ADVANCED CALCULATOR

Problem Statement:

Making advanced calculator which includes trigonometrical and square/square root capabilities along with simple functions , max 2 entities.

Requirements:

1. Python 3.10.9 is a high-level, interpreted, general-purpose programming language. It contains various built-in methods for file handling and supports various file management libraries, making it highly attractive for our use-case.

2. Visual Studio Code 1.75.1 is a light-weight code editor with an extensive extension library.

Installations (Mac):

Python 3.10.9

Step 1:
Download the installer package from Python's official website.
Download Python.

Step 2:
Wait for the download to complete. Once it's finished, double-click the package to start the installation process.

Step 3:
Once the installation is complete, the installer will automatically open Python's installation directory in a new Finder window.

Visual Studio Code 1.75.1:

Step 1:
Open the official Visual Studio Code website and download the latest version for Mac. Download Visual Studio Code - Mac, Linux, Windows

Step 2:
Click on Apple Silicon under the Mac icon to download VS Code's package installer for Mac in a ZIP file.

Step 3:
Open finder in your mac and drag Visual Studio Code.app to the Applications folder, making it available in the macOS Launchpad.

Python Calculator:

The calculator is one application that we all use in our day to day lives. If you are trying to get your hands dirty with programming in python, Calculator is a project which is easy and useful at the same time. Today, we are going to build a Python Calculator using Tkinter with easy to understand steps.

Tkinter:

Python offers various utilities to design the GUI wiz Graphical User Interface, and one such utility is Tkinter which is most commonly used. It is indeed one of the fastest and easiest ways to build GUI application. Moreover, Tkinter is cross-platform, hence the same code works on macOS, Windows, and Linux.

Steps involved-

Step 1: Importing the necessary modules

To use the Tkinter we need to import the Tkinter module. We are also going to import the function factorial from math module.

```
from tkinter import *
import parser
From math import factorial
```

Step 2: Making a window for our calculator

Now we are going to draft the window for our calculator which will accommodate the buttons.

```
root = Tk()
root.title('Advanced - Calculator')
root.mainloop()
```

The above code sets the title of python calculator window as ‘Advanced – Calculator’. When you run the above code, you will get a window which will be blank.

Step 3: Designing the buttons

Now let’s quickly design the buttons for our calculator and put them on our application window.

```
#Code to add buttons to the Calculator
Button(root,text="1",command = lambda :get_variables(1)).grid(row=2,column=0,
sticky=N+S+E+W)
Button(root,text=" 2",command = lambda :get_variables(2)).grid(row=2,column=1,
sticky=N+S+E+W)
Button(root,text=” 3”,command = lambda :get_variables(3)).grid(row=2,column=2,
sticky=N+S+E+W)
```

In this calculator program in python, the “Entry” function helps in making a text input field and we use .grid() method to define the positioning associated with the button or input field. We use the button method to display a button on our application window.

- root – the name with which we refer to our window
- text – text to be displayed on the button
- row – row index of the grid
- column – column index of the grid
- columnspan – spans or combines the number of columns
- sticky – If the resulting cell is larger than the widget then sticky defines how to expand the widget. The combination of constants used S, N, E, and W, or NW, NE, SW, and SE are analogous to the directions in compass. N+E+W+S means that the widget should be expanded in all directions

Step 4: Mapping the buttons to their functionalities

Mapping the digits

```
def get_variables(num):
   global i
   display.insert(i,num)
   i +=1
```

The get_variable() function receives the digit as parameter. The digit is inserted to the input field with .insert() method with parameters ‘i’ and ‘num’. The global variable i is incremented each time to get updated with the position to insert the next digit or next operator.

- i – the position to insert the digit
- num – the digit

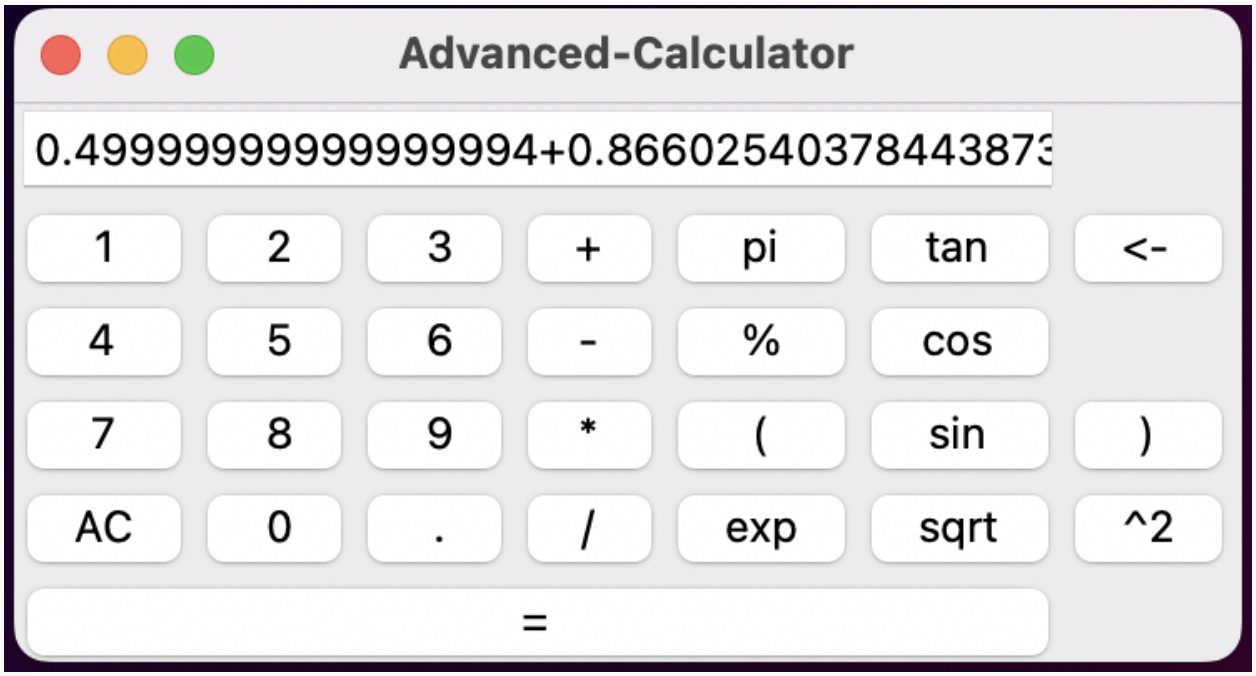
FUNCTION DESCRIPTION:

calculate()
Function to do general calculations.
Parse module is used to scan the displayed expression. expr() method accepts the string as parameter and an abstract syntax tree is created for the evaluation.

Advanced functions()

Note: We are first taking values and after that we take the required function to be performed.
To perform the complex operations like sin30+cos30, we first created a list of all the basic operators. Our main aim was to convert the string entered by the user into a normal expression like example; sin30+cos30 is converted to 0.5+0.8660. Here first, sin 30 is evaluated and then the '+' operator is added and finally the value of cos30 is evaluated. This normal string is further calculated using the normal calculate() method.

Normal expression of sin30 + cos30



Final result