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

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

text = "HELLO WORLD"
shift = 3
encrypted_text = caesar_cipher(text, shift)
print(f"Encrypted Text: {encrypted_text}")
nano caesar_cipher.py
```

```
┌──(kali㉿kali)-[~/nithesh]
└─$ vi caesar_cipher.py


┌──(kali㉿kali)-[~/nithesh]
└─$ python3 caesar_cipher.py

Encrypted Text: KHOOR ZRUOG
```

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

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

text = input("Enter the text: ")
shift = int(input("Enter the shift value: "))
encrypted_text = caesar_cipher(text, shift)
print(f"Encrypted Text: {encrypted_text}")
```

```
┌──(kali㉿kali)-[~/nithesh]
└─$ vi caesar_cipher_user_choice.py


┌──(kali㉿kali)-[~/nithesh]
└─$ python3 caesar_cipher_user_choice.py

Enter the text: I love python
Enter the shift value: 2
Encrypted Text: K nqxg ravjqp
```

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

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

def format_cipher_text(text):
    text = text.upper().replace(" ", "")
    grouped_text = " ".join([text[i:i+5] for i in range(0, len(text), 5)])
    return grouped_text

text = input("Enter the text: ")
shift = int(input("Enter the shift value: "))
encrypted_text = caesar_cipher(text, shift)
formatted_text = format_cipher_text(encrypted_text)
print(f"Formatted Cipher Text: {formatted_text}")
```

```
┌──(kali㉿kali)-[~/nithesh]
└─$ vi format_cipher.py


┌──(kali㉿kali)-[~/nithesh]
└─$ python3 format_cipher.py

Enter the text: Hi how are u
Enter the shift value: 2
Formatted Cipher Text: JKJQY CTGW
```

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

```
from collections import Counter

def character_histogram(text):
    return Counter(text)

text = input("Enter the text: ")
histogram = character_histogram(text)
for char, freq in histogram.items():
    print(f"{char}: {freq}")
```

```
┌──(kali㉿kali)-[~/nithesh]
└─$ vi histogram.py


┌──(kali㉿kali)-[~/nithesh]
└─$ python3 histogram.py

Enter the text: msis is best
m: 1
s: 4
i: 2
 : 2
b: 1
e: 1
t: 1
```

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

```
def read_file(file_path):
    with open(file_path, 'r') as file:
        return file.read()

file_path = input("Enter the file path: ")
file_contents = read_file(file_path)
print(file_contents)
```

```
┌──(kali㉿kali)-[~/nithesh]
└─$ vi read_file.py


┌──(kali㉿kali)-[~/nithesh]
└─$ python3 read_file.py
```

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

```python
def caesar_encrypt(text, shift):
    result = ""
    for char in text:
        if char.isalpha():
            ascii_offset = 65 if char.isupper() else 97
            encrypted_char = chr((ord(char) - ascii_offset + shift) % 26 + ascii_offset)
            result += encrypted_char
        else:
            result += char
    return result

def read_file(filename):
    try:
        with open(filename, 'r') as file:
            return file.read()
    except FileNotFoundError:
        print(f"Error: File '{filename}' not found.")
        return None

# Get user input
filename = input("Enter the filename to read and encrypt: ")
shift = int(input("Enter the shift value: "))

content = read_file(filename)

if content:
    encrypted_content = caesar_encrypt(content, shift)
    print("Encrypted content:")
    print(encrypted_content)
```

```
┌──(kali㉿kali)-[~/nithesh]
└─$ python3 encrypt_file.py

Enter the filename to read and encrypt: file3
Enter the shift value: 2

┌──(kali㉿kali)-[~/nithesh]
└─$ ls -l
total 128
drwxrwxr-x 2 kali kali 4096 Aug 16 10:43 '*[[:alnum:]]*'
drwxrwxr-x 2 kali kali 4096 Aug 16 10:42  alnum
drwxrwxr-x 2 kali kali 4096 Aug 16 10:49 '*[[:alpha:]]*'
drwxrwxr-x 2 kali kali 4096 Aug 16 10:49 '*[[:alpha:]]*,ls'
-rw-rw-r-- 1 kali kali  420 Aug 28 04:28  caesar_cipher.py
-rw-rw-r-- 1 kali kali  468 Aug 28 04:31  caesar_cipher_user_choice.py
drwxr-xr-x 4 kali kali 4096 Aug  6 13:22  CLI
-rw-rw-r-- 1 kali kali  244 Aug 23 10:26  command.sh
-rwxrwxr-x 1 kali kali  220 Aug 23 10:40  count_files.sh
drwxr-xr-x 8 kali kali 4096 Aug 12 10:24  cys
drwxrwsr-x 2 kali kali 4096 Aug 28 04:18  day5
drwxrwxr-x 2 kali kali 4096 Aug 16 10:49 '*[[:digit:]]*,ls'
-rw-rw-r-- 1 kali kali  836 Aug 28 05:26  encrypt_file.py
-rw-r--r-- 1 kali kali    0 Aug  6 12:30  example.txt
-rw-rw-r-- 1 kali kali    0 Aug 12 10:18  file1
-rw-r--r-- 1 kali kali    0 Aug 12 10:18  file2
```

7. Do validation to the python program (2)

```python
def caesar_encrypt(text, shift):
    result = ""
    for char in text:
        if char.isalpha():
            ascii_offset = 65 if char.isupper() else 97
            encrypted_char = chr((ord(char) - ascii_offset + shift) % 26 + ascii_offset)
            result += encrypted_char
        else:
            result += char
    return result

# Example usage
plaintext = "Hello, World!"
shift = 3
ciphertext = caesar_encrypt(plaintext, shift)
print(f"Plaintext: {plaintext}")
print(f"Ciphertext: {ciphertext}")
~
```

```
┌──(kali㉿kali)-[~/nithesh]
└─$ vi encrypt_file.py


┌──(kali㉿kali)-[~/nithesh]
└─$ python3 encrypt_file.py

Plaintext: Hello, World!
Ciphertext: Khoor, Zruog!
```

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):
    # Remove spaces and convert to lowercase
    str1 = str1.replace(" ", "").lower()
    str2 = str2.replace(" ", "").lower()

    # Check if the lengths are the same
    if len(str1) != len(str2):
        return False

    # Create dictionaries to store character counts
    char_count1 = {}
    char_count2 = {}

    # Count characters in both strings
    for char in str1:
        char_count1[char] = char_count1.get(char, 0) + 1
    for char in str2:
        char_count2[char] = char_count2.get(char, 0) + 1

    # Compare the character counts
    return char_count1 == char_count2

# Example usage
string1 = input("Enter the first string: ")
string2 = input("Enter the second string: ")

if are_anagrams(string1, string2):
    print(f"'{string1}' and '{string2}' are anagrams.")
else:
    print(f"'{string1}' and '{string2}' are not anagrams.")
~
```

```
┌──(kali㉿kali)-[~/nithesh]
└─$ vi anagram_check.py


┌──(kali㉿kali)-[~/nithesh]
└─$ python3 anagram_check.py

Enter the first string: anagram
Enter the second string: kali
'anagram' and 'kali' are not anagrams.

┌──(kali㉿kali)-[~/nithesh]
└─$ python3 anagram_check.py

Enter the first string: mug
Enter the second string: gum
'mug ' and 'gum' are anagrams.
```

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

```python
def is_palindrome(s):
    # Remove non-alphanumeric characters and convert to lowercase
    s = ''.join(char.lower() for char in s if char.isalnum())

    # Compare characters from start and end
    left, right = 0, len(s) - 1
    while left < right:
        if s[left] != s[right]:
            return False
        left += 1
        right -= 1

    return True

# Example usage
input_string = input("Enter a string to check if it's a palindrome: ")

if is_palindrome(input_string):
    print(f"'{input_string}' is a palindrome.")
else:
    print(f"'{input_string}' is not a palindrome.")
```

```
┌──(kali㉿kali)-[~/nithesh]
└─$ vi palindrome

┌──(kali㉿kali)-[~/nithesh]
└─$ python3 palindrome
Enter a string to check if it's a palindrome: madam
'madam' is a palindrome.

┌──(kali㉿kali)-[~/nithesh]
└─$ python3 palindrome
Enter a string to check if it's a palindrome: racecar
'racecar' is a palindrome.
```

Example: MADAM
RACECAR
LEVEL
CIVIC

10. Write a Python program to check if a substring is present in a given string.

```
def is_substring(main_string, sub_string):
    main_len = len(main_string)
    sub_len = len(sub_string)

    for i in range(main_len - sub_len + 1):
        if main_string[i:i+sub_len] == sub_string:
            return True

    return False

# Example usage
main_string = input("Enter the main string: ")
sub_string = input("Enter the substring to search for: ")

if is_substring(main_string, sub_string):
    print(f"'{sub_string}' is a substring of '{main_string}'.")
else:
    print(f"'{sub_string}' is not a substring of '{main_string}'.")
```

```
┌──(kali㉿kali)-[~/nithesh]
└─$ python3 substracting_checker.py
Enter the main string: Understand -- stand

Enter the substring to search for: ' ' is a substring of 'Understand -- stand'.
```

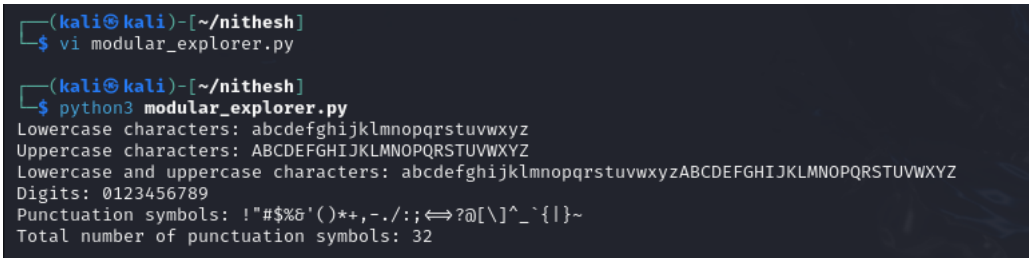
Example: Understand -- stand

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("Lowercase and uppercase characters:", string.ascii_letters)
print("Digits:", string.digits)
print("Punctuation symbols:", string.punctuation)

print("Total number of punctuation symbols:", len(string.punctuation))
~
```



Programming is a skill best acquired by practice and example rather than from books -- unknown

The only way to do great work is to love what you do --Steve Jobs