**1.Swapping 2 no**

def swap\_and\_check(num1, num2):

temp = num1

num1 = num2

num2 = temp

if num1 > 0:

print(f"{num1} is a positive number.")

elif num1 < 0:

print(f"{num1} is a negative number.")

else:

print("The first number is zero.")

if num2 > 0:

print(f"{num2} is a positive number.")

elif num2 < 0:

print(f"{num2} is a negative number.")

else:

print("The second number is zero.")

num1 = float(input("Enter the first number: "))

num2 = float(input("Enter the second number: "))

print(f"Before swapping: num1 = {num1}, num2 = {num2}")

swap\_and\_check(num1, num2)

print(f"After swapping: num1 = {num2}, num2 = {num1}")

**2.string palidrom**

def is\_palindrome(value):

return str(value) == str(value)[::-1]

def factorial(n):

if n == 0 or n == 1:

return 1

else:

return n \* factorial(n - 1)

def main():

while True:

print("Menu:")

print("1. Check if a number is palindrome")

print("2. Check if a string is palindrome")

print("3. Find the factorial of a number")

print("4. Exit")

choice = int(input("Enter your choice: "))

if choice == 1:

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

if is\_palindrome(num):

print(f"{num} is a palindrome.")

else:

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

elif choice == 2:

string = input("Enter a string: ")

if is\_palindrome(string):

print(f"{string} is a palindrome.")

else:

print(f"{string} is not a palindrome.")

elif choice == 3:

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

fact = factorial(num)

print(f'The factorial of {num} is {fact}.')

elif choice == 4:

print("Exiting the program.")

break

else:

print("Invalid choice. Please select a valid option.")

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

main()

**3.use of list in python**

def main():

lst = []

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

for i in range(n):

lst.append(int(input("Enter element: ")))

print("Original list:", lst)

even\_lst = []

odd\_lst = []

for i in lst:

if i % 2 == 0:

even\_lst.append(i)

else:

odd\_lst.append(i)

print("Even list:", even\_lst)

print("Odd list:", odd\_lst)

merged\_lst = even\_lst + odd\_lst

merged\_lst.sort()

print("Merged and sorted list:", merged\_lst)

num1 = int(input("Enter the number to be replaced: "))

merged\_lst[0] = num1

del merged\_lst[len(merged\_lst) // 2]

print("Updated list:", merged\_lst)

max\_num = max(merged\_lst)

min\_num = min(merged\_lst)

print("Max element:", max\_num)

print("Min element:", min\_num)

names = []

n = int(input("Enter the number of names to add: "))

for i in range(n):

names.append(input("Enter name: "))

merged\_lst += names

if "python" in merged\_lst:

print("Python is present in the list.")

else:

print("Python is not present in the list.")

if \_\_name\_\_ == '\_\_main\_\_':

main()

**4.use of tuple**

from pprint import pprint

t =()

l = []

def menu():

print("\n///MENU///")

print("1. Create & display tuple")

print("2. Display details of a student 'Python'")

print("3. Sort nested tuple")

print("4. Exit")

return (int(input("\nEnter your choice: ")))

def read():

global t

n = int(input("\nEnter the no. of students: "))

for i in range(0,n):

print("\nEnter details for student", i+1, ": ")

name = input("\nEnter Name: ")

rno = input("Enter Roll no: ")

print("Enter the Marks for 5 subjects: ")

marks = [int(x) for x in input().split()]

t1 = (rno, name, marks)

l.append(t1)

t = tuple(l)

print("\nDisplaying the student details:-")

for i in range(0,n):

print("\nRoll no: ", t[i][0])

print("Name: ", t[i][1])

print("Marks: ", t[i][2])

print("\n------------------------------")

return n

def find(n):

flag = 0

for x in t:

if x[1]=='Python':

print("\nThe name 'Python' found in tuple")

print(x)

flag = 1

break

if flag==0:

print("\nThe name 'Python' not found")

def sort():

sortedt = sorted(t, key = lambda x : x[1])

print("\nThe sorted tuple is:-\n")

pprint(sortedt)

c=0

while(c!=4):

c=menu()

if c==1:

n=read()

elif c==2:

find(n)

elif c==3:

sort()

elif c!=4:

print("\nIncorrect choice")

**5.demonstrate classes**

class Dog:

# Class Variable

animal = 'dog'

# The init method or constructor

def \_\_init\_\_(self, breed, color):

# Instance Variable

self.breed = breed

self.color = color

# Objects of Dog class

Rodger = Dog("Pug", "brown")

Buzo = Dog("Bulldog", "black")

print('Rodger details:')

print('Rodger is a', Rodger.animal)

print('Breed: ', Rodger.breed)

print('Color: ', Rodger.color)

print('\nBuzo details:')

print('Buzo is a', Buzo.animal)

print('Breed: ', Buzo.breed)

print('Color: ', Buzo.color)

# Class variables can be accessed using class

# name also

print("\nAccessing class variable using class name")

print(Dog.animal)

**6.heritance in python**

class Person:

def \_\_init\_\_(self, id, name):

self.ID = id

self.name = name

def display(self):

print("\nID = ", self.ID)

print("\nName = ", self.name)

class Student(Person):

def \_\_init\_\_(self, x, y, id, name):

super().\_\_init\_\_(id, name)

#allows the child class to access the parent class's init() property.

self.sub1 = x

self.sub2 = y

def display(self):

print("\nStudent ID = ", self.ID)

print("Student Name = ", self.name)

print("Marks in subject 1 = ", self.sub1)

print("Marks in subject 2 = ", self.sub2)

class Sports:

def showsports(self, s=None):

self.sportsmarks = s

if s!=None:

print("\nSports marks = ", self.sportsmarks)

class Result(Student, Sports):

def \_\_init\_\_(self, id, name, x, y):

super().\_\_init\_\_(x, y, id, name)

def total(self):

if self.sportsmarks is not None:

sum1 = self.sub1 + self.sub2 + self.sportsmarks

else:

sum1 = self.sub1 + self.sub2

print("\nWe have, Total Marks = ", sum1, "\n")

print("Enter the student details:-")

id = input("\nEnter Student ID: ")

name = input("\nEnter Student Name: ")

m1 = int(input("\nEnter marks in first subject: "))

m2 = int(input("\nEnter marks in second subject: "))

obj = Result(id, name, m1, m2)

obj.display()

c = input("\nDo you want to enter sports marks? (y/n): ")

if c=='y':

sm = int(input("Enter sports marks: "))

obj.showsports(sm)

else:

obj.showsports()

obj.total()

**7.exceptional handeling**

def divide\_numbers(a, b):

try:

result = a / b

return result

except ZeroDivisionError:

print("Error: Division by zero is not allowed.")

except TypeError:

print("Error: Please provide two numerical values.")

except Exception as e:

print(f"An error occurred: {e}")

else:

print("Division successful!")

finally:

print("Execution complete.")

while True:

try:

num1 = float(input("Enter the first number: "))

num2 = float(input("Enter the second number: "))

print("Result:", divide\_numbers(num1, num2))

break

except ValueError:

print("Error: Please enter a valid number.")

**8.explore file directories**

def append\_data(file\_name, data):

try:

with open(file\_name, 'a') as file:

file.write(data)

print("Data appended successfully.")

except FileNotFoundError:

print("Error: File not found.")

except Exception as e:

print("Error:", e)

def display\_file(file\_name):

try:

with open(file\_name, 'r') as file:

content = file.read()

print("File content:")

print(content)

except FileNotFoundError:

print("Error: File not found.")

except Exception as e:

print("Error:", e)

file\_name = "data.txt"

data = "This is some appended data."

append\_data(file\_name, data)

display\_file(file\_name)

**9.explore file direct**

**a…**

file\_name = input("Enter the name of the file: ")

try:

    with open(file\_name, 'r') as file:

        # Read the content of the file

        content = file.read()

        # Count the number of lines, words, and characters

        lines = content.split('\n')

        num\_lines = len(lines)

        words = content.split()

        num\_words = len(words)

        num\_chars = len(content)

    print("Number of lines:", num\_lines)

    print("Number of words:", num\_words)

    print("Number of characters:", num\_chars)

except FileNotFoundError:

    print("File not found.")

except Exception as e:

    print("An error occurred:", str(e))

**B….**

import os

# Get the list of files in the current directory

files = os.listdir()

print("Files in the current directory:")

for file in files:

    if os.path.isfile(file):

        print(file)

**10.CREATE DATABASE**

import sqlite3

import sqlite3

conn = sqlite3.connect("mydatabase.db")

cursor = conn.cursor()

cursor.execute('''CREATE TABLE IF NOT EXISTS student (

id INTEGER PRIMARY KEY,

name TEXT,

age INTEGER,

grade TEXT

)''')

conn.commit()

def insert\_student(name, age, grade):

cursor.execute("INSERT INTO student (name, age, grade) VALUES (?, ?, ?)", (name, age, grade))

conn.commit()

def display\_students():

cursor.execute("SELECT \* FROM student")

students = cursor.fetchall()

for student in students:

print(f"ID: {student[0]}, Name: {student[1]}, Age: {student[2]}, Grade: {student[3]}")

def update\_student\_grade(student\_id, new\_grade):

cursor.execute("UPDATE student SET grade = ? WHERE id = ?", (new\_grade, student\_id))

conn.commit()

insert\_student("APEKSHA", 20, "A")

insert\_student("ADITYA", 22, "B")

print("Initial Student List:")

display\_students()

update\_student\_grade(2, "C")

print("\nUpdated Student List:")

display\_students()

conn.close()

**11.GUI TKINTER**

import tkinter as tk

import sqlite3

conn = sqlite3.connect("contacts.db")

cursor = conn.cursor()

cursor.execute('''CREATE TABLE IF NOT EXISTS contacts (

id INTEGER PRIMARY KEY,

name TEXT,

email TEXT

)''')

conn.commit()

def add\_contact():

name = name\_entry.get()

email = email\_entry.get()

cursor.execute("INSERT INTO contacts (name, email) VALUES (?, ?)", (name, email))

conn.commit()

update\_contact\_list()

name\_entry.delete(0, tk.END)

email\_entry.delete(0, tk.END)

def update\_contact\_list():

contact\_listbox.delete(0, tk.END)

cursor.execute("SELECT \* FROM contacts")

contacts = cursor.fetchall()

for contact in contacts:

contact\_listbox.insert(tk.END, f"{contact[1]} - {contact[2]}")

root = tk.Tk()

root.title("Contact Manager")

name\_label = tk.Label(root, text="Name:", width=10)

name\_entry = tk.Entry(root, width=30)

email\_label = tk.Label(root, text="Email:", width=10)

email\_entry = tk.Entry(root, width=30)

add\_button = tk.Button(root, text="Add Contact", command=add\_contact)

contact\_listbox = tk.Listbox(root, width=40, height=10)

name\_label.grid(row=0, column=0, padx=10, pady=5)

name\_entry.grid(row=0, column=1, padx=10, pady=5)

email\_label.grid(row=1, column=0, padx=10, pady=5)

email\_entry.grid(row=1, column=1, padx=10, pady=5)

add\_button.grid(row=2, column=0, columnspan=2, padx=10, pady=10)

contact\_listbox.grid(row=3, column=0, columnspan=2, padx=10, pady=5)

root.mainloop()

conn.close()

**12.DATA SERIES AND FRAMES**

import pandas as pd

data\_series = pd.Series([10, 20, 30, 40, 50])

print("Data Series:")

print(data\_series)

print()

print("Accessing elements in a Data Series:")

print(data\_series[0])

print(data\_series[2:4])

print()

data = {

"Name": ['John', 'Emma', 'Tom', 'Lisa', 'Mike'],

'Age': [25, 30, 28, 35, 32],

'City': ['New York', 'London', 'Paris', 'Tokyo', 'Sydney']

}

data\_frame = pd.DataFrame(data)

print("Data Frame:")

print(data\_frame)

print()

print("Accessing columns in a Data Frame:")

print(data\_frame['Name'])

print(data\_frame[['Name', 'Age']])

print()

print("Accessing rows in a Data Frame:")

print(data\_frame.loc[0])

print(data\_frame.loc[1:3])

print("Filtering rows in a Data Frame:")

filtered\_data\_frame = data\_frame[data\_frame['Age'] > 28]

print(filtered\_data\_frame)