ASSIGNMENT-3

Python Programming Assignment: "Secret Code Generator"

Objective: Get creative with Python by building a fun secret code generator that can convert regular text into a coded message and decode it back. This will help you practice using functions, loops, string manipulation, and dictionaries.

Q\n : Create a Secret Code Generator Your task is to write a Python program that can:

- Encode a message by shifting the letters in the alphabet (similar to a Caesar cipher).
- Decode a message back to its original form.
- Allow the user to choose between encoding or decoding.

Sol : # ----------

```
# SECRET CODE GENERATOR (Caesar Cipher)
# ----------

# This function shifts a single character by the shift amount.
# It keeps uppercase/lowercase letters the same and ignores non-letters.
def shift_char(char, shift):
    if char.isalpha():
        # Determine if character is uppercase or lowercase
        base = ord('A') if char.isupper() else ord('a')

        # Convert character to 0–25 range, shift it, wrap using modulo 26
        shifted = (ord(char) - base + shift) % 26

        # Convert back to character
        return chr(base + shifted)
    else:
        # Leave non-letters unchanged
        return char
```

```
# ----------------------------------------
# Function to encode a message
# ----------------------------------------
def encode_message(message, shift):
    encoded = ""
    for char in message:
        encoded += shift_char(char, shift)
    return encoded

# ----------------------------------------
# Function to decode a message
# ----------------------------------------
def decode_message(message, shift):
    # Decoding is just encoding with negative shift
    return encode_message(message, -shift)

# ----------------------------------------
# Function to display the menu and handle user choices
# ----------------------------------------
def menu():
    while True:
        print("\n--- Secret Code Generator ---")
        print("1. Encode a message")
        print("2. Decode a message")
        print("3. Exit")

        choice = input("Choose an option (1, 2, or 3): ")

        if choice == "1":
            message = input("Enter a message to encode: ")
```

```
    # Validate shift
    try:
        shift = int(input("Enter shift number (e.g., 2): "))
    except ValueError:
        print("Invalid shift! Please enter a number.")
        continue

    print("\nEncoded Message:")
    print(encode_message(message, shift))

elif choice == "2":
    message = input("Enter a message to decode: ")

    # Validate shift
    try:
        shift = int(input("Enter shift number used to encode: "))
    except ValueError:
        print("Invalid shift! Please enter a number.")
        continue

    print("\nDecoded Message:")
    print(decode_message(message, shift))

elif choice == "3":
    print("Goodbye! Program exited.")
    break

else:
    print("Invalid choice. Please enter 1, 2, or 3.")
```

```
# ----------------------------------------

# MAIN PROGRAM START

# ----------------------------------------

menu()
```

Outputs :

main.py

```
# ----------------------------------------
# SECRET CODE GENERATOR (Caesar Cipher)
# ----------------------------------------

# This function shifts a single character by the shift amount.
# It keeps uppercase/lowercase letters the same and ignores non-letters.
def shift_char(char, shift):
    if char.isalpha():
        # Determine if character is uppercase or lowercase
        base = ord('A') if char.isupper() else ord('a')

        # Convert character to 0-25 range, shift it, wrap using modulo 26
        shifted = (ord(char) - base + shift) % 26

        # Convert back to character
        return chr(base + shifted)
    else:
        # Leave non-letters unchanged
        return char


# ----------------------------------------
# Function to encode a message
# ----------------------------------------
def encode_message(message, shift):
    encoded = ""
    for char in message:
        encoded += shift_char(char, shift)
    return encoded


# ----------------------------------------
# Function to decode a message
# ----------------------------------------
def decode_message(message, shift):
    # Decoding is just encoding with negative shift
    return encode_message(message, -shift)


# ----------------------------------------
# Function to display the menu and handle user choices
# ----------------------------------------
def menu():
    while True:
        print("\n--- Secret Code Generator ---")
        print("1. Encode a message")
        print("2. Decode a message")
        print("3. Exit")

        choice = input("Choose an option (1, 2, or 3): ")

        if choice == "1":
            message = input("Enter a message to encode: ")

            # Validate shift
            try:
                shift = int(input("Enter shift number (e.g., 2): "))
            except ValueError:
                print("Invalid shift! Please enter a number.")
                continue

            print("\nEncoded Message:")
            print(encode_message(message, shift))
```

Output

```
--- Secret Code Generator ---
1. Encode a message
2. Decode a message
3. Exit
Choose an option (1, 2, or 3): 1
Enter a message to encode: HELLO WORLD
Enter shift number (e.g., 2): 2

Encoded Message:
JGNNQ YQTNF

--- Secret Code Generator ---
1. Encode a message
2. Decode a message
3. Exit
Choose an option (1, 2, or 3): 1
Enter a message to encode: COLGATE FOR CONFIDENCE
Enter shift number (e.g., 2): 3

Encoded Message:
FROJDWH IRU FRQILGHQFH

--- Secret Code Generator ---
1. Encode a message
2. Decode a message
3. Exit
Choose an option (1, 2, or 3): 1
Enter a message to encode: BYE
Enter shift number (e.g., 2): 2
```

main.py

```
def decode_message(message, shift):
    # Decoding is just encoding with negative shift
    return encode_message(message, -shift)


# ----------------------------------------
# Function to display the menu and handle user choices
# ----------------------------------------
def menu():
    while True:
        print("\n--- Secret Code Generator ---")
        print("1. Encode a message")
        print("2. Decode a message")
        print("3. Exit")

        choice = input("Choose an option (1, 2, or 3): ")

        if choice == "1":
            message = input("Enter a message to encode: ")

            # Validate shift
            try:
                shift = int(input("Enter shift number (e.g., 2): "))
            except ValueError:
                print("Invalid shift! Please enter a number.")
                continue

            print("\nEncoded Message:")
            print(encode_message(message, shift))
```

Output

```
Enter a message to encode: HELLO WORLD
Enter shift number (e.g., 2): 2

Encoded Message:
JGNNQ YQTNF

--- Secret Code Generator ---
1. Encode a message
2. Decode a message
3. Exit
Choose an option (1, 2, or 3): 1
Enter a message to encode: COLGATE FOR CONFIDENCE
Enter shift number (e.g., 2): 3

Encoded Message:
FROJDWH IRU FRQILGHQFH

--- Secret Code Generator ---
1. Encode a message
2. Decode a message
3. Exit
Choose an option (1, 2, or 3): 1
Enter a message to encode: BYE
Enter shift number (e.g., 2): 2

Encoded Message:
DAG

--- Secret Code Generator ---
1. Encode a message
2. Decode a message
```

Project Explanation :

1. Project Overview

The Secret Code Generator is a simple Python project that demonstrates how to **encode and decode messages** using the **Caesar cipher technique**.

- Encoding means converting a readable message into a secret code.
- Decoding means converting the secret code back into the original message. This project helps understand basic cryptography concepts like substitution ciphers and modular arithmetic.

2. How It Works

- The program uses a **shift value** (an integer) to move letters forward or backward in the alphabet.
- Example: With a shift of 3,
  - A → D, B → E, C → F, …
  - HELLO → KHOOR
- To decode, the program shifts letters back by the same number.
  - KHOOR → HELLO

3. Main Components

1. **Character Shifting Function (**shift_char**)**
   - Handles individual letters.
   - Wraps around the alphabet using modulo 26.
   - Leaves non-letters (spaces, numbers, punctuation) unchanged.
2. **Encoding Function (**encode_message**)**
   - Loops through each character in the message.
   - Applies the shift to produce the encoded message.
3. **Decoding Function (**decode_message**)**
   - Calls the encoding function with a **negative shift**.
   - Restores the original message.
4. **Menu System (**menu**)**
   - Provides options:
     - Encode a message
     - Decode a message
     - Exit the program

- Uses input() to interact with the user.

4. Features

- Works with both uppercase and lowercase letters.
- Preserves spaces, numbers, and punctuation.
- Validates user input for shift values.
- Interactive menu for easy use.

5. Example Run

Code

--- Secret Code Generator ---

1. Encode a message

2. Decode a message

3. Exit

Choose an option (1, 2, or 3): 1

Enter a message to encode: HELLO WORLD

Enter shift number (e.g., 2): 3

Encoded Message:

KHOOR ZRUOG

Then decoding:

Code

Enter a message to decode: KHOOR ZRUOG

Enter shift number used to encode: 3

Decoded Message:

HELLO WORLD

6. Learning Outcomes

- Understand the basics of **cryptography** and **Caesar cipher**.
- Practice **string manipulation** and **loops** in Python.
- Learn how to build an **interactive menu-driven program**.
- Gain experience with **error handling** using try-except.

- This project is a beginner-friendly way to explore how secret codes work. It teaches the fundamentals of encoding/decoding, user interaction, and modular arithmetic in Python.
- The project highlights that the security of a Caesar cipher depends entirely on the secrecy of the shift number. Without it, decoding becomes trivial — teaching the limitation of simple ciphers.
- This project is a great teaching tool for beginners to learn about both programming and cryptography. It bridges theory (Caesar cipher) with practice (Python implementation).
- The program can be extended to support more complex ciphers (like Vigenère or Base64), automatic brute-force decoding, or even integration with file I/O to encode/decode text files.
- In short: This project is not just a fun way to hide and reveal messages, but also a stepping stone into **cryptography, programming best practices, and user interface design**.