**Python programming**

**CSA0851**

1. write a program in Python to convert decimal numbers to binary input decimal is 65 output binary is 1000001

def decimal\_to\_binary(n):

return bin(n). replace ("0b", "")

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

binary = decimal\_to\_binary(num)

print (f"The binary representation of {num} is {binary}")

INPUT: OUTPUT:

Enter a decimal number: 65

The binary representation of 65 is 1000001

1. write a program to display the diagonal elements in a matrix array and also find the sum of them

matrix= [

[1,2,3],

[4,5,6],

[7,8,9],]

size=Len(matrix)

diagonal1=[matrix[i][i] for i in range(size)]

diagonal2=[matrix[i][size-i-1] for i in range(size)]

print ("1St diagonal is:", diagonal1)

print ("2nd diagonal is:", diagonal2)

print ("Sum of 1st diagonal is:", sum(diagonal1))

print ("sum of 2nd diagonal is:", sum(diagonal2))

INPUT: OUTPUT:

matrix=[ 1st diagonal=[1, 5, 9]

[1,2,3], 2nd diagonal=[3, 5, 7]

[4,5,6], sum of 1st diagonal: 15

[7,8,9], sum of 2nd diagonal:15]

1. Write a Python program to take two arrays as input merge and concatenate two arrays and store the result in a third array

Take two arrays as input

array1 = input("Enter elements of Array 1 (space-separated): ").split()

array2 = input("Enter elements of Array 2 (space-separated): ").split()

# Convert input strings to desired data type (e.g., int, float)

array1 = [int(x) for x in array1]

array2 = [int(x) for x in array2]

# Merge and concatenate two arrays using +

def merge\_arrays(array1, array2):

return array1 + array2

# Store the result in a third array

merged\_array = merge\_arrays(array1, array2)

print("Array 1:", array1)

print("Array 2:", array2)

print("Merged Array:", merged\_array)

# Test the program

Input:

Enter elements of Array 1 (space-separated): 1 2 3 4 5

Enter elements of Array 2 (space-separated): 6 7 8 9 10

Output:

Array 1: [1, 2, 3, 4, 5]

Array 2: [6, 7, 8, 9, 10]

Merged Array: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

4. Define a class taximeter with the following description:

class Taximeter:

def \_\_init\_\_(self, base\_fare, per\_km\_rate, per\_minute\_rate):

self.base\_fare = base\_fare

self.per\_km\_rate = per\_km\_rate

self.per\_minute\_rate = per\_minute\_rate

self.total\_distance = 0

self.total\_time = 0

self.total\_fare = 0

def calculate\_fare(self, distance, time):

self.total\_distance += distance

self.total\_time += time

self.total\_fare = self.base\_fare + (self.per\_km\_rate \* self.total\_distance) + (self.per\_minute\_rate \* self.total\_time)

def display\_fare(self):

print(f"Total Distance: {self.total\_distance} km")

print(f"Total Time: {self.total\_time} minutes")

print(f"Total Fare: ${self.total\_fare:.2f}")

def reset(self):

self.total\_distance = 0

self.total\_time = 0

self.total\_fare = 0

taxi = Taximeter(2.0, 1.5, 0.25)

taxi.calculate\_fare(10, 30)

taxi.display\_fare()

taxi.reset()

taxi.display\_fare()

# Output:

Total Distance: 10 km

Total Time: 30 minutes

Total Fare: $17.00

Total Distance: 0 km

Total Time: 0 minutes

Total Fare: $0.00

print(f"Total Time: {self.total\_time} minutes")

print(f"Total Fare: ${self.total\_fare:.2f}")

1. Print the given number is a perfect number or not

def is\_perfect\_number(n):

Check if a number is perfect.

Parameters:

n: int

The number to check.

Returns:

bool

True if the number is perfect, False otherwise.

if n < 1:

return False

sum\_divisors = 0

for i in range(1, n):

if n % i == 0:

sum\_divisors += i

return sum\_divisors == n

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

if is\_perfect\_number(num):

print(f"{num} is a perfect number.")

else:

print(f"{num} is not a perfect number.")

# Enter a number: 6

# 6 is a perfect number.

# Enter a number: 28

# 28 is a perfect number.

# Enter a number: 12

# 12 is not a perfect number.

1. Two integers’ arrays nums1 and nums2, each element in the result appears as many times in both array's input and output

def intersect(nums1, nums2):

count = {}

res = []

for num in nums1:

if num in count:

count[num] += 1

else:

count[num] = 1

for num in nums2:

if num in count and count[num] > 0:

res.append(num)

count[num] -= 1

return res

# Test the function

nums1 = [1, 2, 2, 1]

nums2 = [2, 2]

print(intersect(nums1, nums2)) # Output: [2, 2]

nums1 = [4, 9, 5]

nums2 = [9, 4, 9, 8, 4]

print(intersect(nums1, nums2)) # Output: [4, 9]

1. Write Integers number sorted in non-decreasing order ending position.

Program

def search\_range(nums, target):

def binary\_search(left, right, find\_first):

while left <= right:

mid = (left + right) // 2

if nums[mid] == target:

if find\_first:

if mid == 0 or nums[mid - 1] < target:

return mid

right = mid - 1

else:

if mid == len(nums) - 1 or nums[mid + 1] > target:

return mid

left = mid + 1

elif nums[mid] < target:

left = mid + 1

else:

right = mid - 1

return -1

first\_position = binary\_search(0, len(nums) - 1, True)

if first\_position == -1:

return [-1, -1]

last\_position = binary\_search(first\_position, len(nums) - 1, False)

return [first\_position, last\_position]

# Test the function

nums = [5, 7, 7, 8, 8, 10]

target = 8

print(search\_range(nums, target)) # Output: [3, 4]

nums = [5, 7, 7, 8, 8, 10]

target = 6

print(search\_range(nums, target)) # Output: [-1, -1]

nums = [5, 7, 7, 8, 8, 10]

target = 5

print(search\_range(nums, target)) # Output: [0, 0]

Input:

nums = [5, 7, 7, 8, 8, 10]

target = 8

Output:[3, 4]

Input:

nums = [5, 7, 7, 8, 8, 10]

target = 6

Output:

[-1, -1]

1. Write a program for a Sum of the digits of the number

def sum\_of\_digits(n):

sum = 0

while n:

sum += n % 10

n //= 10

return sum

# Test the function

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

print("Sum of digits:", sum\_of\_digits(num))

outputs

Enter a number: 123

A sum of digits: 6