

Encode-Decode.py

```
1 class Encoder:
2     def __init__(self, shift):
3         """
4         Initialize the Encoder class with a shift value.
5
6         Args:
7             shift (int): The number of positions to shift the characters.
8         """
9         self.shift = shift
10
11    def encode(self, message):
12        """
13        Encode the message using the Caesar Cipher algorithm.
14
15        Args:
16            message (str): The message to be encoded.
17
18        Returns:
19            str: The encoded message.
20        """
21        encoded_message = ""
22        for char in message:
23            if char.isalpha():
24                ascii_offset = 65 if char.isupper() else 97
25                encoded_char = chr((ord(char) - ascii_offset + self.shift) % 26 +
ascii_offset)
26                encoded_message += encoded_char
27            else:
28                encoded_message += char
29        return encoded_message
30
31
32    class Decoder:
33        def __init__(self, shift):
34            """
35            Initialize the Decoder class with a shift value.
36
37            Args:
38                shift (int): The number of positions to shift the characters.
39            """
40            self.shift = shift
41
42        def decode(self, encoded_message):
43            """
44            Decode the encoded message using the Caesar Cipher algorithm.
45
46            Args:
47                encoded_message (str): The encoded message to be decoded.
48
49            Returns:
50                str: The decoded message.
51            """
```

```
52     decoded_message = ""
53     for char in encoded_message:
54         if char.isalpha():
55             ascii_offset = 65 if char.isupper() else 97
56             decoded_char = chr((ord(char) - ascii_offset - self.shift) % 26 +
ascii_offset)
57             decoded_message += decoded_char
58         else:
59             decoded_message += char
60     return decoded_message
61
62
63 def main():
64     while True:
65         print("Message Encoder and Decoder")
66         print("-----")
67         print("1. Encode a message")
68         print("2. Decode a message")
69         print("3. Quit")
70         choice = input("Enter your choice: ")
71
72         if choice == "1":
73             message = input("Enter the message to encode: ")
74             shift = int(input("Enter the shift value: "))
75             encoder = Encoder(shift)
76             encoded_message = encoder.encode(message)
77             print(f"Encoded Message: {encoded_message}")
78         elif choice == "2":
79             encoded_message = input("Enter the encoded message to decode: ")
80             shift = int(input("Enter the shift value: "))
81             decoder = Decoder(shift)
82             decoded_message = decoder.decode(encoded_message)
83             print(f"Decoded Message: {decoded_message}")
84         elif choice == "3":
85             break
86         else:
87             print("Invalid choice. Please try again.")
88
89
90 if __name__ == "__main__":
91     main()
```

OUTPUT

```
Output Clear
Message Encoder and Decoder
-----
1. Encode a message
2. Decode a message
3. Quit
Enter your choice: 1
Enter the message to encode: Hello
Enter the shift value: 1
Encoded Message: Ifmmp
```

```
Message Encoder and Decoder
-----
1. Encode a message
2. Decode a message
3. Quit
Enter your choice: 2
Enter the encoded message to decode: Ifmmp
Enter the shift value: 1
Decoded Message: Hello
```

```
Message Encoder and Decoder
-----
1. Encode a message
2. Decode a message
3. Quit
Enter your choice: 3

=== Code Execution Successful ===
```