The Rebel Cipher Algorithm

Welcome to my Rebel Cipher Algorithm which takes in an input of a string, encrypts it based on **your** given encryption key, and writes it to a file known as *encrypted_message.txt*

The Encryption Function

Here we convert the message into barebones raw value which we leave consistent throughout our code. This includes checking if the message is **Uppercase**, if it is **Lowercase** we use the `.upper()` method in order to convert the values into a UPPERCASE Value.

We the convert the encrypted characters (*the values*) into their respective **ASCII** values, by using the `chr((ord(char) - 65 + key)%26 + 65)` function in order to convert it into the numerical ASCII value which we will convert later.

Then the function returns the value of the encrypted message

```
def encrypt(message, key):
   encrypted_message = ""
   for char in message:
     if char.isalpha():
      is_upper = char.isupper()
      char = char.upper()
      encrypted_char = chr((ord(char) - 65 + key) % 26 + 65)
      if not is_upper:
        encrypted_char = encrypted_char.lower()
      encrypted_message+=encrypted_char
     else:
      encrypted_message+=char
   return encrypted_message
```

The Decryption Function

The Decryption Function is fairly similar, it returns the encrypted value; except we have a **-key** value which basically flips the value of the key so that it is decrypted from the original key (if it added 5, the decryption subtracts 5)

```
def decrypt(encrypted_message, key):
  return encrypt(encrypted_message, -key)
```

The Saving Function

That was a funny name for a function... In this we have the function which saves the data to a file which we haven't specified. An important note: we are using the *"w"* function so that we are able to write to the given file.

```
def save_to_file(data, filename):
  with open(filename, "w") as file:
  file.write(data)
```

The Reading Function

We do a similar thing as above, except now we are only *"r"* or reading from the file. Since this is a simple code example, it is not really necessary for us to do this; but as a good programming practice, it is important to only use the file in the way you want it to be specified.

```
def read_from_file(filename):
  with open(filename, "r") as file:
  return file.read()
```

Main Loop

In this, we take the user input and ensure that it is not a number through the following

```
while not valid_message:
  message = input("Enter the Message> ")
  if any(char.isdigit() for char in message):
    print("This was an invalid input, please try entering a LETTER VALUE")
  else:
    valid_message = True
```

In here all we are doing is ensuring that we have a value that **ISN'T** a number through the *.isdigit()* method. If it is a number, we ask them to give another input; if it isn't a number, then we allow the programming to continue running by breaking out of the while loop by setting the Boolean Expression of the while loop to **TRUE**

Next, we are gathering all our various inputs

```
key = int(input("Enter the Encryption Key> "))

encrypted_message = encrypt(message, key)
decrypted_message = decrypt(encrypted_message, key)
```

And finally, we are printing out all our values; just converting them to Uppercase by using the *.upper()* function once again:

```
print(f"Your encrypted message is: {encrypted_message.upper()}")
print(f"Your decrypted message is: {decrypted_message.upper()}")

save_to_file(encrypted_message, "encrypted_message.txt")
```

```
encrypted_message_read_from_file = read_from_file("encrypted_message.txt")
print(f"The message read from the file is {encrypted_message_read_from_file.upper()}")
```