1. Write a python script to encrypt the string using Caesar cipher.

```python
def caesar_encrypt(plaintext, shift):
    ciphertext = ""
    for char in plaintext:
        if char.isalpha():
            shift_base = ord('A') if char.isupper() else ord('a')
            ciphertext += chr((ord(char) - shift_base + shift) % 26 + shift_base)
        else:
            ciphertext += char
    return ciphertext

plaintext = "KRUTHI"
shift = 3
ciphertext = caesar_encrypt(plaintext, shift)
print(f"Encrypted message: {ciphertext}")
```

output:

```
┌──(kruthi㉿kali)-[~]
└─$ python kru3.py
Encrypted message: NUXWKL
```

2. Write a Python script to Modify the above script to shift cipher based on user choice.

```python
def caesar_encrypt(plaintext, shift):
    ciphertext = ""
    for char in plaintext:
        if char.isalpha():
            shift_base = ord('A') if char.isupper() else ord('a')
            ciphertext += chr((ord(char) - shift_base + shift) % 26 + shift_base)
        else:
            ciphertext += char
    return ciphertext

# User input
plaintext = input("Enter plaintext: ")
shift = int(input("Enter shift value: "))

# Encrypt and display result
print("Encrypted message:", caesar_encrypt(plaintext, shift))
```

Output:

```
┌──(kruthi㉿kali)-[~]
└─$ python kru3.py
Enter plaintext: kruthi
Enter shift value: 2
Encrypted message: mtwvjk
```

3. Write a Python script to convert cipher text into uppercase characters and split the cipher into group of 5 of characters.

```
def format_ciphertext(ciphertext):
    ciphertext = ciphertext.upper().replace(" ", "")
    return ' '.join(ciphertext[i:i+5] for i in range(0, len(ciphertext), 5))
ciphertext = input("Enter the ciphertext: ")
print("Formatted ciphertext:", format_ciphertext(ciphertext))
```

Output:

```
┌──(kruthi㉿kali)-[~]
└─$ python kru3.py
Enter the ciphertext: kruthi
Formatted ciphertext: KRUTH I
```

4. Write a Python program to Find the histogram for each characters.

```
from collections import Counter
def character_histogram(text):
    histogram = Counter(text)
    return histogram
text = input("Enter the text: ")
histogram = character_histogram(text)
print("Character Histogram:")
for char, count in histogram.items():
    print(f"'{char}': {count}")
```

Output:

```
┌──(kruthi㉿kali)-[~]
└─$ python kru3.py
Enter the text: hii
Character Histogram:
'h': 1
'i': 2
```

5. Write a Python script to read the contents from the file.

```
def read_file(filename):
    with open(filename, 'r') as file:
        content = file.read()
    return content
filename = input("Enter the filename: ")
try:
    content = read_file(filename)
    print("File contents:")
    print(content)
except FileNotFoundError:
    print(f"Error: The file '{filename}' does not exist.")
```

Output:

```
┌──(kruthi㉿kali)-[~]
└─$ python kru3.py
Enter the filename: kruthi
File contents:
```

6. Write a Python script to encrypt the contents from the file.

```python
def caesar_encrypt(text, shift):
    encrypted = ""
    for char in text:
        if char.isalpha():
            base = ord('A') if char.isupper() else ord('a')
            encrypted += chr((ord(char) - base + shift) % 26 + base)
        else:
            encrypted += char
    return encrypted

def encrypt_file(input_file, output_file, shift):
    with open(input_file, 'r') as file:
        content = file.read()
    encrypted_content = caesar_encrypt(content, shift)
    with open(output_file, 'w') as file:
        file.write(encrypted_content)

input_file = input("Enter the input filename: ")
output_file = input("Enter the output filename: ")
shift = int(input("Enter the shift value: "))

encrypt_file(input_file, output_file, shift)
print(f"Encrypted file saved as '{output_file}'.")
```

Output:

```
┌──(kruthi㉿kali)-[~]
└─$ python kru3.py
Enter the input filename: kruthi
Enter the output filename: kruthi
Enter the shift value: 2
Encrypted file saved as 'kruthi'.
```

7. Do validation to the python program (2)
   - not to accept special characters
   - not to accept numeric values
   - not to accept empty value
   - accept only string
   - string should be lowercase if not convert the case

```python
def validate_input(user_input):
    if not user_input:
        return "Error: Input cannot be empty."
    if not user_input.isalpha():
        return "Error: Input must contain only alphabetic characters."
    if not user_input.islower():
        user_input = user_input.lower()

    return f"Validated input: {user_input}"
user_input = input("Enter a string: ")
result = validate_input(user_input)
print(result)
```

output:

```
┌──(kruthi㉿kali)-[~]
└─$ python kru3.py
Enter a string: KRUTHI
Validated input: kruthi
```

8. Write a Python program to checks if two given strings are anagrams of each other.
example: mug, gum
cork, rock
note, tone

```python
def are_anagrams(str1, str2):
    return sorted(str1) == sorted(str2)
str1 = input("Enter the first string: ").lower()
str2 = input("Enter the second string: ").lower()
if are_anagrams(str1, str2):
    print(f"'{str1}' and '{str2}' are anagrams.")
else:
    print(f"'{str1}' and '{str2}' are not anagrams.")
```

output:

```
┌──(kruthi㉿kali)-[~]
└─$ python kru3.py
Enter the first string: kruthi
Enter the second string: somalli
'kruthi' and 'somalli' are not anagrams.
```

9. Write a Python program to check the given string is palindrome or not
Do not use built in functions

Example: MADAM
RACECAR
LEVEL
CIVIC

```python
def is_palindrome(string):
    string = string.lower()
    start = 0
    end = len(string) - 1
    while start < end:
        if string[start] != string[end]:
            return False
        start += 1
        end -= 1
    return True
string = input("Enter a string: ")
if is_palindrome(string):
    print(f"'{string}' is a palindrome.")
else:
    print(f"'{string}' is not a palindrome.")
```

output:

```
┌──(kruthi㉿kali)-[~]
└─$ python kru3.py
Enter a string: MADAM
'MADAM' is a palindrome.

┌──(kruthi㉿kali)-[~]
└─$ python kru3.py
Enter a string: RACECAR
'RACECAR' is a palindrome.

┌──(kruthi㉿kali)-[~]
└─$ python kru3.py
Enter a string: LEVEL
'LEVEL' is a palindrome.

┌──(kruthi㉿kali)-[~]
└─$ python kru3.py
Enter a string: CIVIC
'CIVIC' is a palindrome.
```

10. Write a Python program to check if a substring is present in a given string.
Example: Understand -- stand

```python
def is_substring(main_string, sub_string):
    for i in range(len(main_string) - len(sub_string) + 1):
        if main_string[i:i + len(sub_string)] == sub_string:
            return True
    return False
main_string = input("Enter the main string: ").lower()
sub_string = input("Enter the substring: ").lower()
if is_substring(main_string, sub_string):
    print("Substring is present.")
else:
    print("Substring is not present.")
```

output:

```
┌──(kruthi㉿kali)-[~]
└─$ python kru3.py
Enter the main string: understand
Enter the substring: stand
Substring is present.
```

11. Explore string module
    import the string module in your python script.
    print all the lowercase characters
    print all the uppercase characters
    print all the lowercase and uppercase characters
    print all the digits
    print all the punctuation symbols
    count the total number of punctuation symbols

```
import string
print("Lowercase characters:", string.ascii_lowercase)
print("Uppercase characters:", string.ascii_uppercase)
print("All characters:", string.ascii_letters)
print("Digits:", string.digits)
print("Punctuation symbols:", string.punctuation)
print("Total number of punctuation symbols:", len(string.punctuation))
```

output:

```
┌──(kruthi㉿kali)-[~]
└─$ python kru3.py
Lowercase characters: abcdefghijklmnopqrstuvwxyz
Uppercase characters: ABCDEFGHIJKLMNOPQRSTUVWXYZ
All characters: abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
Digits: 0123456789
Punctuation symbols: !"#$%&'()*+,-./:;⟺?@[\]^_`{|}~
Total number of punctuation symbols: 32
```