**## CA3 Suggestions**

**#Q.1. WAP to count the occurrence of each character in your name.**

st = input("Enter the word to check: ")

freq = []

for i in st:

for c in freq:

if c[0]==i:

c[1]+=1

break

else:

freq.append([i,1])

print("Letter frequencies:",freq)

**#Q.2. WAP to implement Selection Sort.**

def selection\_sort(array):

length = len(array)

for i in range(length-1):

minIndex = i

for j in range(i+1, length):

if array[j]<array[minIndex]:

minIndex = j

array[i], array[minIndex] = array[minIndex], array[i]

return array

array = [72,4,10,38,2]

print("The sorted array is: ", selection\_sort(array))

**#Q.3. WAP to check whether a number is Krishnamurthy or not.**

def factorial(n):

fact=1

for i in range(1,(n+1)):

fact=fact\*i

return fact

if \_\_name\_\_== "\_\_main\_\_":

n=int(input("Enter the number to check: "))

s=0

temp=n

while(n!=0):

r=n%10

s=s+factorial(r)

n=n//10

if(s==temp):

print("%d is a Krishnamurthy Number."%temp)

else:

print("%d is not a Krishnamurthy Number."%temp)

**#Q.4. WAP to check whether a given string is palindrome or not.**

str1 = input("Enter a string: ")

str2 = str1.replace(" ", "").lower()

if str2 == str2[::-1]:

print(f"{str1} is a Palindrome String.")

else:

print(f"{str1} is not a Palindrome String.")'''

**#Q.5. Explain the concept of Bitwise Operator in Python.**

Bitwise Operator: & (AND), | (OR), ^ (X-OR), ~ (Bitwise NOT).

Bitwise operators are used to perform operations on integers at the binary level. They operate directly on the binary representations of numbers. These operators are commonly used in low-level programming, bit manipulation, and situations requiring performance optimizations.

**Bitwise Operators Overview**

Here is a list of Python's bitwise operators and their functions:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Symbol** | **Description** |
| AND | & | Sets each bit to 1 if both bits are 1. |
| OR | ` | ` |
| XOR | ^ | Sets each bit to 1 if only one bit is 1. |
| NOT | ~ | Inverts all the bits. |
| Left Shift | << | Shifts bits to the left, filling with 0. |
| Right Shift | >> | Shifts bits to the right, preserving the sign bit for signed integers. |

**Examples of Bitwise Operators**

**1. Bitwise AND (&)**

Performs a logical AND operation on each pair of corresponding bits.

**Example:**

a = 5 # Binary: 0101

b = 3 # Binary: 0011

result = a & b

print(result) # Output: 1 (Binary: 0001)

**2. Bitwise OR (|)**

Performs a logical OR operation on each pair of corresponding bits.

**Example:**

a = 5 # Binary: 0101

b = 3 # Binary: 0011

result = a | b

print(result) # Output: 7 (Binary: 0111)

**3. Bitwise XOR (^)**

Performs a logical XOR operation. The result is 1 if the corresponding bits are different.

**Example:**

a = 5 # Binary: 0101

b = 3 # Binary: 0011

result = a ^ b

print(result) # Output: 6 (Binary: 0110)

**4. Bitwise NOT (~)**

Inverts all the bits (1's complement). It flips all 0s to 1s and 1s to 0s. The result is equivalent to -(n+1) for integers.

**Example:**

a = 5 # Binary: 0101

result = ~a

print(result) # Output: -6 (Binary: -0110 in two's complement)

**5. Left Shift (<<)**

Shifts bits to the left by a specified number of positions, filling the vacant positions with 0s. Equivalent to multiplying the number by 2^n.

**Example:**

a = 5 # Binary: 0101

result = a << 2

print(result) # Output: 20 (Binary: 10100)

**6. Right Shift (>>)**

Shifts bits to the right by a specified number of positions. The vacant positions are filled based on the sign of the number (0 for positive, 1 for negative).

**Example:**

a = 20 # Binary: 10100

result = a >> 2

print(result) # Output: 5 (Binary: 0101)

**#Q.6. Write a python program to check whether a number is Armstrong or not.**

n=int(input("Enter the number to check: "))

temp=n

cnt=0

s=0

while(n!=0):

cnt=cnt+1

n=n//10

n=temp

while(n!=0):

r=n%10

s=s+r\*\*cnt

n=n//10

if(temp==s):

print("%d is a Armstrong Number."%temp)

else:

print("%d is not a Armstrong Number."%temp)

**#Q.7. WAP to find GCD of two numbers using recursion.**

def gcd(a,b):

if b==0:

return a

return gcd(b,a%b)

x,y = [int(c) for c in input("Enter two numbers:").split()]

res=gcd(x,y)

print(f"The GCD of {x} and {y} is {res}.")