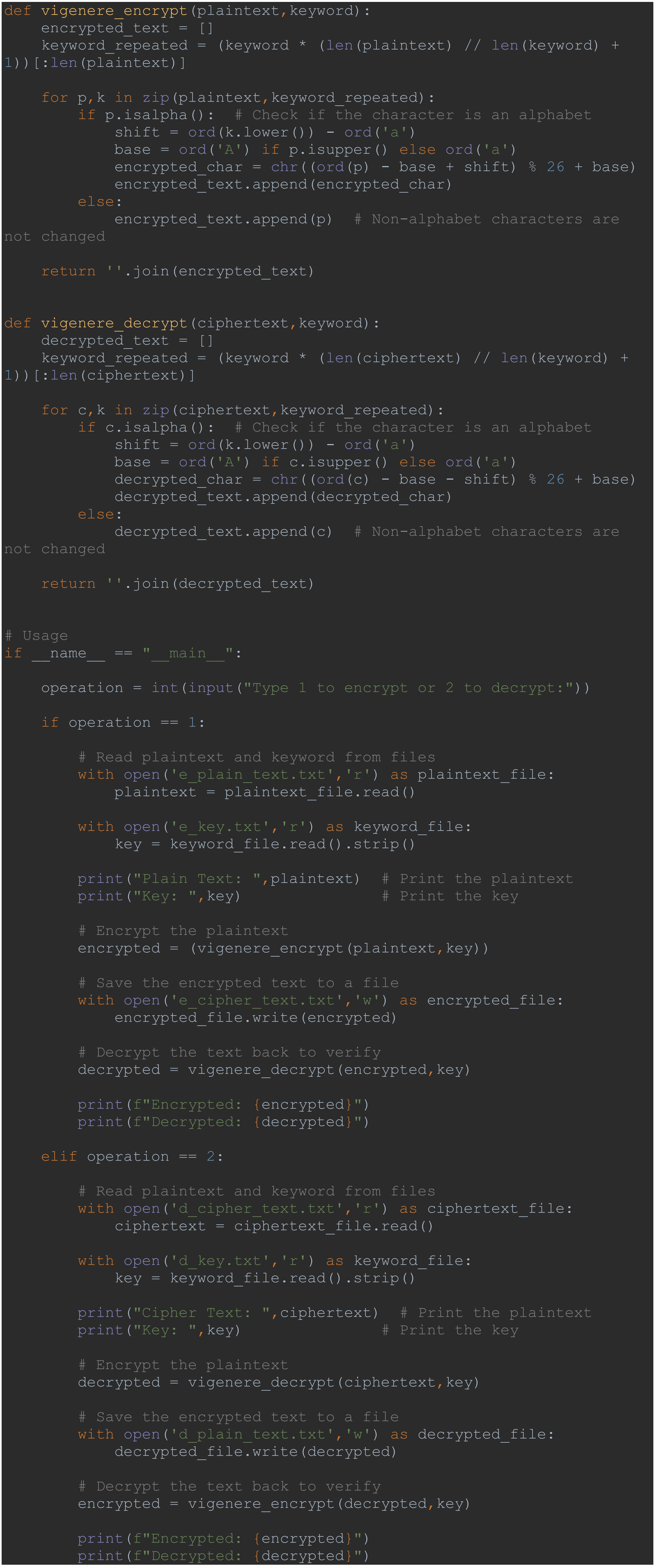
Dr. Lokesh Chauhan Sir

Submitted By:-

Saloni Rangari

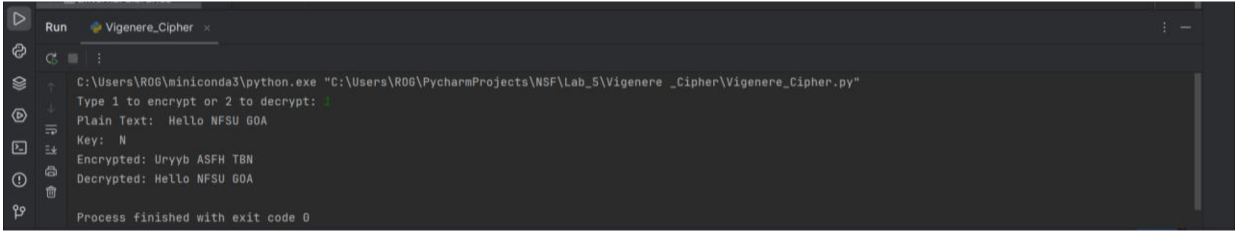
M.Tech. AIDS

1) **Write a program to demonstrate the Vegenere Cipher Input from file and Output into the file**



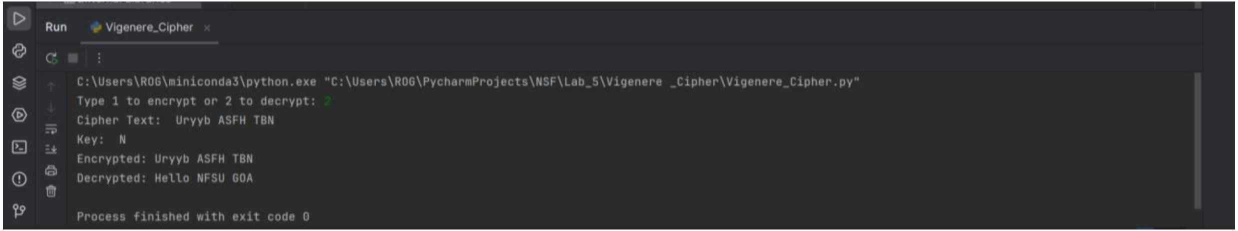
```
def vigenere_encrypt(plaintext,keyword):
    encrypted_text = []
    keyword_repeated = (keyword * (len(plaintext) // len(keyword) +
1))[:len(plaintext)]

    for p,k in zip(plaintext,keyword_repeated):
        if p.isalpha():  # Check if the character is an alphabet
            shift = ord(k.lower()) - ord('a')
            base = ord('A') if p.isupper() else ord('a')
            encrypted_char = chr((ord(p) - base + shift) % 26 + base)
            encrypted_text.append(encrypted_char)
        else:
            encrypted_text.append(p)  # Non-alphabet characters are
not changed

    return ''.join(encrypted_text)


def vigenere_decrypt(ciphertext,keyword):
    decrypted_text = []
    keyword_repeated = (keyword * (len(ciphertext) // len(keyword) +
1))[:len(ciphertext)]

    for c,k in zip(ciphertext,keyword_repeated):
        if c.isalpha():  # Check if the character is an alphabet
            shift = ord(k.lower()) - ord('a')
            base = ord('A') if c.isupper() else ord('a')
            decrypted_char = chr((ord(c) - base - shift) % 26 + base)
            decrypted_text.append(decrypted_char)
        else:
            decrypted_text.append(c)  # Non-alphabet characters are
not changed

    return ''.join(decrypted_text)


# Usage
if __name__ == "__main__":

    operation = int(input("Type 1 to encrypt or 2 to decrypt:"))

    if operation == 1:

        # Read plaintext and keyword from files
        with open('e_plain_text.txt','r') as plaintext_file:
            plaintext = plaintext_file.read()

        with open('e_key.txt','r') as keyword_file:
            key = keyword_file.read().strip()

        print("Plain Text: ",plaintext)  # Print the plaintext
        print("Key: ",key)               # Print the key

        # Encrypt the plaintext
        encrypted = (vigenere_encrypt(plaintext,key))

        # Save the encrypted text to a file
        with open('e_cipher_text.txt','w') as encrypted_file:
            encrypted_file.write(encrypted)

        # Decrypt the text back to verify
        decrypted = vigenere_decrypt(encrypted,key)

        print(f"Encrypted: {encrypted}")
        print(f"Decrypted: {decrypted}")

    elif operation == 2:

        # Read plaintext and keyword from files
        with open('d_cipher_text.txt','r') as ciphertext_file:
            ciphertext = ciphertext_file.read()

        with open('d_key.txt','r') as keyword_file:
            key = keyword_file.read().strip()

        print("Cipher Text: ",ciphertext)  # Print the plaintext
        print("Key: ",key)                 # Print the key

        # Encrypt the plaintext
        decrypted = vigenere_decrypt(ciphertext,key)

        # Save the encrypted text to a file
        with open('d_plain_text.txt','w') as decrypted_file:
            decrypted_file.write(decrypted)

        # Decrypt the text back to verify
        encrypted = vigenere_encrypt(decrypted,key)

        print(f"Encrypted: {encrypted}")
        print(f"Decrypted: {decrypted}")
```

Output: Encryption



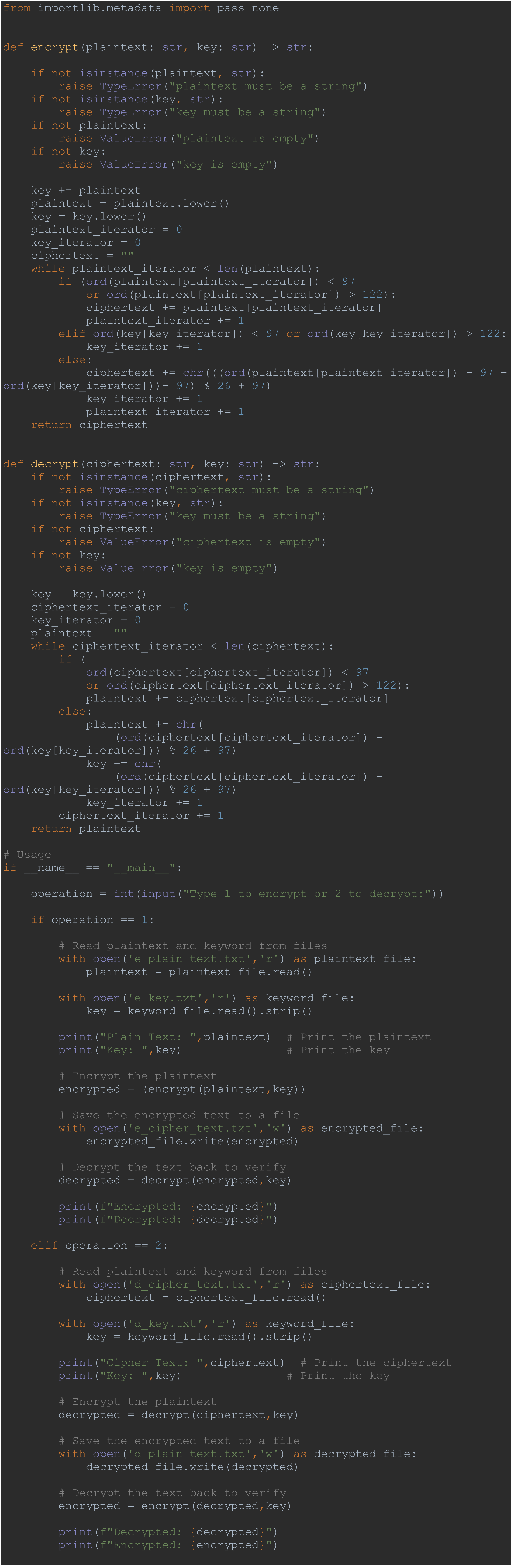
```
C:\Users\ROG\miniconda3\python.exe "C:\Users\ROG\PycharmProjects\NSF\Lab_5\Vigenere _Cipher\Vigenere_Cipher.py"
Type 1 to encrypt or 2 to decrypt: 1
Plain Text:  Hello NFSU GOA
Key:  N
Encrypted: Uryyb ASFH TBN
Decrypted: Hello NFSU GOA

Process finished with exit code 0
```

Output: Decryption



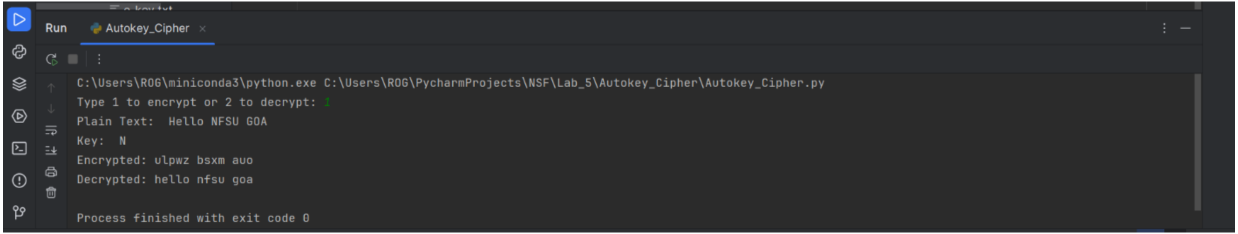
```
C:\Users\ROG\miniconda3\python.exe "C:\Users\ROG\PycharmProjects\NSF\Lab_5\Vigenere _Cipher\Vigenere_Cipher.py"
Type 1 to encrypt or 2 to decrypt: 2
Cipher Text:  Uryyb ASFH TBN
Key:  N
Encrypted: Uryyb ASFH TBN
Decrypted: Hello NFSU GOA

Process finished with exit code 0
```

2) **Write a Program to demonstrate the AutoKey Cipher. Input from file and Output into the file**



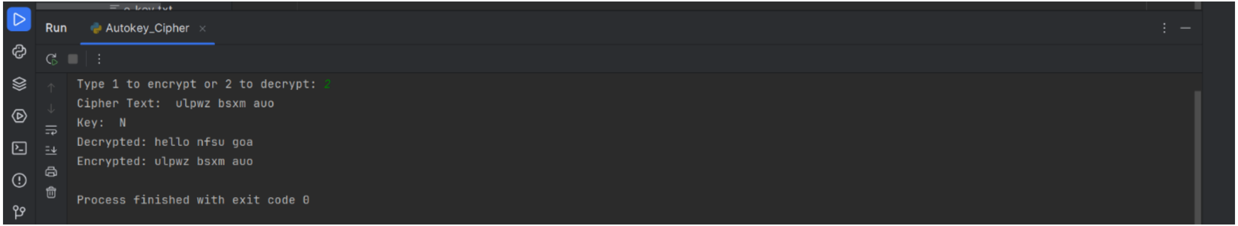
```
from importlib.metadata import pass_none


def encrypt(plaintext: str, key: str) -> str:

    if not isinstance(plaintext, str):
        raise TypeError("plaintext must be a string")
    if not isinstance(key, str):
        raise TypeError("key must be a string")
    if not plaintext:
        raise ValueError("plaintext is empty")
    if not key:
        raise ValueError("key is empty")

    key += plaintext
    plaintext = plaintext.lower()
    key = key.lower()
    plaintext_iterator = 0
    key_iterator = 0
    ciphertext = ""
    while plaintext_iterator < len(plaintext):
        if (ord(plaintext[plaintext_iterator]) < 97
            or ord(plaintext[plaintext_iterator]) > 122):
            ciphertext += plaintext[plaintext_iterator]
            plaintext_iterator += 1
        elif ord(key[key_iterator]) < 97 or ord(key[key_iterator]) > 122:
            key_iterator += 1
        else:
            ciphertext += chr(((ord(plaintext[plaintext_iterator]) - 97 +
ord(key[key_iterator]))- 97) % 26 + 97)
            key_iterator += 1
            plaintext_iterator += 1
    return ciphertext


def decrypt(ciphertext: str, key: str) -> str:
    if not isinstance(ciphertext, str):
        raise TypeError("ciphertext must be a string")
    if not isinstance(key, str):
        raise TypeError("key must be a string")
    if not ciphertext:
        raise ValueError("ciphertext is empty")
    if not key:
        raise ValueError("key is empty")

    key = key.lower()
    ciphertext_iterator = 0
    key_iterator = 0
    plaintext = ""
    while ciphertext_iterator < len(ciphertext):
        if (
            ord(ciphertext[ciphertext_iterator]) < 97
            or ord(ciphertext[ciphertext_iterator]) > 122):
            plaintext += ciphertext[ciphertext_iterator]
        else:
            plaintext += chr(
                (ord(ciphertext[ciphertext_iterator]) -
ord(key[key_iterator])) % 26 + 97)
            key += chr(
                (ord(ciphertext[ciphertext_iterator]) -
ord(key[key_iterator])) % 26 + 97)
            key_iterator += 1
        ciphertext_iterator += 1
    return plaintext

# Usage
if __name__ == "__main__":

    operation = int(input("Type 1 to encrypt or 2 to decrypt:"))

    if operation == 1:

        # Read plaintext and keyword from files
        with open('e_plain_text.txt','r') as plaintext_file:
            plaintext = plaintext_file.read()

        with open('e_key.txt','r') as keyword_file:
            key = keyword_file.read().strip()

        print("Plain Text: ",plaintext)  # Print the plaintext
        print("Key: ",key)               # Print the key

        # Encrypt the plaintext
        encrypted = (encrypt(plaintext,key))

        # Save the encrypted text to a file
        with open('e_cipher_text.txt','w') as encrypted_file:
            encrypted_file.write(encrypted)

        # Decrypt the text back to verify
        decrypted = decrypt(encrypted,key)

        print(f"Encrypted: {encrypted}")
        print(f"Decrypted: {decrypted}")

    elif operation == 2:

        # Read plaintext and keyword from files
        with open('d_cipher_text.txt','r') as ciphertext_file:
            ciphertext = ciphertext_file.read()

        with open('d_key.txt','r') as keyword_file:
            key = keyword_file.read().strip()

        print("Cipher Text: ",ciphertext)  # Print the ciphertext
        print("Key: ",key)                 # Print the key

        # Encrypt the plaintext
        decrypted = decrypt(ciphertext,key)

        # Save the encrypted text to a file
        with open('d_plain_text.txt','w') as decrypted_file:
            decrypted_file.write(decrypted)

        # Decrypt the text back to verify
        encrypted = encrypt(decrypted,key)

        print(f"Decrypted: {decrypted}")
        print(f"Encrypted: {encrypted}")
```

Output: Encryption



```
C:\Users\ROG\miniconda3\python.exe C:\Users\ROG\PycharmProjects\NSF\Lab_5\Autokey_Cipher\Autokey_Cipher.py
Type 1 to encrypt or 2 to decrypt: 1
Plain Text:  Hello NFSU GOA
Key:  N
Encrypted: ulpwz bsxm auo
Decrypted: hello nfsu goa

Process finished with exit code 0
```

Output: Decryption



```
Type 1 to encrypt or 2 to decrypt: 2
Cipher Text:  ulpwz bsxm auo
Key:  N
Decrypted: hello nfsu goa
Encrypted: ulpwz bsxm auo

Process finished with exit code 0
```