IS LAB_1

~ Prof. Dr. Aashka Raval

✅ **Title 1 : Caesar Cipher Encryption in Python**.

📝 Objective:

To implement a Caesar Cipher encryption algorithm in Python that converts a plain text input into an encrypted (cipher) text using a key value.

📖 Introduction:

The **Caesar Cipher** is one of the simplest and most widely known encryption techniques. It is a type of **substitution cipher** where each letter in the plain text is shifted by a fixed number of positions (called the **key**) down the alphabet.

For example, with a shift of 3:

- `'a'` becomes `'d'`
- `'b'` becomes `'e'`
- `'z'` wraps around and becomes `'c'`

Non-alphabetic characters like numbers, spaces, and symbols remain unchanged in the encrypted text.

🔍 Concepts Used:

- ASCII conversion using `ord()` and `chr()`
- Modulo operator `%` for wrapping around the alphabet
- String manipulation
- Conditional checking using `isalpha()` to process only alphabet characters

---

🧠 Logic:

1. Loop through each character in the plain text.
2. If it is a letter:
    - Convert it to lowercase.
    - Convert it to its ASCII value using `ord()`.
    - Apply the Caesar shift formula:
      `(ord(char) - 97 + key) % 26 + 97`
    - Convert back to character using `chr()` and append to the cipher string.

3. If it is not a letter (like number, symbol, or space), append it unchanged.
4. Return the complete cipher

💻 Python Code:

```
def encryptCipher(plainText, key) :
    cipher = ""

    for ch in plainText:
        if ch.isalpha():
            encrypted = chr(((ord(ch.lower()) - 97 + int(key)) %
26) + 97)
            cipher += encrypted

        else : # to deal with special char and nums.
            cipher += ch

    return cipher

plainText = input("Enter the plain Text here : ")
key = input("Enter the integer key Value here : ")
cipherText = encryptCipher(plainText, key)
print("Cipher Text:", cipherText)
```

🧪 Sample Output:

```
Enter the plain Text here : Jay Swaminarayan
Enter the integer key Value here : 3
Cipher Text: mdb vzdplqdudbdq
PS D:\Ghanu Study\Sem-5> python -u "d:\Ghanu
Enter the plain Text here : PDPU
Enter the integer key Value here : 4
Cipher Text: thty
```

✅ Conclusion:

This lab helped me understand how classical encryption methods like Caesar Cipher work. I also learned how to manipulate character values using ASCII in Python and apply logic to encrypt messages securely and logically.

💻 Lab Title 2: Enhanced Caesar Cipher with Advanced Features and Menu Interface

---

🔍 Objective:

To implement an advanced Caesar Cipher program in Python that includes encryption, decryption, random key generation, selective encryption, and logging—all managed through a user-friendly menu system.

📚 Theory:

The **Caesar Cipher** is one of the earliest known encryption techniques, in which each letter in the plaintext is shifted a fixed number of places down the alphabet. For example, with a shift of 3, A would be replaced by D, B would become E, etc.

Enhancements in This Program:

- **Random Key Generation**: The program can auto-generate a random key for secure encryption.
- **Case Preservation**: It preserves uppercase and lowercase letters in their original forms.
- **Selective Encryption**: Encrypt only specific words in the text like names or secrets.
- **Logging**: Stores a history of all encryption/decryption operations with timestamps.
- **Decryption**: Reverses the encryption using the same key.
- **Menu-based Interface**: Allows users to navigate options like encrypt, decrypt, view history, etc.

🧠 Algorithm:

1. **Encryption**:
    - Loop through each character in the input.
    - If the character is alphabetical:
        - Shift it by the key value.
        - Preserve the case (uppercase/lowercase).
    - If selective encryption is enabled, apply encryption only to selected words.
2. **Decryption**:
    - Reverse the shift using the same key.
    - Preserve case.
3. **Logging**:
    - Store each action in `cipher_history.log` with timestamp, key, and input/output.
4. **Menu Interface**:
    - Present user with options: Encrypt, Decrypt, View History, Exit.

📜 Python Code:

```
import random
from datetime import datetime


def log_action(action, key, input_text, output_text):

    with open("cipher_history.log", "a") as log_file:
        log_file.write(f"{datetime.now()} | {action} | Key: {key} | Input:
{input_text} | Output: {output_text}\n")

def encrypt_char(ch, key):
    if ch.islower():
        return chr((ord(ch) - ord('a') + key) % 26 + ord('a'))
    elif ch.isupper():
        return chr((ord(ch) - ord('A') + key) % 26 + ord('A'))
    else:
        return ch

def decrypt_char(ch, key):
    if ch.islower():
        return chr((ord(ch) - ord('a') - key) % 26 + ord('a'))
    elif ch.isupper():
        return chr((ord(ch) - ord('A') - key) % 26 + ord('A'))
    else:
        return ch

def encrypt_text(text, key):
    return ''.join(encrypt_char(ch, key) for ch in text)

def decrypt_text(text, key):
    return ''.join(decrypt_char(ch, key) for ch in text)

def selective_encrypt(text, key, words_to_encrypt):
    words = text.split()
    encrypted_words = []
    for word in words:
        if word in words_to_encrypt:
            encrypted_words.append(encrypt_text(word, key))
        else:
            encrypted_words.append(word)
    return ' '.join(encrypted_words)



def show_menu():
    print("\n----- Caesar Cipher Menu -----")
    print("1. Encrypt full text with manual key")
```

```
    print("2. Encrypt full text with random key")
    print("3. Encrypt only specific words (selective encryption)")
    print("4. Decrypt text with known key")
    print("5. View log history")
    print("6. Exit")

while True:

    show_menu()
    choice = input("Choose an option: ")

    if choice == '1':
        plain_text = input("Enter the text to encrypt: ")
        key = int(input("Enter the key (1-25): "))
        cipher_text = encrypt_text(plain_text, key)
        log_action("Manual Encrypt", key, plain_text, cipher_text)
        print("Encrypted Text:", cipher_text)

    elif choice == '2':
        plain_text = input("Enter the text to encrypt: ")
        key = random.randint(1, 25)
        cipher_text = encrypt_text(plain_text, key)
        log_action("Random Encrypt", key, plain_text, cipher_text)
        print(f"Encrypted Text: {cipher_text} (Key: {key})")

    elif choice == '3':
        plain_text = input("Enter the full sentence: ")
        key = int(input("Enter the key (1-25): "))
        specific_words = input("Enter the words to encrypt (comma separated):
").split(',')
        specific_words = [w.strip() for w in specific_words]
        cipher_text = selective_encrypt(plain_text, key, specific_words)
        log_action("Selective Encrypt", key, plain_text, cipher_text)
        print("Encrypted Text:", cipher_text)

    elif choice == '4':
        cipher_text = input("Enter the text to decrypt: ")
        key = int(input("Enter the key used during encryption: "))
        decrypted_text = decrypt_text(cipher_text, key)
        log_action("Decrypt", key, cipher_text, decrypted_text)
        print("Decrypted Text:", decrypted_text)

    elif choice == '5':
        print("\n--- Cipher History Log ---")
        try:
            with open("cipher_history.log", "r") as log_file:
                print(log_file.read())
        except FileNotFoundError:
            print("No log history found.")

    elif choice == '6':
```

```
        print("Exited.")
        break

    else:
        print("Invalid choice. Please try again.")
```

🧪 Sample Output:

```
----- Caesar Cipher Menu -----
1. Encrypt full text with manual key
2. Encrypt full text with random key
3. Encrypt only specific words (selective encryption)
4. Decrypt text with known key
5. View log history
6. Exit
Choose an option: 3
Enter the full sentence: chetan gadhiya is anadi mukta
Enter the key (1-25): 3
Enter the words to encrypt (comma separated): chetan, mukta
Encrypted Text: fkhwdq gadhiya is anadi pxnwd

----- Caesar Cipher Menu -----
1. Encrypt full text with manual key
2. Encrypt full text with random key
3. Encrypt only specific words (selective encryption)
4. Decrypt text with known key
5. View log history
6. Exit
Choose an option: 5

--- Cipher History Log ---
2025-07-22 11:18:56.292964 | Selective Encrypt | Key: 3 | Input: chetan gadhiya is anadi mukta | Output: chetan gadhiya is anadi mukta
2025-07-22 11:19:29.958200 | Selective Encrypt | Key: 3 | Input: chetan gadhiya is anadi mukta | Output: fkhwdq gadhiya is anadi pxnwd
```

📌 Conclusion:

This program demonstrates how classical ciphers can be expanded with modern features like randomization, case handling, user interactivity, and logging. These enhancements make it more practical and educational for understanding basic cryptography concepts.