Assignment -2

(By MEGAVARSHINI A)

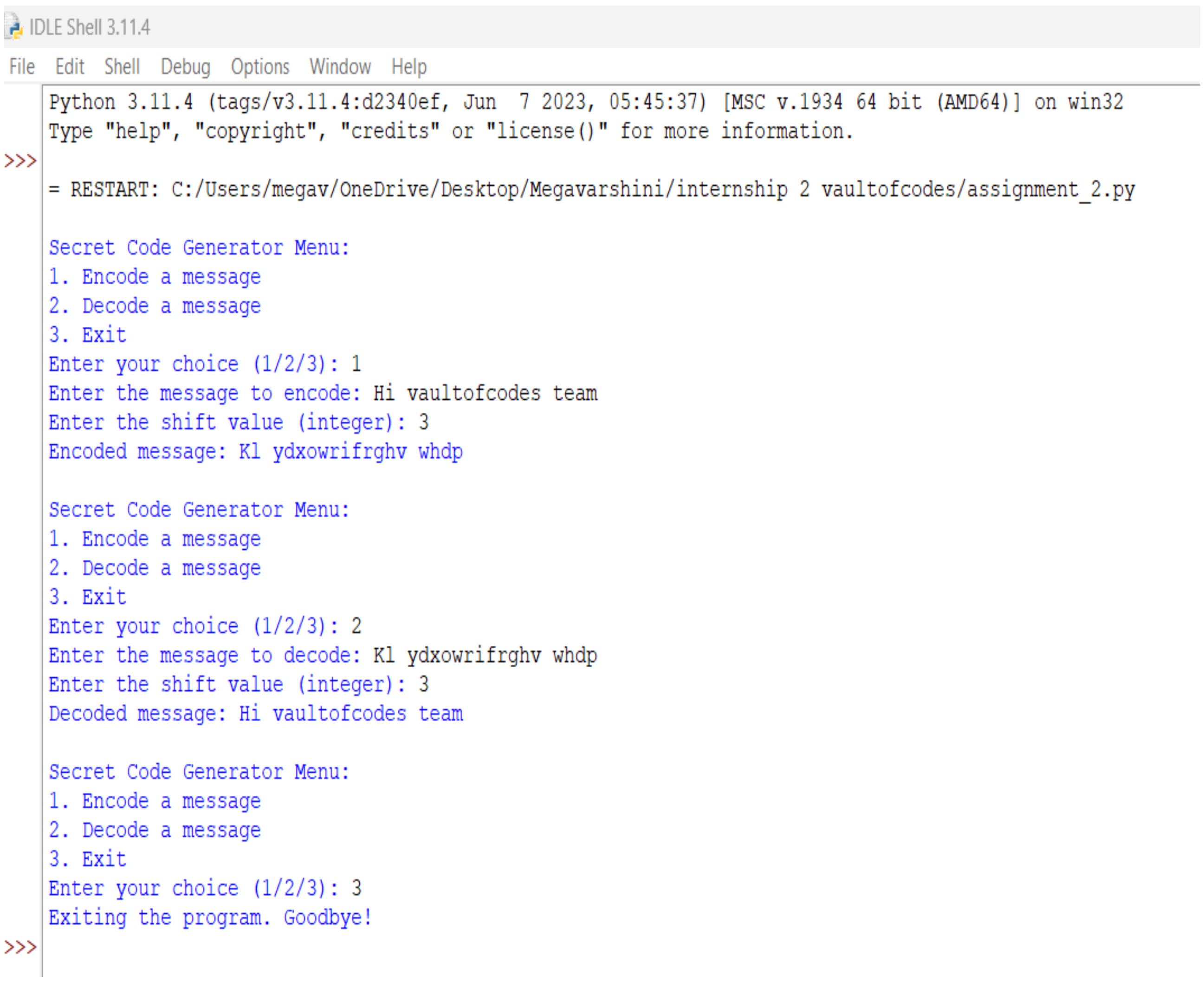
Secrete Code Generator

SOURCE CODE:

```
# Function to encode a message
def encode_message(message, shift):
    encoded_message = ""
    for char in message:
        if char.isalpha():
            shift_base = ord('A') if char.isupper() else ord('a')
            encoded_message += chr((ord(char) - shift_base + shift) % 26 +
shift_base)
        else:
            encoded_message += char  # Keep non-alphabetic characters unchanged
    return encoded_message

# Function to decode a message
def decode_message(message, shift):
    return encode_message(message, -shift)

# Function to display the menu and handle user choices
def menu():
    while True:
        print("\nSecret Code Generator Menu:")
print("1. Encode a message")
print("2. Decode a message")
print("3. Exit")
choice = input("Enter your choice (1/2/3): ")

if choice == "1":
    message = input("Enter the message to encode: ")
    try:
        shift = int(input("Enter the shift value (integer): "))
        print(f"Encoded message: {encode_message(message, shift)}")
    except ValueError:
        print("Invalid shift value. Please enter an integer.")

elif choice == "2":
    message = input("Enter the message to decode: ")
    try:
        shift = int(input("Enter the shift value (integer): "))
        print(f"Decoded message: {decode_message(message, shift)}")
    except ValueError:
        print("Invalid shift value. Please enter an integer.")

elif choice == "3":
    print("Exiting the program. Goodbye!")
    break

else:
```

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



# Main program execution
if __name__ == "__main__":
    menu()
```

OUTPUT SCREENSHOT:



```
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun  7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/megav/OneDrive/Desktop/Megavarshini/internship 2 vaultofcodes/assignment_2.py

Secret Code Generator Menu:
1. Encode a message
2. Decode a message
3. Exit
Enter your choice (1/2/3): 1
Enter the message to encode: Hi vaultofcodes team
Enter the shift value (integer): 3
Encoded message: Kl ydxowrifrghv whdp

Secret Code Generator Menu:
1. Encode a message
2. Decode a message
3. Exit
Enter your choice (1/2/3): 2
Enter the message to decode: Kl ydxowrifrghv whdp
Enter the shift value (integer): 3
Decoded message: Hi vaultofcodes team

Secret Code Generator Menu:
1. Encode a message
2. Decode a message
3. Exit
Enter your choice (1/2/3): 3
Exiting the program. Goodbye!
>>>
```

EXPLANATION:

The above code implements a **Secret Code Generator** using a Caesar cipher, a simple encryption technique that shifts the letters of a message by a specified number. The program consists of three main parts: **encoding**, **decoding**, and a **user menu**. The encoding function takes a message and a shift value, then moves each letter forward in the alphabet by the shift amount, wrapping around from Z to A when needed. Similarly, the decoding function shifts the letters backward by reversing the shift value. Both functions ignore spaces, numbers, and special characters, leaving them unchanged. A menu allows the user to choose whether to encode a message, decode a message, or exit the program. The program handles invalid inputs, such as non-integer shift values, gracefully. By breaking the logic into modular functions, the code is clean, reusable, and easy to understand, making it user-friendly and robust for encoding and decoding messages.

Algorithm:

Step 1: Import Necessary Libraries

No external libraries are required since the program uses built-in Python functions like ord() and chr().

Step 2: Define the encode_message Function

1. Accepts a message and a shift value as inputs.
2. Loops through each character in the message:
   - If the character is a letter (uppercase or lowercase), it shifts it forward in the alphabet using modular arithmetic.
   - If the character is not a letter (like spaces or punctuation), it remains unchanged.
3. Returns the encoded (shifted) message.

Step 3: Define the decode_message Function

1. Uses the same logic as encode_message but shifts letters backward by reversing the shift value (using -shift).
2. Calls the encode_message function with the negative shift for reusability.

Step 4: Define the menu Function

1. Displays a menu with three options:
    - **Encode a message**: Asks the user for a message and a shift value, then calls encode_message to display the encoded result.
    - **Decode a message**: Asks the user for an encoded message and a shift value, then calls decode_message to display the original message.
    - **Exit the program**: Ends the program.
2. Handles invalid inputs, such as non-integer shift values or incorrect menu choices, by displaying appropriate error messages.
3. Repeats the menu until the user chooses to exit.

Step 5: Execute the Program

1. The if __name__ == "__main__": block runs the menu function to start the program.
2. The program loops until the user selects the "Exit" option.