# CYBERSECURITY INTERNSHIP

Saurav Kumar
email: sauravsingh9708558744@gmail.com

## INTERNSHIP TASK 01

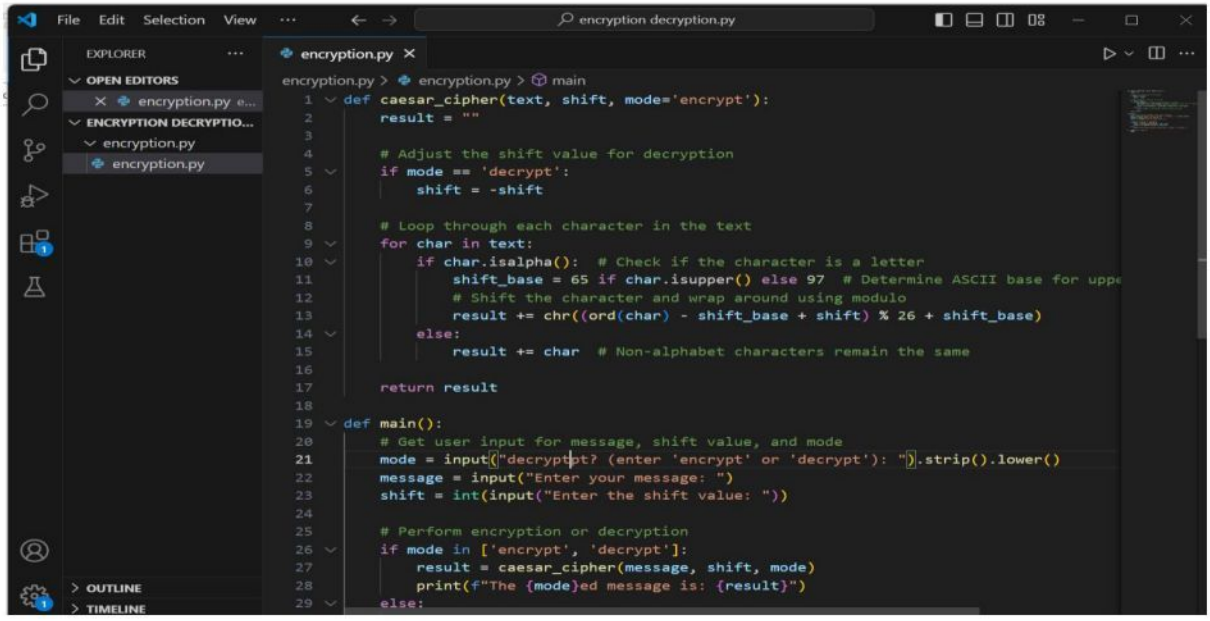
create a python program that can encrypt or decrypt text using the Caesar cipher algorithm. Allow user to input a message and a shift value to perform encryption and decryption.

## CODE:

```
def caesar_cipher(text, shift, mode='encrypt'):
    result = ""

    # Adjust the shift value for decryption
    if mode == 'decrypt':
        shift = -shift

    # Loop through each character in the text
    for char in text:
        if char.isalpha():  # Check if the character is a letter
            shift_base = 65 if char.isupper() else 97  # Determine ASCII base for
uppercase/lowercase
            # Shift the character and wrap around using modulo
            result += chr((ord(char) - shift_base + shift) % 26 + shift_base)
        else:
            result += char  # Non-alphabet characters remain the same

    return result
```

```
def main():
    # Get user input for message, shift value, and mode
    mode = input("Do you want to encrypt or decrypt? (enter 'encrypt' or 'decrypt'): ").strip().lower()
    message = input("Enter your message: ")
    shift = int(input("Enter the shift value: "))

    # Perform encryption or decryption
    if mode in ['encrypt', 'decrypt']:
        result = caesar_cipher(message, shift, mode)
        print(f"The {mode}ed message is: {result}")
    else:
        print("Invalid mode selected. Please enter 'encrypt' or 'decrypt'.")

if __name__ == "__main__":
    main()
```

## SCREENSHOT:



```python
def caesar_cipher(text, shift, mode='encrypt'):
        else:
            result += char  # Non-alphabet characters remain the same

    return result

def main():
    # Get user input for message, shift value, and mode
    mode = input("decryptpt? (enter 'encrypt' or 'decrypt'): ").strip().lower()
    message = input("Enter your message: ")
    shift = int(input("Enter the shift value: "))

    # Perform encryption or decryption
    if mode in ['encrypt', 'decrypt']:
        result = caesar_cipher(message, shift, mode)
        print(f"The {mode}ed message is: {result}")
    else:
        print("Invalid mode selected. Please enter 'encrypt' or 'decrypt'.")

if __name__ == "__main__":
    main()
```

## OUTPUT SCREENSHOT:

```python
def caesar_cipher(text, shift, mode='encrypt'):
        else:
            result += char  # Non-alphabet characters remain the same

    return result

def main():
    # Get user input for message, shift value, and mode
    mode = input("decryptdecryptpt? (enter 'encrypt' or 'decrypt'): ").strip().lower()
    message = input("Enter your message: ")
    shift = int(input("Enter the shift value: "))

    # Perform encryption or decryption
    if mode in ['encrypt', 'decrypt']:
        result = caesar_cipher(message, shift, mode)
        print(f"The {mode}ed message is: {result}")
    else:
        print("Invalid mode selected. Please enter 'encrypt' or 'decrypt'.")
```

```
PS D:\encryption decryption.py> & C:/Users/khush/AppData/Local/Programs/Python/Python312/python.exe "d:/
encryption decryption.py/encryption.py/encryption.py"
decryptpt? (enter 'encrypt' or 'decrypt'): decrypt
Enter your message: khushi
Enter the shift value: 6
The decrypted message is: ebombc
PS D:\encryption decryption.py> & C:/Users/khush/AppData/Local/Programs/Python/Python312/python.exe "d:/
```

## Task:-3

1. **Build a tool that assesses the strength of a password based on criteria such as length, presence of uppercase and lowercase letters, numbers, and special characters. Provide feedback to users on the password's strength.**

```
import tkinter as tk
from tkinter import messagebox


# Function to check password complexity
def check_password_strength(password):
    length = len(password)
    has_upper = any(c.isupper() for c in password)
    has_lower = any(c.islower() for c in password)
    has_digit = any(c.isdigit() for c in password)
    has_special = any(c in "!@#$%^&*()-_+=" for c in password)

    strength_criteria = {
        'Length': length >= 8,
        'Uppercase Letter': has_upper,
        'Lowercase Letter': has_lower,
        'Digit': has_digit,
        'Special Character': has_special
    }

    strength = sum(strength_criteria.values())

    if strength == 5:
        return "Strong", strength_criteria
    elif strength >= 3:
        return "Medium", strength_criteria
    else:
        return "Weak", strength_criteria


# Function to update feedback
def update_feedback(event):
    password = password_entry.get()
    strength, criteria = check_password_strength(password)
    feedback = f"Password Strength: {strength}\n"
    for criterion, met in criteria.items():
        feedback += f"{criterion}: {'✓' if met else '✗'}\n"
    feedback_label.config(text=feedback, fg=color_map[strength])


# Function to copy password to clipboard
```

```
def copy_to_clipboard():
    root.clipboard_clear()
    root.clipboard_append(password_entry.get())
    messagebox.showinfo("Password Complexity Checker", "Password copied to clipboard!")



# GUI setup
root = tk.Tk()
root.title("Password Complexity Checker")
root.geometry("400x350")
root.config(bg="#1a1a1a")

# Color map for feedback
color_map = {
    "Strong": "#39ff14",
    "Medium": "#ffd700",
    "Weak": "#ff073a"
}

# Labels and entry
title_label = tk.Label(root, text="Enter Password:", font=("Helvetica", 14), fg="#39ff14", bg="#1a1a1a")
title_label.pack(pady=10)

password_entry = tk.Entry(root, show="*", font=("Helvetica", 14), fg="#39ff14", bg="#333333",
                          insertbackground="#39ff14")
password_entry.pack(pady=10)
password_entry.bind("<KeyRelease>", update_feedback)

feedback_label = tk.Label(root, text="Password Strength: ", font=("Helvetica", 14), fg="#39ff14", bg="#1a1a1a",
                          justify=tk.LEFT)
feedback_label.pack(pady=10)

# Copy button
copy_button = tk.Button(root, text="Copy Password", command=copy_to_clipboard, font=("Helvetica", 12), fg="#39ff14",
                        bg="#333333", activebackground="#555555", activeforeground="#39ff14")
copy_button.pack(pady=10)

# Run the application
root.mainloop()
```