**Task#1 2403A51286**

write a python code that performs sorting of a list using both the bubble sort algorithm and python's built\_in\_sort() function.

**CODE:**

import time

def bubble\_sort(arr):

    n = len(arr)

    for i in range(n):

        for j in range(0, n-i-1):

            if arr[j] > arr[j+1]:

                arr[j], arr[j+1] = arr[j+1], arr[j]

    return arr

# Create a sample list

my\_list = [64, 34, 25, 12, 22, 11, 90]

# Bubble sort

start\_time = time.time()

bubble\_sorted\_list = bubble\_sort(my\_list.copy()) # Use a copy to keep original list

bubble\_sort\_time = time.time() - start\_time

print("Original list:", my\_list)

print("Bubble sorted list:", bubble\_sorted\_list)

print("Bubble sort time:", bubble\_sort\_time)

# Python's built-in sort

start\_time = time.time()

built\_in\_sorted\_list = sorted(my\_list)

built\_in\_sort\_time = time.time() - start\_time

print("\nOriginal list:", my\_list) # Original list is unchanged

print("Built-in sorted list:", built\_in\_sorted\_list)

print("Built-in sort time:", built\_in\_sort\_time)

OUTPUT:

Original list: [64, 34, 25, 12, 22, 11, 90]

Bubble sorted list: [11, 12, 22, 25, 34, 64, 90]

Bubble sort time: 0.00010085105895996094

Original list: [64, 34, 25, 12, 22, 11, 90]

Built-in sorted list: [11, 12, 22, 25, 34, 64, 90]

Built-in sort time: 8.416175842285156e-05

Diff btw Built-in sorted and Bubble sort

* **Bubble Sort:** This is a simple sorting algorithm. It repeatedly steps through the list, compares adjacent elements and swaps them if they are in the wrong order. As you can see from the Bubble sort time, it took slightly longer than the built-in sort for this specific list. Bubble sort is generally less efficient for larger lists compared to more advanced algorithms.
* **Python's Built-in sorted():** This function uses Timsort, which is a hybrid stable sorting algorithm derived from merge sort and insertion sort. As shown by the Built-in sort time, it was faster than bubble sort for this list. Python's built-in sort is highly optimized and generally the preferred method for sorting in Python.

**Task#2**

Generate a python function that takes a string returns:

The number of vowels, the number of constants, the number of digits in the string.

**CODE:**

def count\_chars(input\_string):

    """

    Counts the number of vowels, consonants, and digits in a string.

    Args:

        input\_string: The string to analyze.

    Returns:

        A tuple containing:

            - The number of vowels.

            - The number of consonants.

            - The number of digits.

    """

    vowels = "aeiouAEIOU"

    consonants = "bcdfghjklmnpqrstvwxyzBCDFGHJKLMNPQRSTVWXYZ"

    digits = "0123456789"

    vowel\_count = 0

    consonant\_count = 0

    digit\_count = 0

    for char in input\_string:

        if char in vowels:

            vowel\_count += 1

        elif char in consonants:

            consonant\_count += 1

        elif char in digits:

            digit\_count += 1

    return vowel\_count, consonant\_count, digit\_count

# Example usage:

test\_string = "Hello World 123!"

vowels, consonants, digits = count\_chars(test\_string)

print(f"String: {test\_string}")

print(f"Number of vowels: {vowels}")

print(f"Number of consonants: {consonants}")

print(f"Number of digits: {digits}")

**OUTPUT:**

String: Hello World 123!

Number of vowels: 3

Number of consonants: 7

Number of digits: 3

**Task#3**

generate a python program that performs file handiling. craete a text file. write simple text. read and display the content expected output is functional code that caretes a txt.file , writes content to it and reads it back

CODE:

# Define the filename

filename = "my\_sample\_file.txt"

# --- Create and Write to the file ---

try:

    # Open the file in write mode ('w')

    with open(filename, "w") as file:

        # Write some content to the file

        file.write("This is the first line.\n")

        file.write("This is the second line.\n")

        file.write("This is the third line.")

    print(f"Successfully created and wrote to '{filename}'")

except IOError as e:

    print(f"Error writing to file: {e}")

# --- Read from the file ---

try:

    # Open the file in read mode ('r')

    with open(filename, "r") as file:

        # Read the entire content of the file

        content = file.read()

    print(f"\nSuccessfully read from '{filename}'")

    print("\nFile Content:")

    print(content)

except FileNotFoundError:

    print(f"Error: The file '{filename}' was not found.")

except IOError as e:

    print(f"Error reading from file: {e}")

**OUTPUT:**

Successfully created and wrote to 'my\_sample\_file.txt'

Successfully read from 'my\_sample\_file.txt'

File Content:

This is the first line.

This is the second line.

This is the third line.

Task#4

generate a python program that implements a simple calculator using functions(add,substract,multiply and dvide) then explain how code work

**CODE:**

import tkinter as tk

from tkinter import messagebox

import time

# Function to update date and time

def update\_time():

current\_time = time.strftime("%d-%m-%Y %H:%M:%S")

time\_label.config(text=current\_time)

root.after(1000, update\_time) # Update every 1 second

# Function to update the input field

def press(num):

entry\_text.set(entry\_text.get() + str(num))

# Function to clear the input field

def clear():

entry\_text.set("")

# Function to delete last character

def delete():

entry\_text.set(entry\_text.get()[:-1])

# Function to evaluate the expression

def equal():

try:

result = str(eval(entry\_text.get()))

history\_list.insert(tk.END, entry\_text.get() + " = " + result)

entry\_text.set(result)

except ZeroDivisionError:

messagebox.showerror("Error", "Cannot divide by zero")

except:

messagebox.showerror("Error", "Invalid Input")

# Function to clear history

def clear\_history():

history\_list.delete(0, tk.END)

# Create main window

root = tk.Tk()

root.title("Sravs Calculator")

root.geometry("400x580")

root.resizable(False, False)

# Date and Time Label

time\_label = tk.Label(root, font=('Arial', 14, 'bold'), fg="blue")

time\_label.pack(pady=5)

update\_time()

# Entry field for numbers

entry\_text = tk.StringVar()

entry = tk.Entry(root, textvariable=entry\_text, font=('Arial', 20), bd=5, relief="sunken", justify='right')

entry.pack(fill="x", padx=10, pady=10)

# Frame for buttons

button\_frame = tk.Frame(root)

button\_frame.pack()

# Button layout

buttons = [

('7', 0, 0), ('8', 0, 1), ('9', 0, 2), ('/', 0, 3),

('4', 1, 0), ('5', 1, 1), ('6', 1, 2), ('\*', 1, 3),

('1', 2, 0), ('2', 2, 1), ('3', 2, 2), ('-', 2, 3),

('0', 3, 0), ('.', 3, 1), ('+', 3, 2), ('=', 3, 3)

]

for (text, row, col) in buttons:

if text == "=":

tk.Button(button\_frame, text=text, width=5, height=2, font=('Arial', 14), bg="orange",

command=equal).grid(row=row, column=col, padx=5, pady=5)

else:

tk.Button(button\_frame, text=text, width=5, height=2, font=('Arial', 14),

command=lambda t=text: press(t)).grid(row=row, column=col, padx=5, pady=5)

# Extra buttons for clear and delete

tk.Button(button\_frame, text="C", width=5, height=2, font=('Arial', 14), bg="red", fg="white",

command=clear).grid(row=4, column=0, padx=5, pady=5)

tk.Button(button\_frame, text="⌫", width=5, height=2, font=('Arial', 14), bg="lightblue",

command=delete).grid(row=4, column=1, padx=5, pady=5)

# History box

tk.Label(root, text="History", font=('Arial', 14, 'bold')).pack(pady=5)

history\_list = tk.Listbox(root, height=7, font=('Arial', 12))

history\_list.pack(fill="both", padx=10, pady=5)

# Clear History button

tk.Button(root, text="Clear History", font=('Arial', 12), bg="gray", fg="white",

command=clear\_history).pack(pady=5)

root.mainloop()

OUTPUT:

