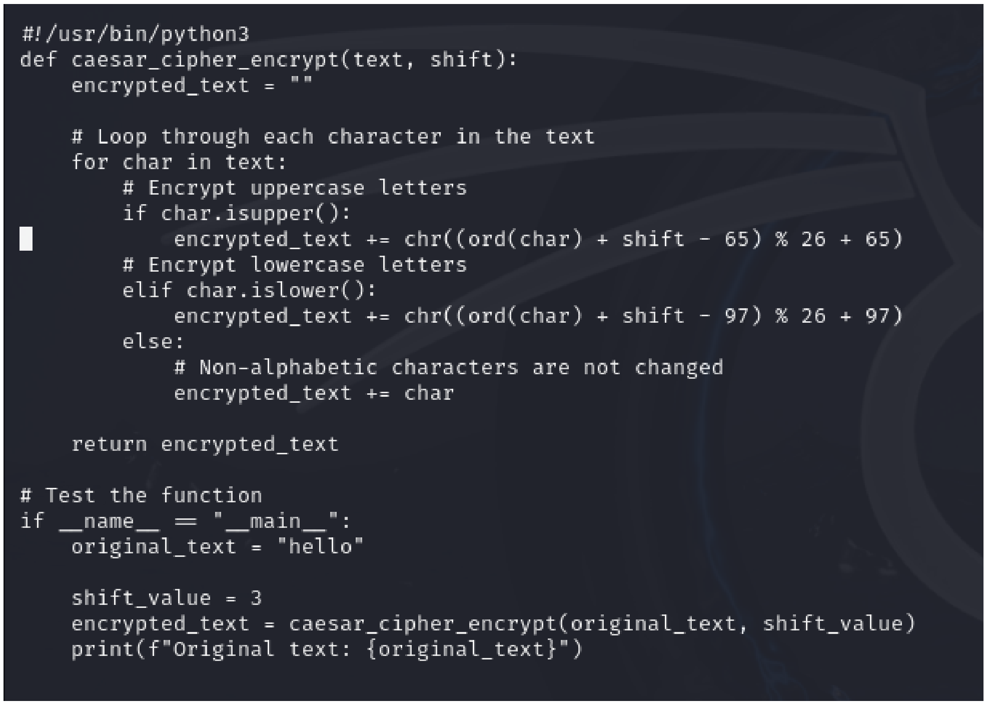
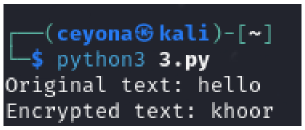
1. Write a python script to encrypt the string using Caesar cipher.

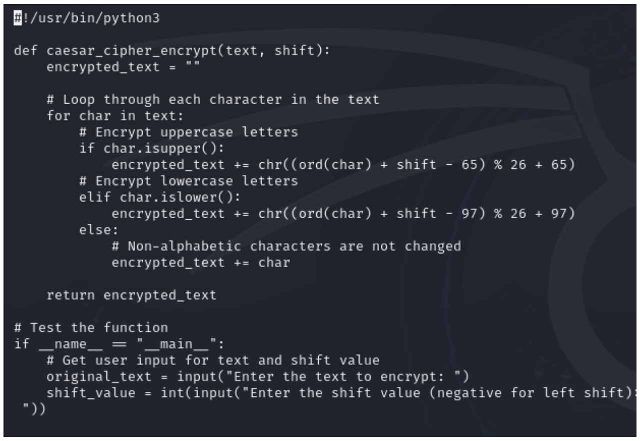


```
#!/usr/bin/python3
def caesar_cipher_encrypt(text, shift):
    encrypted_text = ""

    # Loop through each character in the text
    for char in text:
        # Encrypt uppercase letters
        if char.isupper():
            encrypted_text += chr((ord(char) + shift - 65) % 26 + 65)
        # Encrypt lowercase letters
        elif char.islower():
            encrypted_text += chr((ord(char) + shift - 97) % 26 + 97)
        else:
            # Non-alphabetic characters are not changed
            encrypted_text += char

    return encrypted_text

# Test the function
if __name__ == "__main__":
    original_text = "hello"

    shift_value = 3
    encrypted_text = caesar_cipher_encrypt(original_text, shift_value)
    print(f"Original text: {original_text}")
```

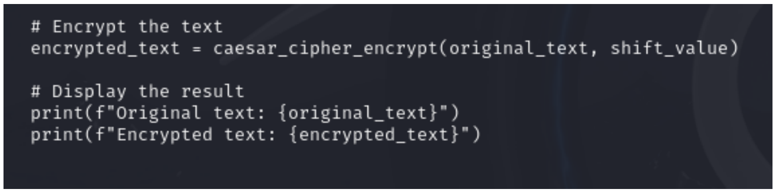


```
┌──(ceyona㉿kali)-[~]
└─$ python3 3.py
Original text: hello
Encrypted text: khoor
```

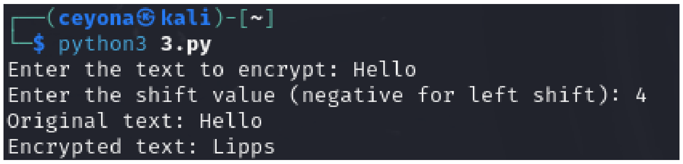
2. Write a Python script to Modify the above script to shift cipher based on user choice.



```
#!/usr/bin/python3

def caesar_cipher_encrypt(text, shift):
    encrypted_text = ""

    # Loop through each character in the text
    for char in text:
        # Encrypt uppercase letters
        if char.isupper():
            encrypted_text += chr((ord(char) + shift - 65) % 26 + 65)
        # Encrypt lowercase letters
        elif char.islower():
            encrypted_text += chr((ord(char) + shift - 97) % 26 + 97)
        else:
            # Non-alphabetic characters are not changed
            encrypted_text += char

    return encrypted_text

# Test the function
if __name__ == "__main__":
    # Get user input for text and shift value
    original_text = input("Enter the text to encrypt: ")
    shift_value = int(input("Enter the shift value (negative for left shift):
 "))
```

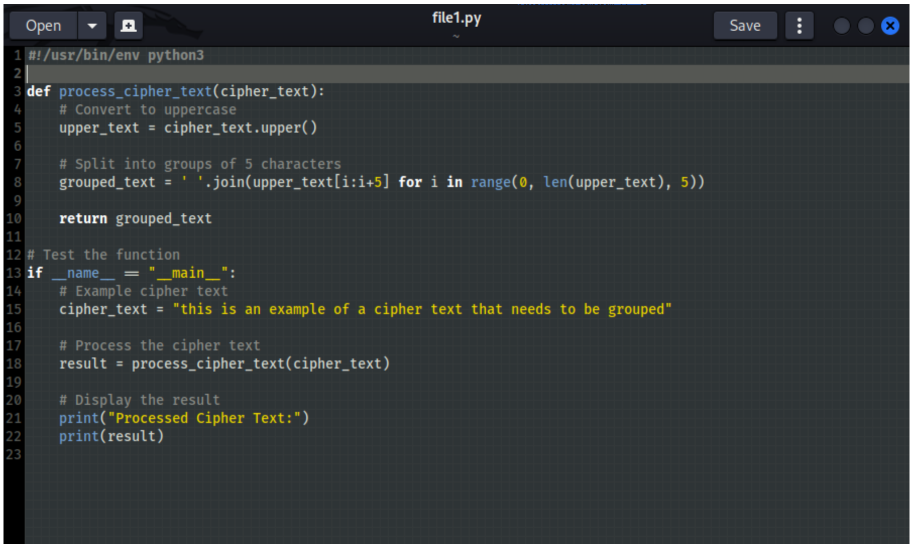


```
# Encrypt the text
encrypted_text = caesar_cipher_encrypt(original_text, shift_value)

# Display the result
print(f"Original text: {original_text}")
print(f"Encrypted text: {encrypted_text}")
```

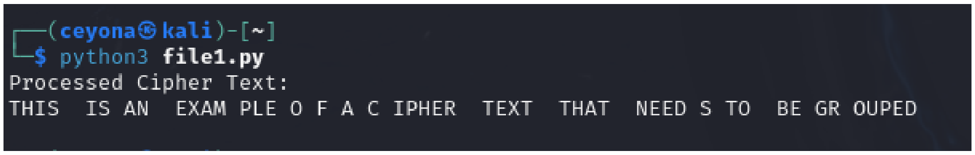


```
┌──(ceyona㉿kali)-[~]
└─$ python3 3.py
Enter the text to encrypt: Hello
Enter the shift value (negative for left shift): 4
Original text: Hello
Encrypted text: Lipps
```

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

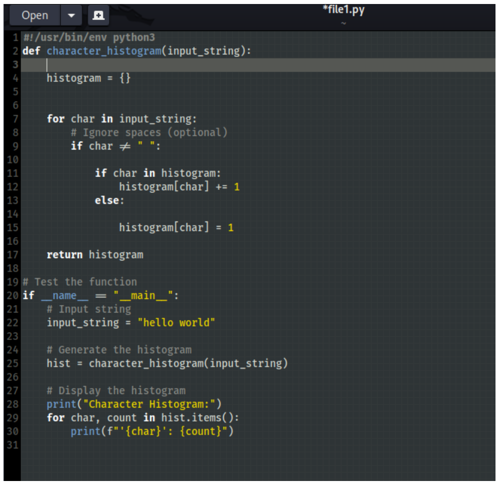


```python
#!/usr/bin/env python3

def process_cipher_text(cipher_text):
    # Convert to uppercase
    upper_text = cipher_text.upper()

    # Split into groups of 5 characters
    grouped_text = ' '.join(upper_text[i:i+5] for i in range(0, len(upper_text), 5))

    return grouped_text

# Test the function
if __name__ == "__main__":
    # Example cipher text
    cipher_text = "this is an example of a cipher text that needs to be grouped"

    # Process the cipher text
    result = process_cipher_text(cipher_text)

    # Display the result
    print("Processed Cipher Text:")
    print(result)
```

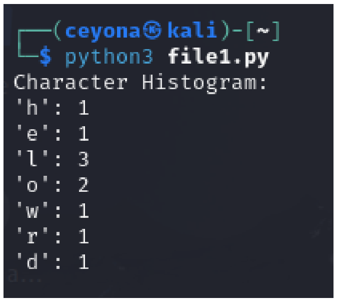


```
┌──(ceyona㉿kali)-[~]
└─$ python3 file1.py
Processed Cipher Text:
THIS  IS AN  EXAM PLE O F A C IPHER  TEXT  THAT  NEED S TO  BE GR OUPED
```

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

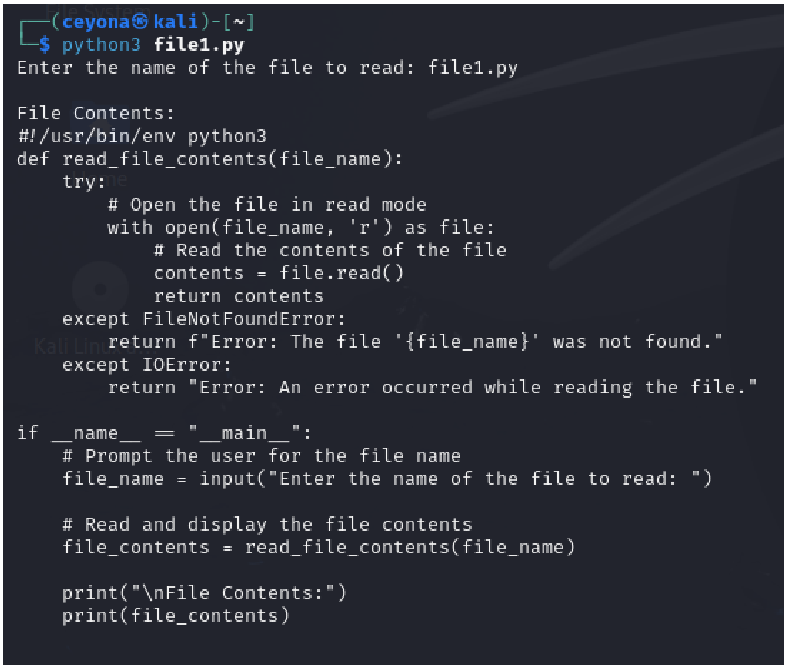


```python
#!/usr/bin/env python3
def character_histogram(input_string):

    histogram = {}


    for char in input_string:
        # Ignore spaces (optional)
        if char != " ":

            if char in histogram:
                histogram[char] += 1
            else:

                histogram[char] = 1

    return histogram

# Test the function
if __name__ == "__main__":
    # Input string
    input_string = "hello world"

    # Generate the histogram
    hist = character_histogram(input_string)

    # Display the histogram
    print("Character Histogram:")
    for char, count in hist.items():
        print(f"'{char}': {count}")
```



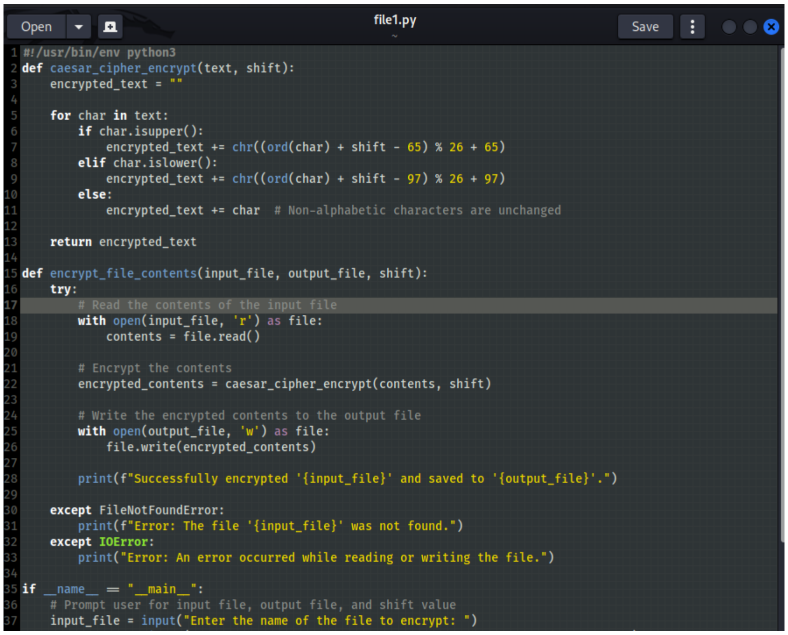
```
┌──(ceyona㉿kali)-[~]
└─$ python3 file1.py
Character Histogram:
'h': 1
'e': 1
'l': 3
'o': 2
'w': 1
'r': 1
'd': 1
```

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

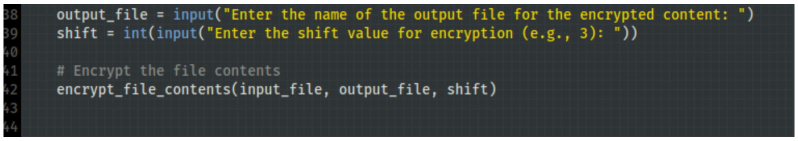


```
┌──(ceyona㉿kali)-[~]
└─$ python3 file1.py
Enter the name of the file to read: file1.py

File Contents:
#!/usr/bin/env python3
def read_file_contents(file_name):
    try:
        # Open the file in read mode
        with open(file_name, 'r') as file:
            # Read the contents of the file
            contents = file.read()
            return contents
    except FileNotFoundError:
        return f"Error: The file '{file_name}' was not found."
    except IOError:
        return "Error: An error occurred while reading the file."

if __name__ == "__main__":
    # Prompt the user for the file name
    file_name = input("Enter the name of the file to read: ")

    # Read and display the file contents
    file_contents = read_file_contents(file_name)

    print("\nFile Contents:")
    print(file_contents)
```

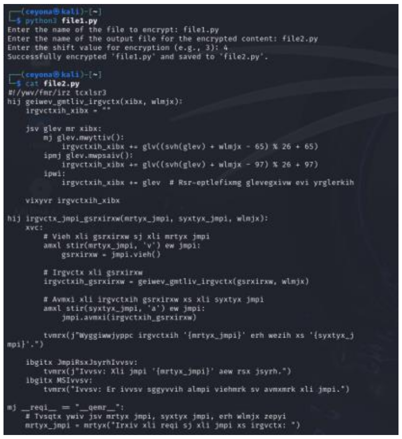
6. Write a Python script to encrypt the contents from the file.



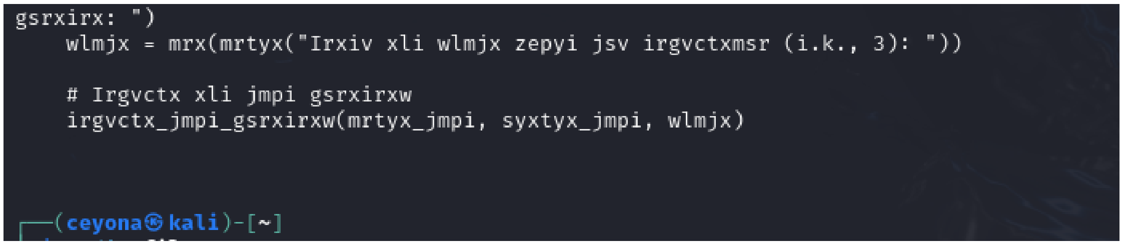
```python
#!/usr/bin/env python3
def caesar_cipher_encrypt(text, shift):
    encrypted_text = ""

    for char in text:
        if char.isupper():
            encrypted_text += chr((ord(char) + shift - 65) % 26 + 65)
        elif char.islower():
            encrypted_text += chr((ord(char) + shift - 97) % 26 + 97)
        else:
            encrypted_text += char  # Non-alphabetic characters are unchanged

    return encrypted_text

def encrypt_file_contents(input_file, output_file, shift):
    try:
        # Read the contents of the input file
        with open(input_file, 'r') as file:
            contents = file.read()

        # Encrypt the contents
        encrypted_contents = caesar_cipher_encrypt(contents, shift)

        # Write the encrypted contents to the output file
        with open(output_file, 'w') as file:
            file.write(encrypted_contents)

        print(f"Successfully encrypted '{input_file}' and saved to '{output_file}'.")

    except FileNotFoundError:
        print(f"Error: The file '{input_file}' was not found.")
    except IOError:
        print("Error: An error occurred while reading or writing the file.")

if __name__ == "__main__":
    # Prompt user for input file, output file, and shift value
    input_file = input("Enter the name of the file to encrypt: ")
```



```python
    output_file = input("Enter the name of the output file for the encrypted content: ")
    shift = int(input("Enter the shift value for encryption (e.g., 3): "))

    # Encrypt the file contents
    encrypt_file_contents(input_file, output_file, shift)
```



```
┌──(ceyona㉿kali)-[~]
└─$ python3 file1.py
Enter the name of the file to encrypt: file1.py
Enter the name of the output file for the encrypted content: file2.py
Enter the shift value for encryption (e.g., 3): 4
Successfully encrypted 'file1.py' and saved to 'file2.py'.

┌──(ceyona㉿kali)-[~]
└─$ cat file2.py
#!/ywv/fmr/irz tcxlsr3
hij geiwev_gmtliv_irgvctx(xibx, wlmjx):
    irgvctxih_xibx = ""

    jsv glev mr xibx:
        mj glev.mwyttiv():
            irgvctxih_xibx += glv((svh(glev) + wlmjx - 65) % 26 + 65)
        ipmj glev.mwpsaiv():
            irgvctxih_xibx += glv((svh(glev) + wlmjx - 97) % 26 + 97)
        ipwi:
            irgvctxih_xibx += glev  # Rsr-eptlefixmg glevegxivw evi yrglerkih

    vixyvr irgvctxih_xibx

hij irgvctx_jmpi_gsrxirxw(mrtyx_jmpi, syxtyx_jmpi, wlmjx):
    xvc:
        # Vieh xli gsrxirxw sj xli mrtyx jmpi
        amxl stir(mrtyx_jmpi, 'v') ew jmpi:
            gsrxirxw = jmpi.vieh()

        # Irgvctx xli gsrxirxw
        irgvctxih_gsrxirxw = geiwev_gmtliv_irgvctx(gsrxirxw, wlmjx)

        # Avmxi xli irgvctxih gsrxirxw xs xli syxtyx jmpi
        amxl stir(syxtyx_jmpi, 'a') ew jmpi:
            jmpi.avmxi(irgvctxih_gsrxirxw)

        tvmrx(j"Wyggiwwjyppc irgvctxih '{mrtyx_jmpi}' erh wezih xs '{syxtyx_j
mpi}'.")

    ibgitx JmpiRsxJsyrhIvvsv:
        tvmrx(j"Ivvsv: Xli jmpi '{mrtyx_jmpi}' aew rsx jsyrh.")
    ibgitx MSIvvsv:
        tvmrx("Ivvsv: Er ivvsv sggyvvih almpi viehmrk sv avmxmrk xli jmpi.")

mj __reqi__ == "__qemr__":
    # Tvsqtx ywiv jsv mrtyx jmpi, syxtyx jmpi, erh wlmjx zepyi
    mrtyx_jmpi = mrtyx("Irxiv xli reqi sj xli jmpi xs irgvctx: ")
```



```
gsrxirx: ")
    wlmjx = mrx(mrtyx("Irxiv xli wlmjx zepyi jsv irgvctxmsr (i.k., 3): "))

    # Irgvctx xli jmpi gsrxirxw
    irgvctx_jmpi_gsrxirxw(mrtyx_jmpi, syxtyx_jmpi, wlmjx)


┌──(ceyona㉿kali)-[~]
```

7. Do validation to the python program (2)

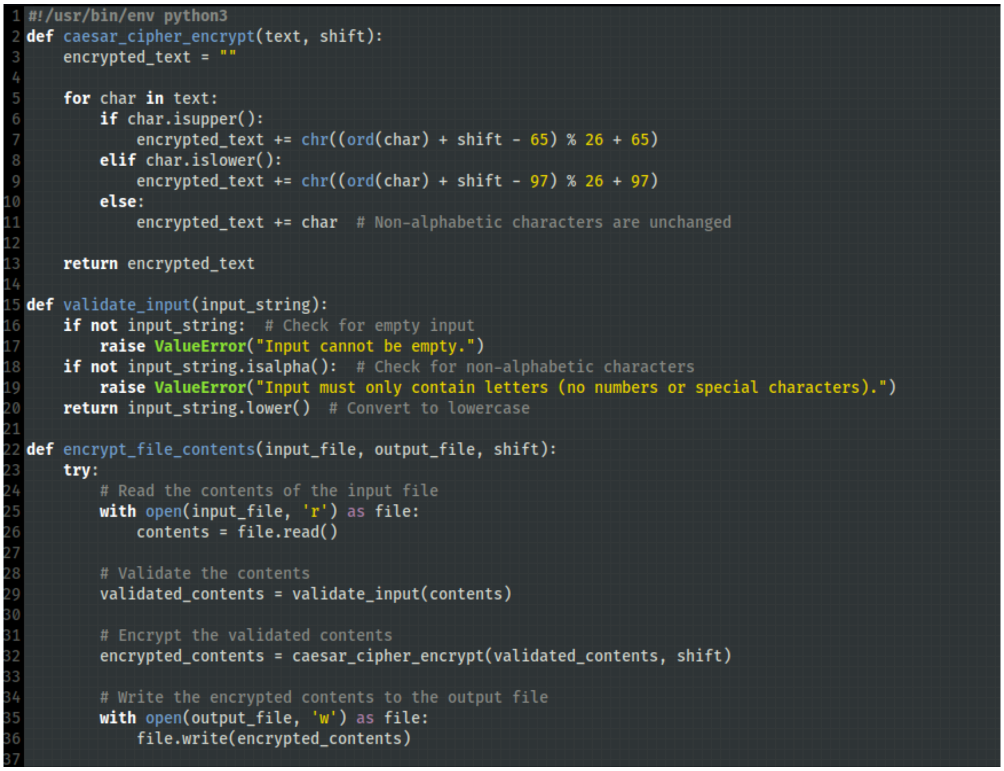
- not to accept special characters

- not to accept numeric values

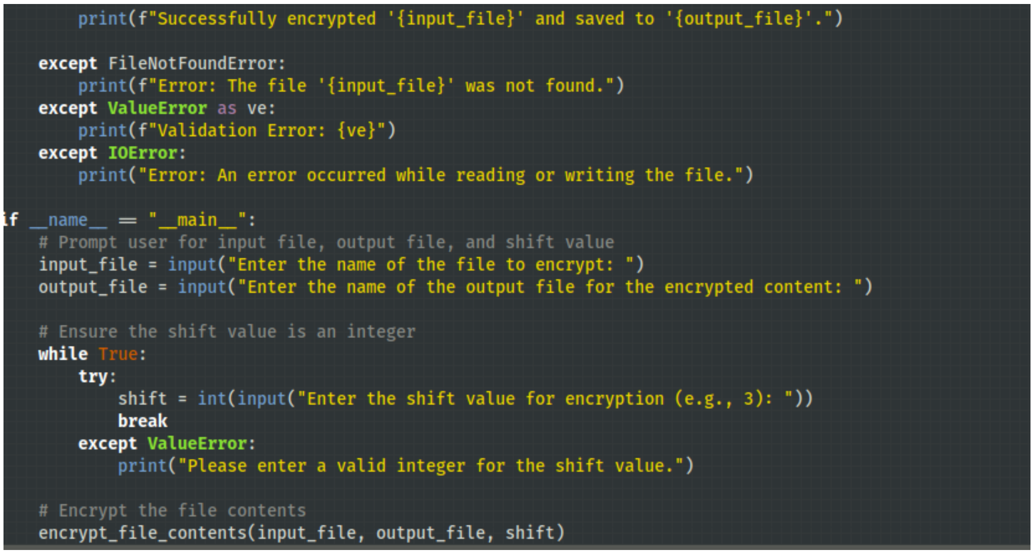
- not to accept empty value

- accept only string

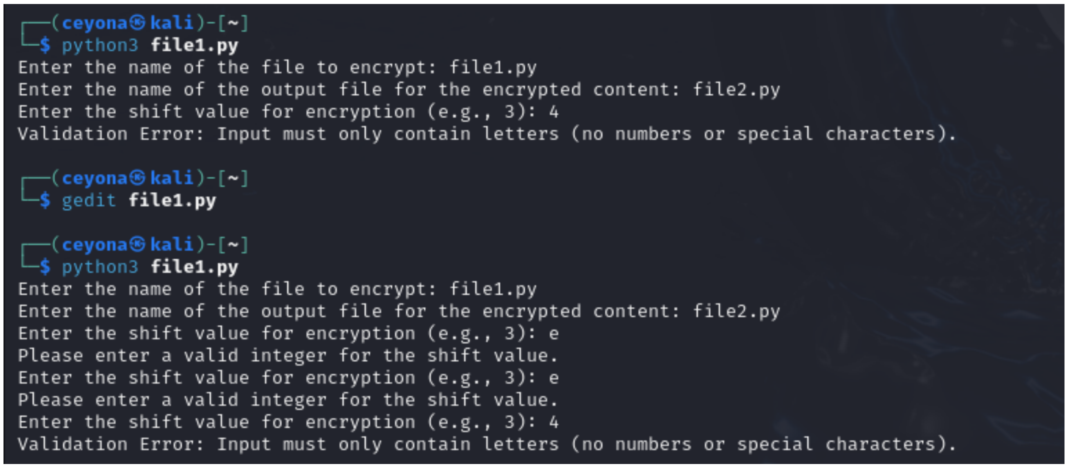
- string should be lowercase if not convert the case



```python
#!/usr/bin/env python3
def caesar_cipher_encrypt(text, shift):
    encrypted_text = ""

    for char in text:
        if char.isupper():
            encrypted_text += chr((ord(char) + shift - 65) % 26 + 65)
        elif char.islower():
            encrypted_text += chr((ord(char) + shift - 97) % 26 + 97)
        else:
            encrypted_text += char  # Non-alphabetic characters are unchanged

    return encrypted_text

def validate_input(input_string):
    if not input_string:  # Check for empty input
        raise ValueError("Input cannot be empty.")
    if not input_string.isalpha():  # Check for non-alphabetic characters
        raise ValueError("Input must only contain letters (no numbers or special characters).")
    return input_string.lower()  # Convert to lowercase

def encrypt_file_contents(input_file, output_file, shift):
    try:
        # Read the contents of the input file
        with open(input_file, 'r') as file:
            contents = file.read()

        # Validate the contents
        validated_contents = validate_input(contents)

        # Encrypt the validated contents
        encrypted_contents = caesar_cipher_encrypt(validated_contents, shift)

        # Write the encrypted contents to the output file
        with open(output_file, 'w') as file:
            file.write(encrypted_contents)

```



```python
        print(f"Successfully encrypted '{input_file}' and saved to '{output_file}'.")

    except FileNotFoundError:
        print(f"Error: The file '{input_file}' was not found.")
    except ValueError as ve:
        print(f"Validation Error: {ve}")
    except IOError:
        print("Error: An error occurred while reading or writing the file.")

if __name__ == "__main__":
    # Prompt user for input file, output file, and shift value
    input_file = input("Enter the name of the file to encrypt: ")
    output_file = input("Enter the name of the output file for the encrypted content: ")

    # Ensure the shift value is an integer
    while True:
        try:
            shift = int(input("Enter the shift value for encryption (e.g., 3): "))
            break
        except ValueError:
            print("Please enter a valid integer for the shift value.")

    # Encrypt the file contents
    encrypt_file_contents(input_file, output_file, shift)
```



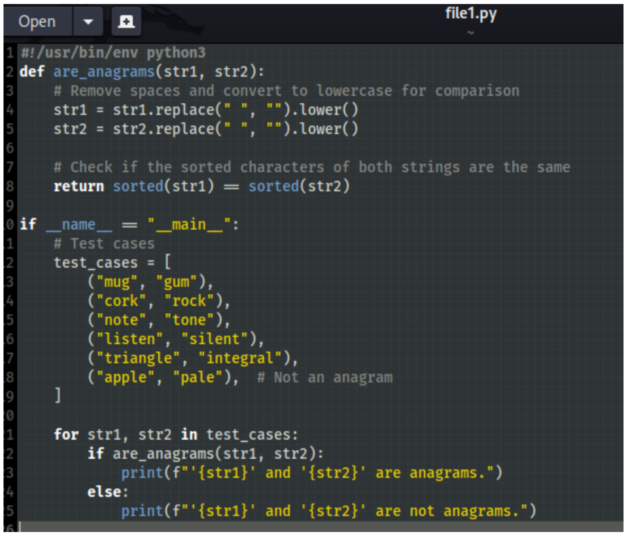
```
┌──(ceyona㉿kali)-[~]
└─$ python3 file1.py
Enter the name of the file to encrypt: file1.py
Enter the name of the output file for the encrypted content: file2.py
Enter the shift value for encryption (e.g., 3): 4
Validation Error: Input must only contain letters (no numbers or special characters).

┌──(ceyona㉿kali)-[~]
└─$ gedit file1.py

┌──(ceyona㉿kali)-[~]
└─$ python3 file1.py
Enter the name of the file to encrypt: file1.py
Enter the name of the output file for the encrypted content: file2.py
Enter the shift value for encryption (e.g., 3): e
Please enter a valid integer for the shift value.
Enter the shift value for encryption (e.g., 3): e
Please enter a valid integer for the shift value.
Enter the shift value for encryption (e.g., 3): 4
Validation Error: Input must only contain letters (no numbers or special characters).
```

8. Write a Python program to checks if two given strings are anagrams of each other.

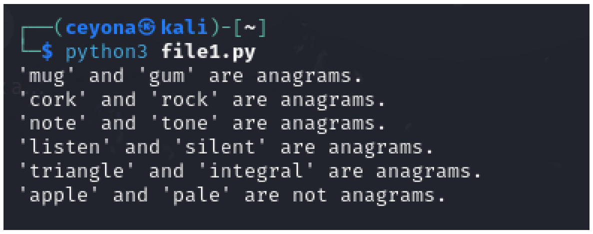
example: mug, gum

cork, rock

note, tone



```python
#!/usr/bin/env python3
def are_anagrams(str1, str2):
    # Remove spaces and convert to lowercase for comparison
    str1 = str1.replace(" ", "").lower()
    str2 = str2.replace(" ", "").lower()

    # Check if the sorted characters of both strings are the same
    return sorted(str1) == sorted(str2)

if __name__ == "__main__":
    # Test cases
    test_cases = [
        ("mug", "gum"),
        ("cork", "rock"),
        ("note", "tone"),
        ("listen", "silent"),
        ("triangle", "integral"),
        ("apple", "pale"),  # Not an anagram
    ]

    for str1, str2 in test_cases:
        if are_anagrams(str1, str2):
            print(f"'{str1}' and '{str2}' are anagrams.")
        else:
            print(f"'{str1}' and '{str2}' are not anagrams.")
```



```
┌──(ceyona㉿kali)-[~]
└─$ python3 file1.py
'mug' and 'gum' are anagrams.
'cork' and 'rock' are anagrams.
'note' and 'tone' are anagrams.
'listen' and 'silent' are anagrams.
'triangle' and 'integral' are anagrams.
'apple' and 'pale' are not anagrams.
```

9. Write a Python program to check the given string is palindrome or not

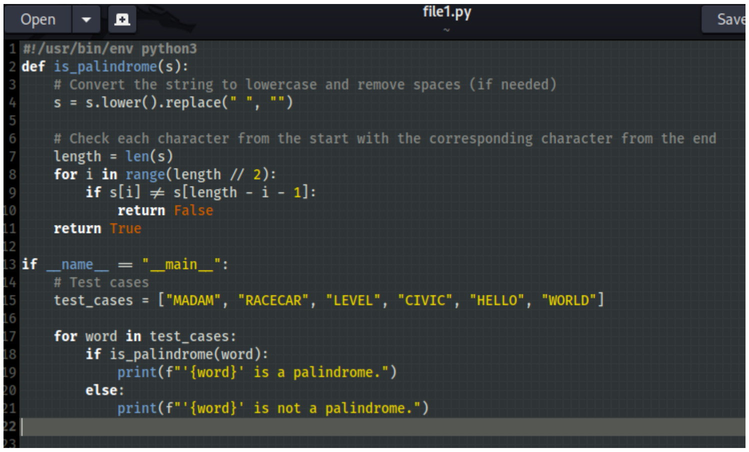
Do not use built in functions

Example: MADAM

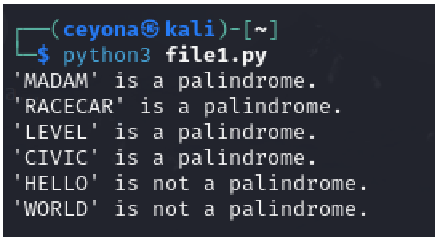
RACECAR

LEVEL

CIVIC



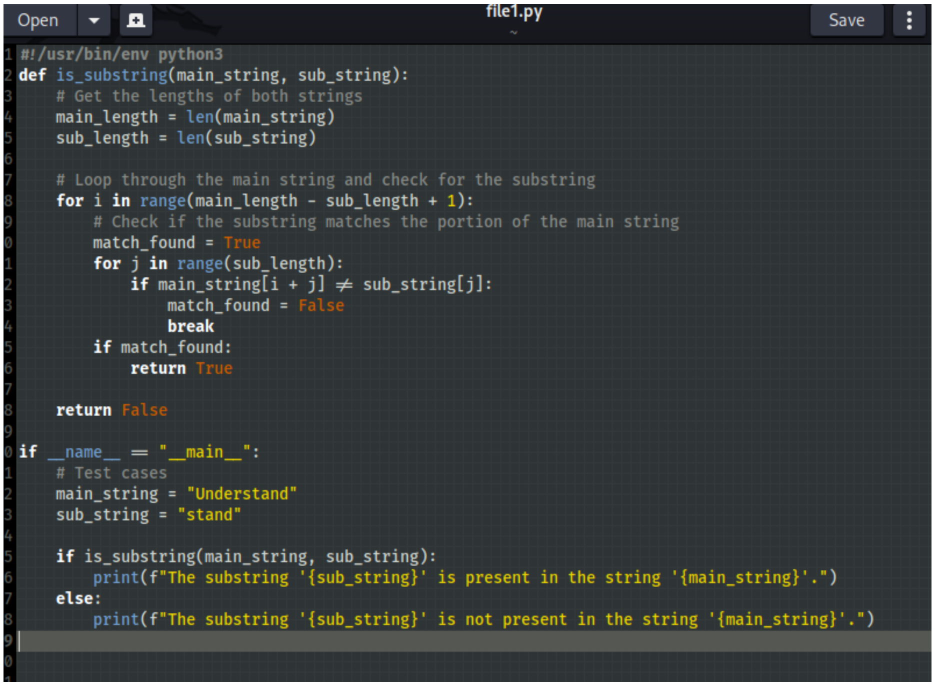
```
#!/usr/bin/env python3
def is_palindrome(s):
    # Convert the string to lowercase and remove spaces (if needed)
    s = s.lower().replace(" ", "")

    # Check each character from the start with the corresponding character from the end
    length = len(s)
    for i in range(length // 2):
        if s[i] != s[length - i - 1]:
            return False
    return True

if __name__ == "__main__":
    # Test cases
    test_cases = ["MADAM", "RACECAR", "LEVEL", "CIVIC", "HELLO", "WORLD"]

    for word in test_cases:
        if is_palindrome(word):
            print(f"'{word}' is a palindrome.")
        else:
            print(f"'{word}' is not a palindrome.")
```



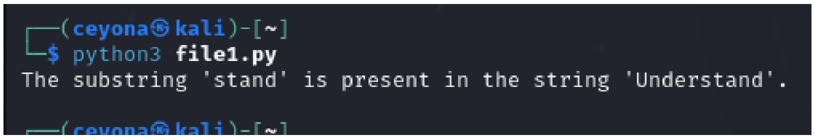
```
┌──(ceyona㉿kali)-[~]
└─$ python3 file1.py
'MADAM' is a palindrome.
'RACECAR' is a palindrome.
'LEVEL' is a palindrome.
'CIVIC' is a palindrome.
'HELLO' is not a palindrome.
'WORLD' is not a palindrome.
```

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

Example: Understand – stand



```python
#!/usr/bin/env python3
def is_substring(main_string, sub_string):
    # Get the lengths of both strings
    main_length = len(main_string)
    sub_length = len(sub_string)

    # Loop through the main string and check for the substring
    for i in range(main_length - sub_length + 1):
        # Check if the substring matches the portion of the main string
        match_found = True
        for j in range(sub_length):
            if main_string[i + j] != sub_string[j]:
                match_found = False
                break
        if match_found:
            return True

    return False

if __name__ == "__main__":
    # Test cases
    main_string = "Understand"
    sub_string = "stand"

    if is_substring(main_string, sub_string):
        print(f"The substring '{sub_string}' is present in the string '{main_string}'.")
    else:
        print(f"The substring '{sub_string}' is not present in the string '{main_string}'.")
```



```
┌──(ceyona㉿kali)-[~]
└─$ python3 file1.py
The substring 'stand' is present in the string 'Understand'.

┌──(ceyona㉿kali)-[~]
```

11. Explore string module

import the string module in your python script.

print all the lowercase characters

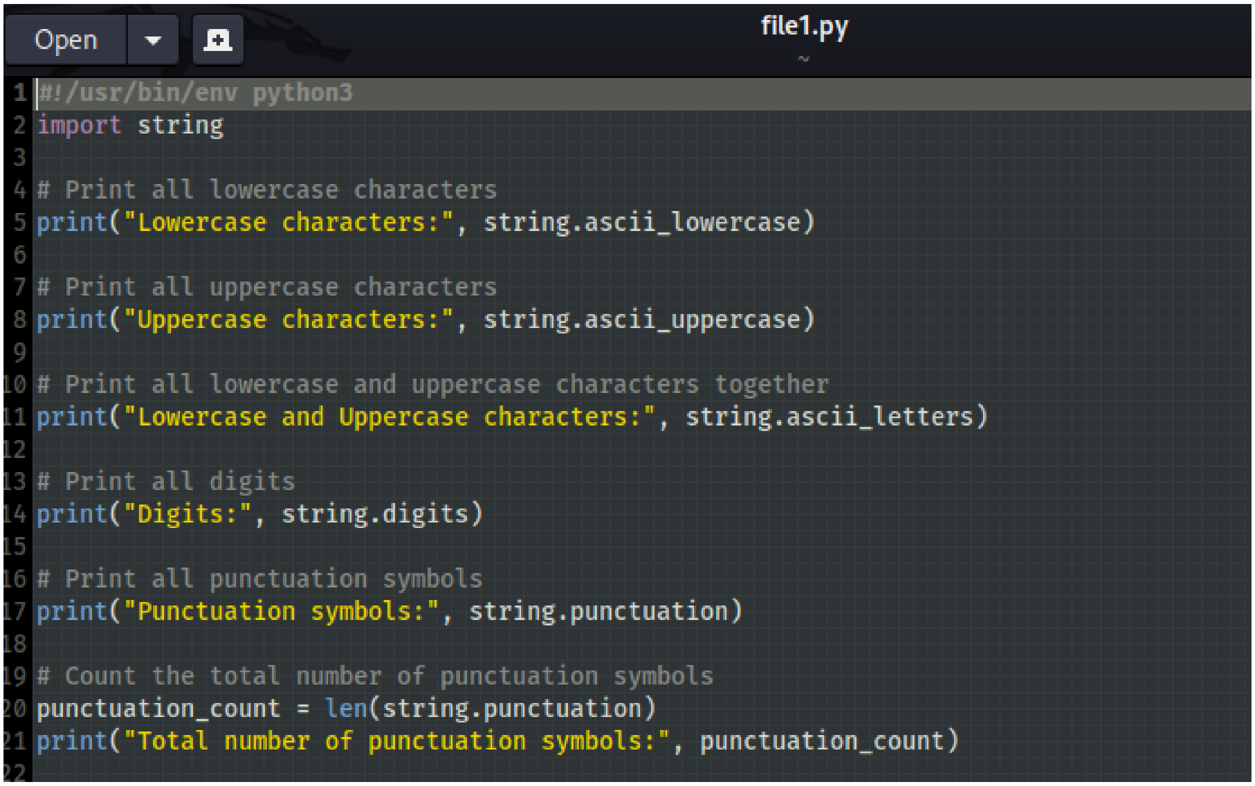
print all the uppercase characters

print all the lowercase and uppercase characters

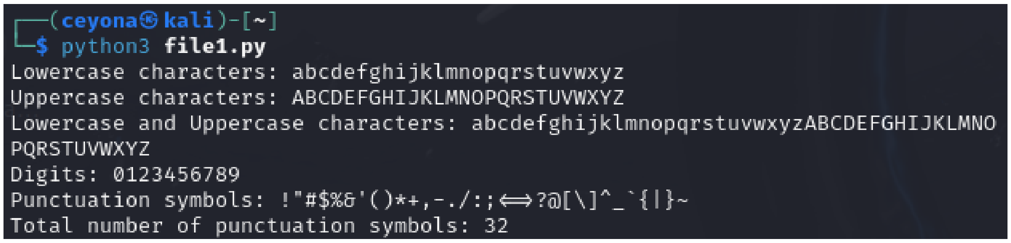
print all the digits

print all the punctuation symbols

count the total number of punctuation symbols



```python
#!/usr/bin/env python3
import string

# Print all lowercase characters
print("Lowercase characters:", string.ascii_lowercase)

# Print all uppercase characters
print("Uppercase characters:", string.ascii_uppercase)

# Print all lowercase and uppercase characters together
print("Lowercase and Uppercase characters:", string.ascii_letters)

# Print all digits
print("Digits:", string.digits)

# Print all punctuation symbols
print("Punctuation symbols:", string.punctuation)

# Count the total number of punctuation symbols
punctuation_count = len(string.punctuation)
print("Total number of punctuation symbols:", punctuation_count)
```



```
┌──(ceyona㉿kali)-[~]
└─$ python3 file1.py
Lowercase characters: abcdefghijklmnopqrstuvwxyz
Uppercase characters: ABCDEFGHIJKLMNOPQRSTUVWXYZ
Lowercase and Uppercase characters: abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNO
PQRSTUVWXYZ
Digits: 0123456789
Punctuation symbols: !"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
Total number of punctuation symbols: 32
```

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