Reverse a Given Number

a) Take the value of the integer and store in a variable.

b) Using a while loop, get each digit of the number and store the reversed number in another variable.

c) Print the reverse of the number.

2. Print largest permutation number of a given number

3. Find the number of ones in the binary representation of a number

Reverse a Given Number:

def reverse\_number(num):

reversed\_num = 0

while num > 0:

digit = num % 10

reversed\_num = (reversed\_num \* 10) + digit

num = num // 10

return reversed\_num

num = int(input("Enter a number: "))

reversed\_num = reverse\_number(num)

print("Reverse of the number:", reversed\_num)

Print the largest permutation number of a given number:

def largest\_permutation(num):

num\_str = str(num)

num\_digits = [int(d) for d in num\_str]

num\_digits.sort(reverse=True)

largest\_num = int(''.join(map(str, num\_digits)))

return largest\_num

num = int(input("Enter a number: "))

largest\_num = largest\_permutation(num)

print("Largest permutation number:", largest\_num)

Find the number of ones in the binary representation of a number:

def count\_ones\_binary(num):

count = 0

while num:

count += num & 1

num >>= 1

return count

num = int(input("Enter a number: "))

ones\_count = count\_ones\_binary(num)

print("Number of ones in binary representation:", ones\_count)

Set-2:

1. Find the cumulative sum of a list where the i-th element is the sum of the first i+1 elements from the original list.

2. Given a list of sorted numbers and a variable K, where K is also a number, write a Python program using binary search to find the number in the list which is closest to the given number K

3. Given a list of tuples, write a Python program to remove all the duplicated tuples from the given list using the concept of set.

4. Given an unsorted list of some elements (may or may not be integers), Find the frequency of each distinct element in the list using a dictionary.

5. Given two words, check whether they are anagrams using dictionary.

6. Find common elements in three sorted lists using sets.

7. Write a python program to find the most occurrence character and its number of occurrences.

8. Find Symmetric Pairs in dictionary using loop.

9. Determine common prefix from a list of strings.

Find the cumulative sum of a list:

def cumulative\_sum(lst):

cumulative = []

total = 0

i = 0

while i < len(lst):

total += lst[i]

cumulative.append(total)

i += 1

return cumulative

# Taking user input for the list

lst = []

n = int(input("Enter the number of elements in the list: "))

i = 0

while i < n:

element = int(input("Enter element " + str(i + 1) + ": "))

lst.append(element)

i += 1

result = cumulative\_sum(lst)

print("Cumulative sum:", result)

Find the number in a sorted list closest to a given number using binary search:

def closest\_number(sorted\_list, K):

low = 0

high = len(sorted\_list) - 1

closest = None

while low <= high:

mid = (low + high) // 2

if sorted\_list[mid] == K:

return sorted\_list[mid]

elif sorted\_list[mid] < K:

closest = sorted\_list[mid]

low = mid + 1

else:

high = mid - 1

if closest is None:

return sorted\_list[low]

elif abs(closest - K) < abs(sorted\_list[low] - K):

return closest

else:

return sorted\_list[low]

# Taking user input for the sorted list

sorted\_numbers = []

n = int(input("Enter the number of elements in the sorted list: "))

i = 0

while i < n:

element = int(input("Enter element " + str(i + 1) + ": "))

sorted\_numbers.append(element)

i += 1

K = int(input("Enter the value of K: "))

closest = closest\_number(sorted\_numbers, K)

print("Closest number:", closest)

Remove duplicated tuples from a list using sets:

def remove\_duplicates(tuples\_list):

result = []

for tuple\_item in tuples\_list:

if tuple\_item not in result:

result.append(tuple\_item)

return result

tuples\_list = [(1, 2), (3, 4), (1, 2), (5, 6), (3, 4)]

result = remove\_duplicates(tuples\_list)

print("List with duplicates removed:", result)

Find the frequency of each distinct element in an unsorted list using a dictionary:

def element\_frequency(lst):

freq\_dict = {}

for element in lst:

found = False

for key in freq\_dict:

if key == element:

freq\_dict[key] += 1

found = True

break

if not found:

freq\_dict[element] = 1

return freq\_dict

unsorted\_list = [1, 2, 3, 4, 2, 3, 1, 4, 5]

frequency = element\_frequency(unsorted\_list)

print("Frequency of each element:", frequency)

Check whether two words are anagrams using a dictionary:

def are\_anagrams(word1, word2):

if len(word1) != len(word2):

return False

char\_count1 = {}

char\_count2 = {}

for char in word1:

if char not in char\_count1:

char\_count1[char] = 1

else:

char\_count1[char] += 1

for char in word2:

if char not in char\_count2:

char\_count2[char] = 1

else:

char\_count2[char] += 1

return char\_count1 == char\_count2

word1 = input("Enter the first word: ")

word2 = input("Enter the second word: ")

print("Are the words anagrams?", are\_anagrams(word1, word2))

Set-3:

1. Check Whether a String is a Palindrome or not Using Recursion.

2. Find the GCD of two numbers using recursion

3. Count the number of words in a text file

4. Read a text file and print all numbers present in the text file.

5. Count the occurrences of a word in a text file.

6. Print all the .mp3 files of current working directory.

7. Copy each line of one file into another file if the line does not begin with ‘#’.

8. Find out every common word in the two files

Check Whether a String is a Palindrome or not Using Recursion:

def is\_palindrome(string):

if len(string) <= 1:

return True

if string[0] != string[-1]:

return False

return is\_palindrome(string[1:-1])

user\_input = input("Enter a string: ")

if is\_palindrome(user\_input):

print("The string is a palindrome.")

else:

print("The string is not a palindrome.")

Find the GCD of two numbers using recursion:

def gcd(a, b):

if b == 0:

return a

return gcd(b, a % b)

num1 = int(input("Enter first number: "))

num2 = int(input("Enter second number: "))

print("GCD of", num1, "and", num2, "is:", gcd(num1, num2))

Count the number of words in a text file:

file = open("text\_file.txt", "r")

word\_count = 0

for line in file:

for word in line.split():

word\_count += 1

file.close()

print("Number of words in the text file:", word\_count)

1. Read a text file and print all numbers present in the text file:

python  
file = open("text\_file.txt", "r")

numbers = []

for line in file:

for word in line.split():

if word.isdigit():

numbers.append(int(word))

file.close()

print("Numbers present in the text file:", numbers)

Count the occurrences of a word in a text file:

file = open("text\_file.txt", "r")

word\_to\_count = input("Enter the word to count: ")

count = 0

for line in file:

for word in line.split():

if word == word\_to\_count:

count += 1

file.close()

print("Occurrences of '{}' in the text file: {}".format(word\_to\_count, count))

Print all the .mp3 files of the current working directory:

import os

for file in os.listdir():

if file.endswith(".mp3"):

print(file)

Copy each line of one file into another file if the line does not begin with '#':

input\_file = open("input.txt", "r")

output\_file = open("output.txt", "w")

for line in input\_file:

if not line.startswith("#"):

output\_file.write(line)

input\_file.close()

output\_file.close()

Find out every common word in the two files:

file1 = open("file1.txt", "r")

file2 = open("file2.txt", "r")

common\_words = []

for line1 in file1:

for word1 in line1.split():

file2.seek(0)

for line2 in file2:

for word2 in line2.split():

if word1 == word2 and word1 not in common\_words:

common\_words.append(word1)

file1.close()

file2.close()

print("Common words in the two files:", common\_words)