Jagdish - Advanced Calculator

The software performs two essential operations that contain complex calculus features merged with fundamental arithmetic capabilities. Users can perform trigonometric calculations for sin, cos, tan while benefiting from built-in memory storage and differentiation and integration features in this software. The app features a conversion feature that allows users to switch between the binary, hexadecimal and octal and decimal number formats.  
**User Interface Design**

**Description:**  
The Jagdish Advanced Calculator offers a friendly design which lets users reach all vital functions using the display screen.

**Input Display:** Displays the current expression at the top of the screen for easy tracking.

Operators (+ - \* /) and numbers from zero to nine appear in a spaced grid forming an easy entry system.

The Advanced Functions section of the Jagdish Advanced Calculator includes three core functions namely d/dx for differentiation and ∫dx for integration and sin, cos, tan for trigonometry.

The memory storage systems of the memory functions consist of MC and MR alongside MS and M+ and M-. The system allows users to store calculation results through memory functions and retrieve them when necessary.

Conversion Options: Supports binary, decimal, octal, and hexadecimal conversions.

Each part of the user interface has a logical structure allowing customers to find all functions easily to maximize their  
operating time.   
**Basic Operations**

**Overview:**

The Jagdish Advanced Calculator can handle simple math tasks like adding, subtracting, multiplying, and dividing. It also includes trigonometric functions like sine, cosine, and tangent, plus some calculus functions such as differentiation and integration. Users just need to type in their expression (for example), and the results show up when they hit the = button. There are special buttons for input (CCE) and a dashed entry to make calculations straightforward and user-friendly.  
**Memory Features**

The Jagdish Advanced Calculator makes it simple for users to manage their calculation results with some helpful memory options.

MC (Memory Clear): This feature clears any saved value from memory.

MR (Memory Recall): With this, you can easily retrieve a saved value for more calculations.

MS (Memory Store): Use this to save the current result in memory for later use.

M+ / M- (Memory Add / Subtract): These functions let you add to or subtract from what’s already stored in memory.  
 **Advanced Calculus Functions**

**Differentiate (d/dx):** Evaluates the derivative of input expression (e.g. d/dx x^2 2\*x).  
**Integration (dx):** Evaluates integration integral of entered expression (dx x2 x3/3).

**Process:**

Mathematical expression entered by the user

Select the calculus operation key.

It gives you the result along with the solution.  
**Trigonometric functions**

It has three trigonometric functions: sin cos and tan for which input angle is automatically converted before calculation to radians.

For example for sin(30), this calculator first converts 30 degrees to radians calculate the result and it is 0.5.

The result of a trigonometric function is presented based on the input value. In this way it gives incorrect outputs for any angle entered.  
**Number System Conversion**

Conversion Options:

bin→dec: Converts binary to decimal.

dec→bin: Converts decimal to binary.

dec→oct: Converts decimal to octal.

dec→hex: Converts decimal to hexadecimal.  
**Error Handling**

The user when entering an invalid input or calculation error message will pop up in the message box after that.

For example entering a nonmathematical expression will trigger an error message to inform the user of the mistake.  
**Interface and Interactions:**

Buttons: Users interact with the calculator using a grid of buttons for numbers, operations, and special functions.

Display: The current calculation and result are shown on the input screen.  
Special buttons (C, CE) allow users to reset the screen or clear  
inputs.  
How it work:  
When you press a button on the calculator it updates the expression that you entered into the equation. If the input is valid, the calculator performs the operation and shows the result. Advanced functions like differentiation and integration are processed symbolically while trigonometric computations and conversions happen instantly.

Memory functions store results for later use and make it easier to handle multi-step calculations.

This combination of basic advanced and memory features makes the Advanced Calculator a powerful tool for both simplified and complex calculations.  
**code for advance calculator**import tkinter as tk

from tkinter import messagebox

import sympy as sp  # Import sympy for calculus operations

import math

class Jagdish:

    def \_\_init\_\_(self, root):

        self.root = root

        self.root.title("Jagdish - Advanced Calculator")

        self.root.geometry("500x600")

        self.root.resizable(False, False)

        self.expr = ""

        self.memory = 0

        self.input\_text = tk.StringVar()

        self.create\_widgets()

    def create\_widgets(self):

        entry = tk.Entry(self.root, textvariable=self.input\_text, font=("Arial", 18), bd=5,

                         relief=tk.RIDGE, justify='right')

        entry.grid(row=0, column=0, columnspan=6, ipadx=8, ipady=8)

        buttons = [

            ('MC', 1, 0), ('MR', 1, 1), ('MS', 1, 2), ('M+', 1, 3), ('M-', 1, 4),

            ('7', 2, 0), ('8', 2, 1), ('9', 2, 2), ('/', 2, 3), ('C', 2, 4),

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

            ('1', 4, 0), ('2', 4, 1), ('3', 4, 2), ('-', 4, 3), ('⌫', 4, 4),

            ('0', 5, 0), ('.', 5, 1), ('+', 5, 2), ('=', 5, 3),

            ('d/dx', 6, 0), ('∫dx', 6, 1), ('sin', 6, 2), ('cos', 6, 3), ('tan', 6, 4),

            ('bin→dec', 7, 0), ('dec→bin', 7, 1), ('dec→oct', 7, 2), ('dec→hex', 7, 3)

        ]

        for (text, row, col) in buttons:

            tk.Button(self.root, text=text, font=("Arial", 14), width=6, height=2,

                      command=lambda t=text: self.on\_button\_click(t)).grid(row=row, column=col, padx=2, pady=2)

    def on\_button\_click(self, button):

        try:

            if button.isdigit() or button == '.' or button == 'x':

                self.expr += button

            elif button in '+-\*/':

                self.expr += f' {button} '

            elif button == '=':

                self.expr = str(eval(self.expr))

            elif button == 'C' or button == 'CE':

                self.expr = ""

            elif button == '⌫':

                self.expr = self.expr[:-1]

            elif button == 'MS':

                self.memory = float(eval(self.expr)) if self.expr else 0

            elif button == 'MR':

                self.expr += str(self.memory)

            elif button == 'MC':

                self.memory = 0

            elif button == 'M+':

                self.memory += float(eval(self.expr)) if self.expr else 0

            elif button == 'M-':

                self.memory -= float(eval(self.expr)) if self.expr else 0

            elif button == 'd/dx':  # Differentiation

                x = sp.symbols('x')

                expr\_sympy = sp.sympify(self.expr)

                self.expr = str(sp.diff(expr\_sympy, x))

            elif button == '∫dx':  # Integration

                x = sp.symbols('x')

                expr\_sympy = sp.sympify(self.expr)

                self.expr = str(sp.integrate(expr\_sympy, x))

            elif button in ['sin', 'cos', 'tan']:  # Trigonometric functions

                self.expr = str(getattr(math, button)(math.radians(float(eval(self.expr)))))

            elif button == 'bin→dec':

                self.expr = str(int(self.expr, 2))

            elif button == 'dec→bin':

                self.expr = bin(int(self.expr))[2:]

            elif button == 'dec→oct':

                self.expr = oct(int(self.expr))[2:]

            elif button == 'dec→hex':

                self.expr = hex(int(self.expr))[2:].upper()

        except:

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

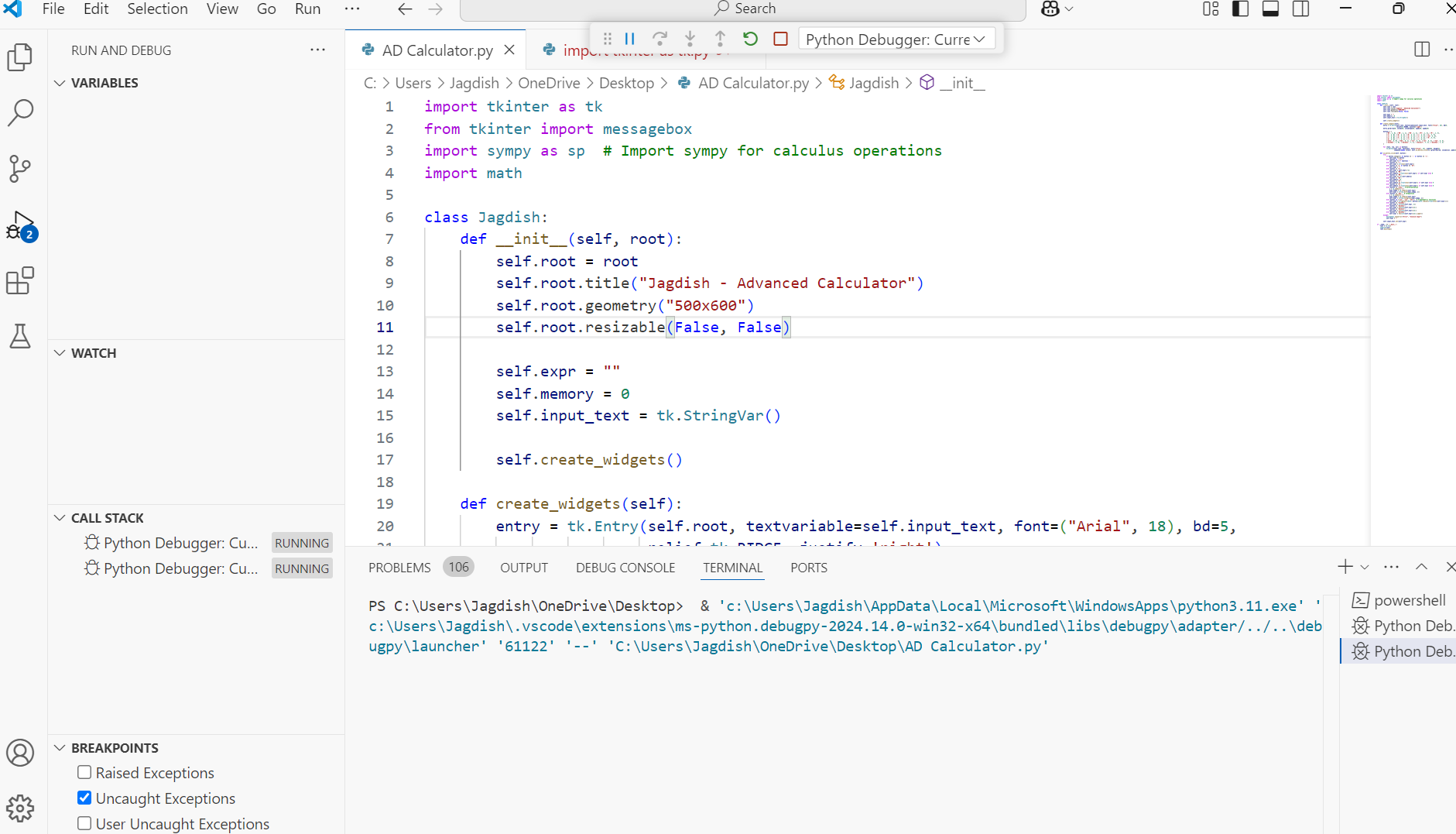
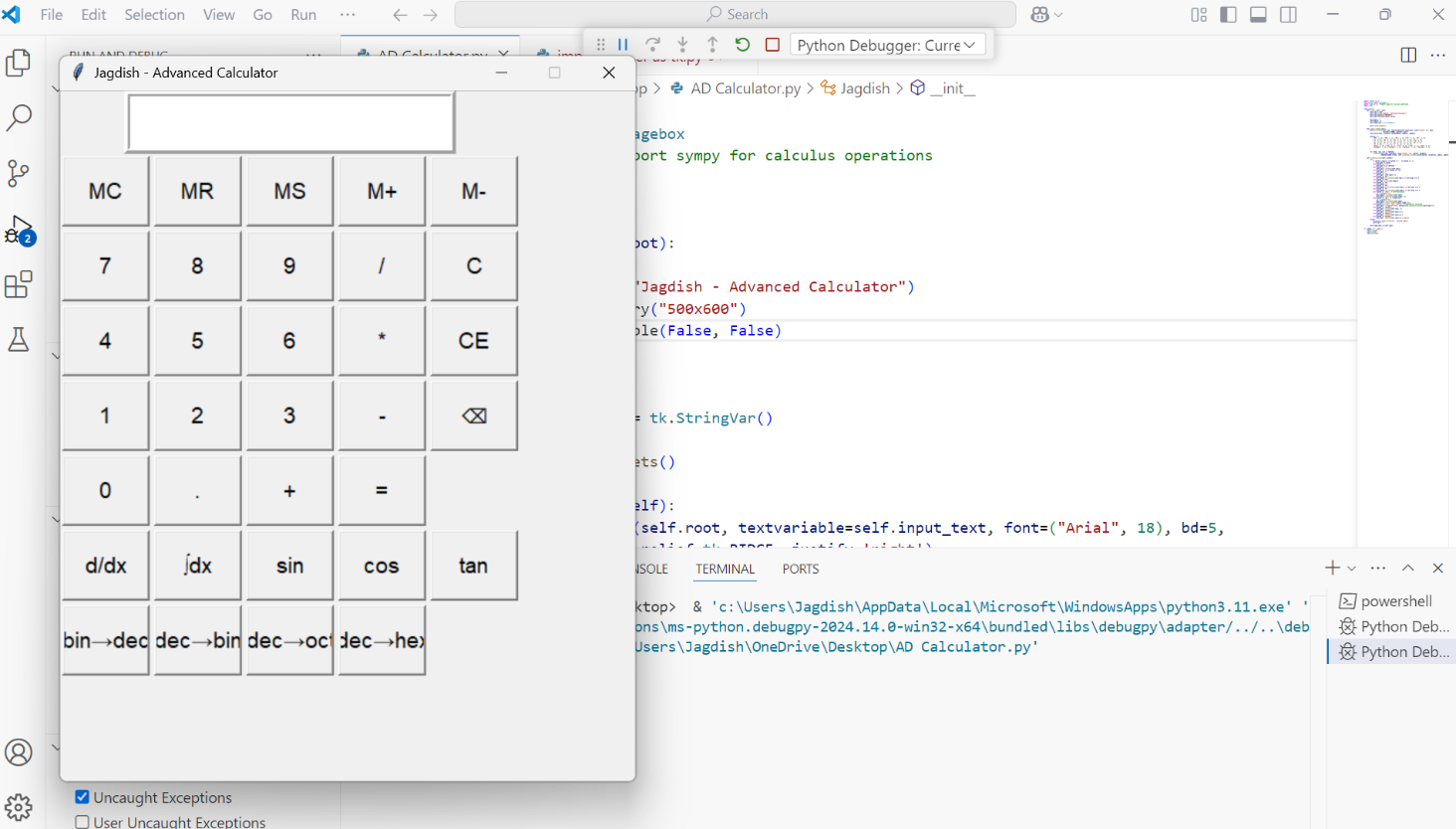
            self.expr = ""

        self.input\_text.set(self.expr)

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

    root = tk.Tk()

    Jagdish(root)

    root.mainloop()  
**screen shot of working** ****

