Encryption and decryption functions

Santiago Baca - A01656580

The provided segment of the Python program is designed for Vigenère cipher decryption. It consists of a function, `vigenereDecryption`, that takes an encrypted message and a key as input and returns the decrypted message.

The main part of the program prompts the user to input an encrypted message and a key, then prints the resulting decrypted message using the Vigenère decryption function.

The decryption process involves repeating the key to match the length of the encrypted message and shifting each character in the encrypted message based on the corresponding character in the key. This allows users to easily decrypt messages encrypted with the Vigenère cipher.

In [ ]:
```
def vigenereDecryption(encrypted_message, key):
    """
    Decrypts a message using the Vigenère cipher.

    Args:
        encrypted_message (str): The encrypted message.
        key (str): The key used for decryption.

    Returns:
        str: The decrypted message.
    """

    # Define the extended alphabet used for encryption
    extended_alphabet = "ABCDEFGHIJKLMNÑOPQRSTUVWXYZ0123456789"
    alphabet_indices = {char: idx for idx, char in enumerate(extended_alphabet)}
    # Repeat the key to match the length of the encrypted message
    key = key * (len(encrypted_message) // len(key)) + key[:len(encrypted_message) % len(key)]
    decrypted_message = [] # Empty list to store the decrypted chars

    for i in range(len(encrypted_message)):
        # Check if the character is an alphabet letter or a digit
        if encrypted_message[i].isalpha() or encrypted_message[i].isdigit():
            # Calculate the index of the decrypted character and add it to the decrypted message
            idx = alphabet_indices[encrypted_message[i]] - int(key[i])
            decrypted_message.append(extended_alphabet[idx])
        # If the character is not an alphabet letter or a digit, keep it unchanged
        else:
            decrypted_message.append(encrypted_message[i])

    return ''.join(decrypted_message)

def __main__():
    # Get the encrypted message and key, and decrypt it using the Vigenère decryption function
    encrypted_message = input("Enter the encrypted message: ").upper()
    key = input("Enter the key: ").upper()
    print("Decrypted message:", vigenereDecryption(encrypted_message, key))

__main__()
```

```
Decrypted message: 6IX9INE
```

The provided segment of the Python program implements a Vigenère cipher for encryption. It includes a function, `vigenereEncryption`, which takes a message and a key as input and returns the encrypted message.

The program prompts the user to input a message and a key, and then it prints the resulting encrypted message using the Vigenère cipher.

The encryption process involves repeating the key to match the length of the message and shifting each character in the message based on the corresponding character in the key. This allows users to easily encrypt their messages using the Vigenère cipher

In [ ]:
```
def vigenereEncryption(message, key):
    """
    Encrypts a message using the Vigenère cipher.

    Args:
        message (str): The message to be encrypted.
        key (str): The key used for encryption.

    Returns:
        str: The encrypted message.
    """

    # Define the extended alphabet used for encryption
    extended_alphabet = "ABCDEFGHIJKLMNÑOPQRSTUVWXYZ0123456789"
    alphabet_indices = {char: idx for idx, char in enumerate(extended_alphabet)}
    # Repeat the key to match the length of the message
    key = key * (len(message) // len(key)) + key[:len(message) % len(key)]
    encrypted_message = []  # Empty list to store the encrypted chars

    for i in range(len(message)):
        # Check if the character is an alphabet letter or a digit
        if message[i].isalpha() or message[i].isdigit():
            # Calculate the index of the encrypted character and add it to the encrypted message
            idx = (alphabet_indices[message[i]] + int(key[i])) % len(extended_alphabet)
            encrypted_message.append(extended_alphabet[idx])
        # If the character is not an alphabet letter or a digit, keep it unchanged
        else:
            encrypted_message.append(message[i])

    return ''.join(encrypted_message)

def __main__():
    # Get the message and key, and encrypt it using the Vigenère encryption function
    message = input("Enter the message: ").upper()
    key = input("Enter the key: ").upper()
    print("Encrypted message:", vigenereEncryption(message, key))

__main__()
```

```
Encrypted message: BI3EKÑH
```

This segment of the Python program is designed for a brute-force attack on the Vigenère cipher without knowing the key. It includes functions to load a database of artists from a CSV file, decrypt a message using the Vigenère cipher, and perform a brute-force attack. Here is a summary of the key functionalities:

1. `loadCSVFile(csv_file)`: Loads a database of 3,000 artists from a CSV file, removes spaces and converts names to uppercase, and returns a set of artists.
2. `vigenereDecryption(encrypted_message, key)`: Decrypts a message using the Vigenère cipher with a specified key. It repeats the key to match the message length and shifts characters accordingly.
3. `bruteForceAttack(encrypted_message, top_artists, full_mode)`: Performs a brute-force attack on the Vigenère cipher. It iterates through all possible keys and decrypts the message. The function then checks for matches with 3,000 known artists, printing the key and decrypted message if a match is found.

The program aims to decrypt an encrypted message representing an artist's name by trying all possible keys and comparing the results with a list of 3,000 known artists. It provides options for decrypting either the full artist name or the first few characters, depending on the length of the encrypted message.

In [ ]:
```
# Import libraries used
import csv
from unidecode import unidecode

def loadCSVFile(csv_file):
    """
    Loads a database of artists from a CSV file and returns a set of artists.

    Args:
        csv_file (str): The path to the CSV file.

    Returns:
        set: A set containing the names of artists.
    """

    # Initialize an empty set to store artist names
    top_artists = set()

    try:
        # Open the CSV file in read mode with UTF-8 encoding
        with open(csv_file, 'r', encoding='utf-8') as file:
            csv_reader = csv.reader(file)
            # Iterate through each row in the CSV file
            for row in csv_reader:
                # Remove spaces and convert to uppercase using unidecode for each artist name
                artist = unidecode(row[0].replace(" ", "")).upper()
                top_artists.add(artist)

    # Handle exceptions
    except FileNotFoundError:
        print(f"File '{csv_file}' was not found.")
    except Exception as e:
        print(f"Error loading CSV file: {e}")

    return top_artists


def vigenereDecryption(encrypted_message, key):
    """
    Decrypts a message using the Vigenère cipher.

    Args:
        encrypted_message (str): The encrypted message.
        key (str): The key used for decryption.

    Returns:
        str: The decrypted message.
    """

    # Define the extended alphabet used for encryption
    extended_alphabet = "ABCDEFGHIJKLMNÑOPQRSTUVWXYZ0123456789"
    alphabet_indices = {char: idx for idx, char in enumerate(extended_alphabet)}

    # Repeat the key to match the length of the encrypted message
    key = key * (len(encrypted_message) // len(key)) + key[:len(encrypted_message) % len(key)]
    decrypted_message = [] # Empty list to store the decrypted chars

    for i in range(len(encrypted_message)):
        # Check if the character is an alphabet letter or a digit
        if encrypted_message[i].isalpha() or encrypted_message[i].isdigit():
            # Calculate the index of the decrypted character and add it to the decrypted message
            idx = alphabet_indices[encrypted_message[i]] - int(key[i])
            decrypted_message.append(extended_alphabet[idx])
        # If the character is not an alphabet letter or a digit, keep it unchanged
        else:
            decrypted_message.append(encrypted_message[i])

    return ''.join(decrypted_message)


def bruteForceAttack(encrypted_message, top_artists, full_mode):
    """
    Performs a brute-force attack on the Vigenère cipher.

    Args:
        encrypted_message (str): The encrypted message.
        top_artists (set): A set of known artists' names.
        full_mode (bool): True for full artist name, False for first characters.

    Returns:
        int: 1 if a match is found, 0 otherwise.
    """

    # Initialize a variable to track if a boolean coincidence is found
    coincidence = 0

    for digit in range(10**len(encrypted_message)):
        # Convert the digit to a string and pad with zeros to match the length of the encrypted message
        key_str = str(digit).zfill(len(encrypted_message))
        # Decrypt the message using the Vigenère decryption function
        decrypted_message = vigenereDecryption(encrypted_message, key_str)

        # Check the mode (full or first characters)
        if full_mode:
            # Check if the decrypted message is in the list of top artists
            if decrypted_message in top_artists:
                print(f"Key: {key_str}, Decrypted Message: {decrypted_message}")
                coincidence = 1
        else:
            # Find artists that start with the decrypted message
            matching_artists = [artist for artist in top_artists if artist.startswith(decrypted_message)]

            # Print the key and matching artists if there are any
            for artist in matching_artists:
                print(f"Key: {key_str}, Decrypted Message: {artist}")

            if matching_artists: coincidence = 1

    return coincidence


def __main__():
    # Load the list of top artists from the CSV file
    top_artists = loadCSVFile('3000artists.csv')

    # Display a welcome message and explain the program modes
    print("Welcome to the message decryptor program, which aims to decrypt the name of your favorite artist encrypted with the Vigenère cipher. \nThis program has 2 mod

    full_mode = 2 # True for full artist name / False for first chars.   1 / 0

    # Validate user input for the mode
    while full_mode not in {0,1}:
        full_mode = int(input("\nPress '0' for the first mode, or press '1' for the second one: "))
        if full_mode not in {0,1}: print("Invalid Option. Try again.")

    # Get the encrypted message from the user and convert it to uppercase
    encrypted_message = input("Enter the encrypted message: ").upper()

    # Check if a match is found using the brute-force attack function
    if not bruteForceAttack(encrypted_message, top_artists, full_mode):
        print("No match found... Try again.")

__main__()
```

```
Welcome to the message decryptor program, which aims to decrypt the name of your favorite artist encrypted with the Vigenère cipher.
This program has 2 modes:
        Option 0.- The first 4 or 5 letters of your artist.
                   We recommend this variant when the length of the encrypted message is greater than or equal to 9 characters.
        Option 1.- Full name of your artist.
                   We recommend this variant when the length of the encrypted message is less than 9 characters.
Key: 671976, Decrypted Message: WAYLONJENNINGS
Key: 971971, Decrypted Message: TAYLORSWIFT
```