***IMP QUESTIONS***

1. **Discount**

**code:-**

original\_price = float(input("Enter the original price: "))

discount\_percentage = float(input("Enter the discount percentage: "))

discount\_amount = original\_price \* (discount\_percentage / 100)

discounted\_price = original\_price - discount\_amount

print("Original Price: $", original\_price)

print("Discount Percentage: ", discount\_percentage, "%")

print("Discount Amount: $", discount\_amount)

print("Discounted Price: $", discounted\_price)

1. **Sum of rows and columns in a matrix**

**code:-**

matrix = [

[1, 2, 3],

[4, 5, 6],

[7, 8, 9]

]

Row\_sums = [sum(i) for i in matrix]

columns = [sum(j) for j in zip(\*matrix)]

diagonal=sum(matrix[i][i] for i in range(len(matrix)))

print("Sum of rows:", row\_sums)

print("Sum of columns:", columns)

print("sum of diagonal:",diagonal)

1. **In daily shares trading a buyer buyes shares in the morning and sells them on the same day. if the trader allowed to make atmost 2 transactions in a day, where as the second transaction can only start after the first transaction is completed(buy-sell-buy-sell).give stact prices throughout the day. find out maximum profit that a share trader could have made.**

**code:-**

prices = [3, 3, 5, 0, 0, 3, 1, 4]

n = len(prices)

profit = 0

for i in range(1, n):

if prices[i] > prices[i - 1]:

profit += prices[i] - prices[i - 1]

print(profit)

1. **U are climbing a staircase. it takes n steps to reach top. each time u can either climb 1 and 2 steps. in how many distinct ways can you climb to top . , give with out using functions**

**code:-**

n = 5

a, b = 1, 2

for \_ in range(n - 2):

a, b = b, a + b

print(a)

1. **Geometric progression**

**code:-**

a = 1

r = 3

n = 5

progression = []

for i in range(n):

term = a \* (r \*\* i)

progression.append(term)

print("Geometric progression:", progression)

1. **Spiral matrix**

**code:-**

matrix=[

[1,4,7,9],

[9,3,6,10],

[2,5,8,11],

[12,15,13,80]

]

result=[]

while matrix:

result+=matrix.pop(0)

if matrix:

matrix=list(zip(\*matrix))[::-1]

print(result)

1. **Lucky number**

**code:-**

n=123

sum=0

p=1

while(n>0):

m=n%10

sum=sum+m

p=p\*m

n=n//10

print("sum:",sum)

print("product:",p)

if(sum==p):

print(" lucky number")

else:

print(" not lucky number**")**

1. **3 matrix addition**

**code:-**

import numpy as np

a=np.array([[1,2,3],

[4,5,6],

[7,8,9]])

b=np.array([[1,2,3],

[4,5,6],

[7,8,9]])

c=np.array([[1,2,3],

[4,5,6],

[7,8,9]])

sum=a+b+c

mult=np.dot(np.dot(a,b),c)

print(sum)

print(mult)

**Or**

a=[[1,2],[3,4]]

b=[[6,7],[8,9]]

c=[[3,6],[7,9]]

d=[[0,0],[0,0]]

e=[[0,0],[0,0]]

for i in range(len(a)):

for j in range(len(b)):

d[i][j]=a[i][j]+b[i][j]

for i in d:

print(i)

print("\n")

for i in range(len(d)):

for j in range(len(c)):

e[i][j]=d[i][j]+c[i][j]

for i in e:

print(i)

1. **Binary to decimal**

**code:-**

a="1100100"

b=int(a,2)

print(b)

1. **Calender**

**code:-**

import calendar

y=2024

m=2

c=calendar.month(y,m)

print(c)

1. **Date is given print day**

**code:-**

from datetime import datetime

a = input("Enter date (YYYY-MM-DD): ")

try:

p = datetime.strptime(a, "%Y-%m-%d") # Corrected format string

f = p.strftime("%A")

print(f)

except ValueError:

print("Enter the date in correct manner")

1. **Check given string is a valid password or nat**

**code:-**

import re

password="Sai@123sai"

if len(password)<8:

print("invalid : it must contain >8 characters")

elif not re.search(r'[A-Z]',password):

print("invalid : it must contain 1 capital letter")

elif not re.search(r'\d',password):

print("inalid : it must contain one numerical")

elif not re.search(r'[a-z]',password):

print("invalid : it must contain on small letter")

elif not re.search(r'[@#$^%!?&]',password):

print("invalid : it must contain one special character")

else:

print("valid password")

1. **Find the root of quardratic equation**

**code:-**

a=1

b=5

c=6

root=b\*\*2-4\*a\*c

if root>=0:

root1=(-b+(root\*\*0.5))/(2\*a)

root2=(-b-(root\*\*0.5))/(2\*a)

print("root1:-",root1)

print("root2:-",root2)

else:

print("the roots are complex numbers")

1. **Check if given string is valid e-mail or not**

**code:-**

import re

def is\_valid\_email(email):

pattern=r'^[\w\.-]+@[a-zA-z0-9-]+\.[a-zA-z]{2,}$'

if re.match(pattern, email):

return True

else:

return False

email="sai@email.com"

if is\_valid\_email(email):

print("vaild mail")

else:

print("not vaild mail")

1. **Tech number**

**code:-**

x=2025

y=str(x)

length=len(y)

if length%2==0:

mid=length//2

a=int(y[:mid])

b=int(y[mid:])

c=(a+b)\*\*2

if c==x:

print("tech")

else:

print("not tech")

else:

print("not")

1. **Prime numbers in range**

**code:-**

for i in range(1,100,1):

sum=0

for j in range(1,i):

if i%j==0:

sum=sum+i

if sum==i:

print(i,"is prime")

else:

print(i,"is not prime")

1. **Perfect number in range**

**code:-**

for i in range(1,100,1):

sum=0

for j in range(1,i):

if i%j==0:

sum=sum+j

if sum==i:

print(i,"is perfect")

1. **Gcd of 3 numbers**

**code:-**

a=15

b=10

c=20

lcm=0

gcd=0

gcd1=0

for i in range(1,a+1 or b+1):

if(a%i==0 and b%i==0):

gcd=i

lcm=a\*b//gcd

for i in range(1,gcd+1 or c+1):

if(gcd%i==0 or c%i==0):

gcd1=i

print("gcd1 :",gcd1)

print("lcm :",lcm)

1. **Lcm and gcd**

**code:-**

a=12

b=20

gcd=0

lcm=0

for i in range(1,a+1 or b+1,1):

if(a%i==0 and b%i==0):

gcd=i

print("gcd : ",gcd)

lcm=a\*b//gcd

print("lcm : ",lcm)

***PREVIOUS SEM EXAM QUESTIONS***

1. **Sum of squares (1\*2+2\*2+3\*3)**

**Code:-**

n=5

sum=0

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

sum=sum+i\*\*2

print(sum)

1. **Least significant digit and most significant digit**

**code:-**

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

least\_significant\_digit = num % 10

while num >= 10:

num //= 10

most\_significant\_digit = num

print("Least significant digit:", least\_significant\_digit)

print("Most significant digit:", most\_significant\_digit)

1. **sum of DIAGONAL**

**Code:-**

matrix=[

[1,2,6],

[4,3,5],

[7,9,8]

]

rowsum=[sum(i) for i in matrix]

columnsum=[sum(j) for j in matrix]

diagonalsum=sum(matrix[i][i] for i in range(len(matrix)))

print(rowsum)

print(columnsum)

print(diagonalsum)

1. **peak elements**

**code:-**

array = [1, 3, 20, 4, 1, 0]

peak\_element = None

for i in range(1, len(array) - 1):

if array[i] >= array[i - 1] and array[i] >= array[i + 1]:

peak\_element = array[i]

break

if peak\_element is not None:

print("Peak element:", peak\_element)

else:

print("No peak element found")

1. **SORT**

**code:-**

a=[1,8,4,90,2]

a.sort()

print(a)

1. **Sqrt and cube root**

**code:-**

import math

number = 64

square\_root = math.sqrt(number)

print("Square root of", number, ":", square\_root)

cube\_root = math.cbrt(number)

print("Cube root of", number, ":", cube\_root)

**Or**

num = 64

square\_root = num \*\* 0.5

print("Square root of", num, ":", square\_root)

cube\_root = num \*\* (1/3)

print("Cube root of", num, ":", cube\_root)

1. **array mean, mode, median**

**code:-**

import numpy as np

my\_list = [7, 2, 5, 5, 3, 9, 5, 8, 4]

mean = np.mean(my\_list)

median = np.median(my\_list)

mode = np.argmax(np.bincount(my\_list))

print("Mean:", mean)

print("Median:", median)

print("Mode:", mode)

**Or**

import statistics

my\_list = [7, 2, 1, 6, 3, 9, 5, 8, 4]

mean = statistics.mean(my\_list)

median = statistics.median(my\_list)

mode = statistics.mode(my\_list)

print("Mean:", mean)

print("Median:", median)

print("Mode:", mode)

1. **123 combinations**

**code:-**

1. **Cube and square of number**

**code:-**

a=2

sq=a\*\*2

cb=a\*\*3

print(sq,cb)

1. **3 or 5 multiples range 12 and sum them**

**code:-**

a=12

sum=0

res=[]

for i in range (1,a,1):

if i%3==0 or i%5==0:

sum=sum+i

res.append(i)

print(sum)

print(res)

1. **Sum of series 1fact/2+3fact**

**code:-**

1. **3 matrixs multiplication**

**code:-**

import numpy as np

a=np.array([[1,2,3],

[4,5,6],

[7,8,9]])

b=np.array([[1,2,3],

[4,5,6],

[7,8,9]])

c=np.array([[1,2,3],

[4,5,6],

[7,8,9]])

sum=a+b+c

mult=np.dot(np.dot(a,b),c)

print(sum)

print(mult)

**Or**

a=[[1,1],[1,1]]

b=[[2,2],[2,2]]

c=[[2,2],[2,2]]

d=[[0,0],[0,0]]

e=[[0,0],[0,0]]

for i in range(len(a)):

for j in range(len(b)):

for k in range(len(b)):

d[i][j]+=a[i][k]\*b[k][j]

for i in d:

print(i)

print("\n")

for i in range(len(d)):

for j in range(len(c)):

for k in range(len(c)):

e[i][j]+=d[i][k]\*c[k][j]

for i in e:

print(i)

1. **Insert an element in 1d array at specific index**

**Or**

**Insert element and at specific position**

**code:-**

arr = [1, 2, 3, 4, 5]

index = 2

element = 100

for i in range(len(arr) - 1, index - 1, -1):

arr[i] = arr[i - 1]

arr[index] = element

print(arr)

**By deleting the index**

arr = [1, 2, 3, 4, 5]

index = 2

element = 100

arr[index] = element

print(arr)

1. **An array with random numbers find the sum and min of them**

**code:-**

a=[2,4,6,8,2]

b=sum(a)

c=min(a)

d=max(a)

print(b,c,d)

1. **Leap year**

**code:-**

n=int(input("entter n: "))

if(n%4==0 and n%100!=0 or n%400==0):

print(n,"leapyear")

else:

print(n,"not a leapyear")

1. **Sum of square of given range**

**code:-**

a=4

sum=0

for i in range(a):

b=i\*\*2

sum+=b

print(b)

print(sum)

1. **Create a class on showroom with members name, phone number, cost , discount, after discount amount**

**code:-**

class Showroom:

def \_\_init\_\_(self, name, phone\_number, cost, discount):

self.name = name

self.phone\_number = phone\_number

self.cost = cost

self.discount = discount

self.after\_discount\_amount = cost - (cost \* discount / 100)

showroom = Showroom("XYZ Showroom", "1234567890", 1000, 10)

print("Showroom Name:", showroom.name)

print("Phone Number:", showroom.phone\_number)

print("Cost:", showroom.cost)

print("Discount:", showroom.discount)

print("After Discount Amount:", showroom.after\_discount\_amount)

**Or without using functions**

class Showroom:

name = "XYZ Showroom"

phone\_number = "1234567890"

cost = 1000

discount = 10

after\_discount\_amount = cost - (cost \* discount / 100)

print("Showroom Name:", Showroom.name)

print("Phone Number:", Showroom.phone\_number)

print("Cost:", Showroom.cost)

print("Discount:", Showroom.discount)

print("After Discount Amount:", Showroom.after\_discount\_amount)

1. **Search a number that present in matrix**

**code:-**

a=[[1,2,3],[4,5,6],[7,8,9]]

found=False

target=8

for i in a:

if target in i:

found=True

print(found)

1. **Adding two strings,array**

**code:-**

string1 = "Hello"

string2 = "World"

concatenated\_string = string1 + string2

print(concatenated\_string)

array1 = [1, 2, 3]

array2 = [4, 5, 6]

concatenated\_array = array1 + array2

print(concatenated\_array)

1. **Sum of rows column and diagonal**

**code:-**

matrix=[

[1,2,6],

[4,3,5],

[7,9,8]

]

rowsum=[sum(i) for i in matrix]

columnsum=[sum(j) for j in matrix]

diagonalsum=sum(matrix[i][i] for i in range(len(matrix)))

print(rowsum)

print(columnsum)

print(diagonalsum)

1. **sum of positive number in list (-1)**

**code:-**

my\_list = [1, 2, -3, 4, 5, 6, 7, -2]

positive\_sum = 0

negative\_sum=0

for num in my\_list:

if num == -1:

break

elif num > 0:

positive\_sum += num

else:

negative\_sum+=num

print("Sum of positive numbers until -1:", positive\_sum)

print("Sum of negative\_sum :", negative\_sum)

1. **decimal to binary**

**code:-**

a=100

print(bin(a)[2::])