PROJECT REPORT

On

**Project Name:**

**SCIENTIFIC CALCULATOR**

Course Code: INT 213

Faculty: Navpreet Rupal

**BACHELOR OF TECHNOLOGY**

**in**

**COMPUTER SCIENCE AND ENGINEERING**

Team Members:

|  |  |  |  |
| --- | --- | --- | --- |
| **S no** | **Name** | **Registration number** | **Roll No.** |
| 1 | ***Komati Durganadh*** | 12111551 | A15 |
| **2** | Isha Aggarwal | 12115309 | B39 |
| 3 | Abhishek Singh | 12116215 | B64 |



**INTRODUCTION:**

Project allocated to our team is Scientific Calculator. It is a Single User.

This Mini project of “Scientific Calculator” is purely made in python, with a good user-friendly interface which lets the user to enter the various values required to be input in order to calculate simple and complex problems in science, engineering & mathematics.

It’s a GUI-based project used with the tkinter module to organize all the elements that work under Scientific Calculator in Python.

The main window consists of all the entry fields that the user is required to fill in order to generate a output. This main window also has a button to give output when operation is performed which has the required values.

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

**OBJECTIVE:**

For those who do not know, a **calculator** is basically a program on a **computer that simulates** the behavior of any **hand-held calculator** useful for **performing Mathematical Calculations**. It is a very basic device used in our everyday lives. Now all the smartphones also have a Calculator application in them.

While creating any GUI Application there are mainly two steps:

* The first step is to **create a User Interface**.
* The second step is the most important one and in this, to **add functionalities to the GUI**

Now let's begin with **creating a Scientific Calculator** using Tkinter in Python which is used for **calculations**.

**LIBRARIES USED:**

This project uses tkinter library which is used to make the G.U.I. Some modules of tkinter used are buttons, entry box, labels and progress bars. All here are used to represent the report graphically to the user.

**HARDWARE:**

As a part of hardware, the user should have a laptop or desktop computer which is capable of handling python idle. It should also have minimum of 2GB ram for the easy compilation and smooth running of the program.

**LOGIC USED:**

Program used a bunch of if statements used to decide the output according to the input given by user.

This is the basic logic on which the program works.

The calculation of problems or statements and other values is done by using formulas and data collected is used to calculate the output of a particular value in terms of numbers.

There are various functions and if statement which also control output when a certain value is entered.

**BASIC FUNCTIONS:**

# Addition

The addition (sum function) is used by clicking on the "+" button or using the keyboard. The function results in a+b.

# Subtraction

The subtraction (minus function) is used by clicking on the "-" button or using the keyboard. The function results in a-b.

# Multiplication

The multiplication (times function) is used by clicking on the "x" button or using the keyboard"\*" key. The function results in a\*b.

# Division

The division (divide function) is used by clicking on the "/" button or using the keyboard "/"key. The function results in a/b.

# Sign

The sign key (negative key) is used by clicking on the "(-)" button. The function results in -1\*x.

# Square

The square function is used by clicking on the "x^2" button or type "^2". The function results inx\*x.

# Square Root

The square root function is used by clicking on the "x" button or type "sqrt()". This function represents x^.5 where the result squared is equal to x.

# Raise to the Power

The raise to the power (y raised to the x function) is used by clicking on the "y^x" button ortype "^".

# Natural Exponential

The natural exponential (e raised to the x) is used by clicking on the "e^x" button or type "exp()". The result is e (2.71828...) raised to x.

**Logarithm**

The logarithm (LOG) is used by clicking on the "LOG" button or type "LOG()".

**Natural Logarithm**

The Natural logarithm (LN) is used by clicking on the "LN" button or type "LN()".

# Inverse

Multiplicative inverse (reciprocal function) is used by pressing the "1/x" button or typing "inv()". This function is the same as x^-1 or dividing 1 by the number.

# Exponent

Numbers with exponents of 10 are displayed with an "e", for example 4.5e+100 or 4.5e-100.This function represents 10^x. Numbers are automatically displayed in the format when the number is too large or too small for the display. To enter a number in this format, use the exponent key "EEX". To do this enter the mantissa (the nonexponent part) then press "EEX" or type "e" and then enter the exponent.

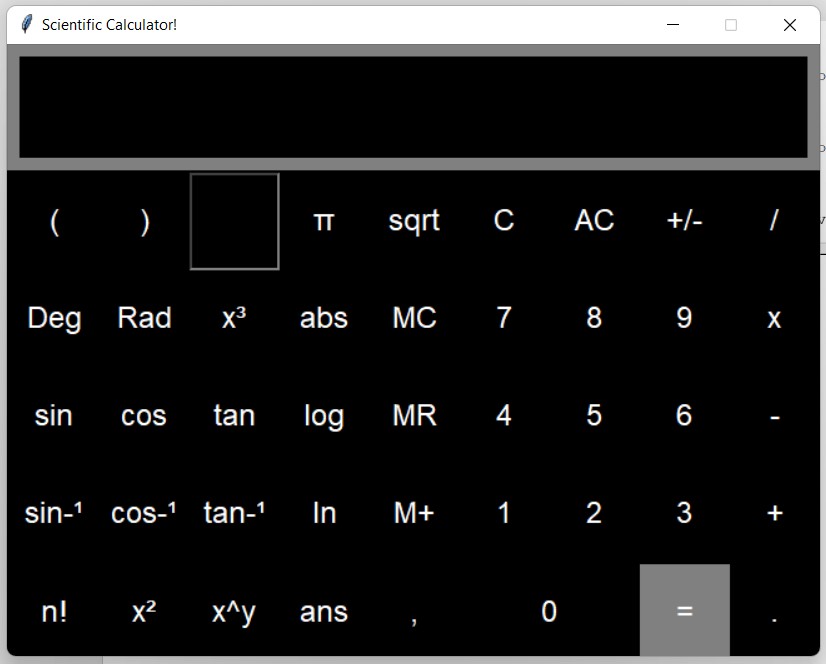
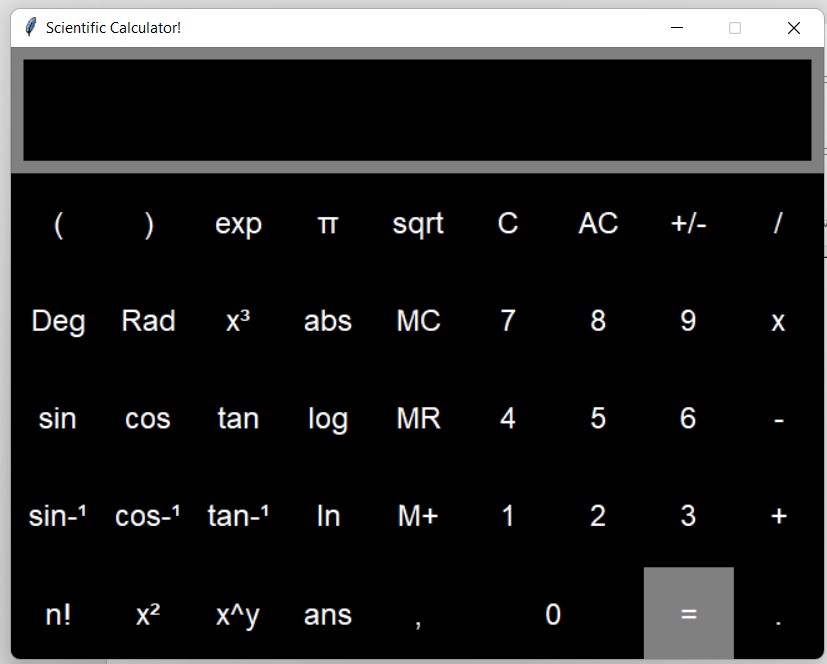
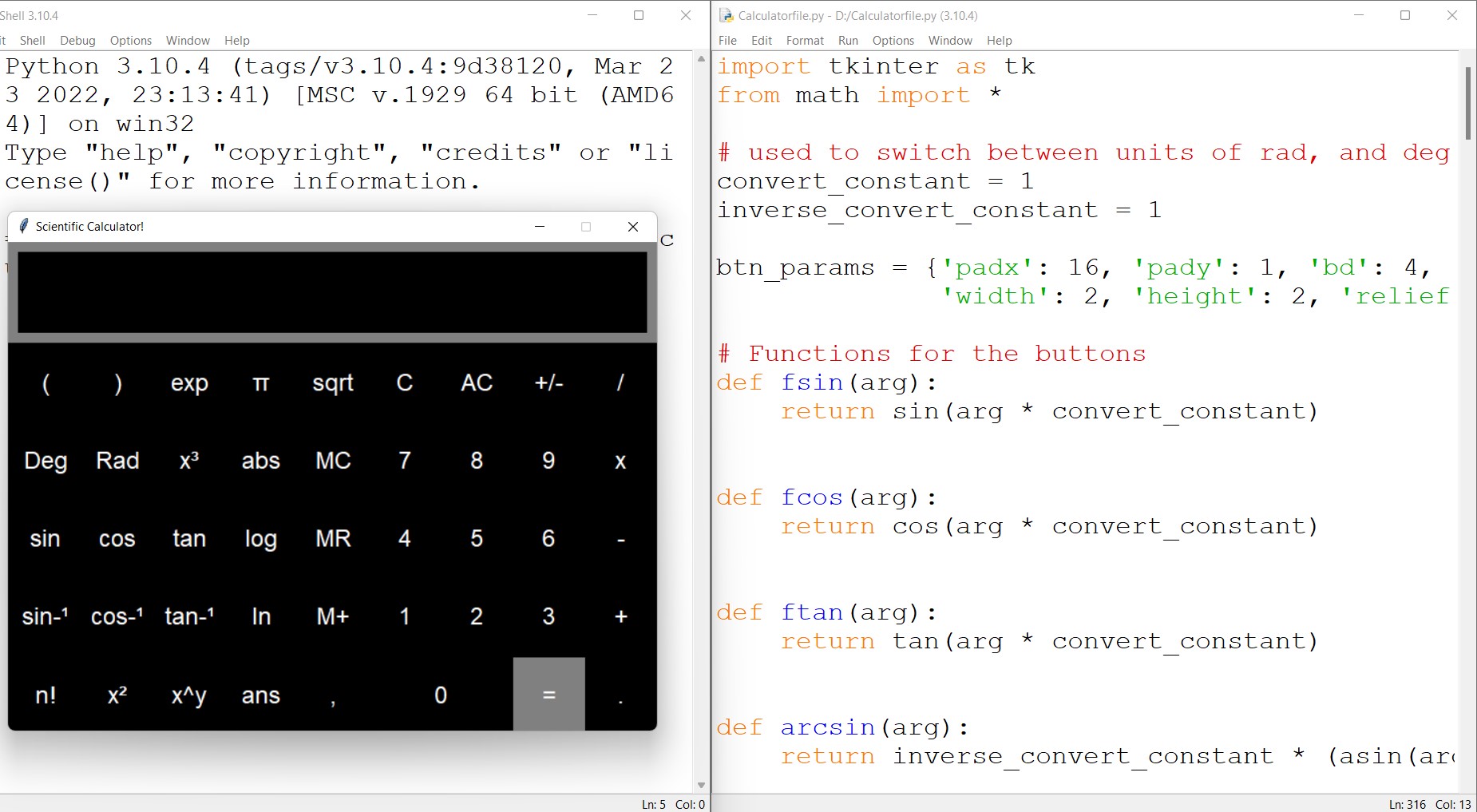
**Factorial**

The Factorial function is used by clicking the "!" button or type "!".

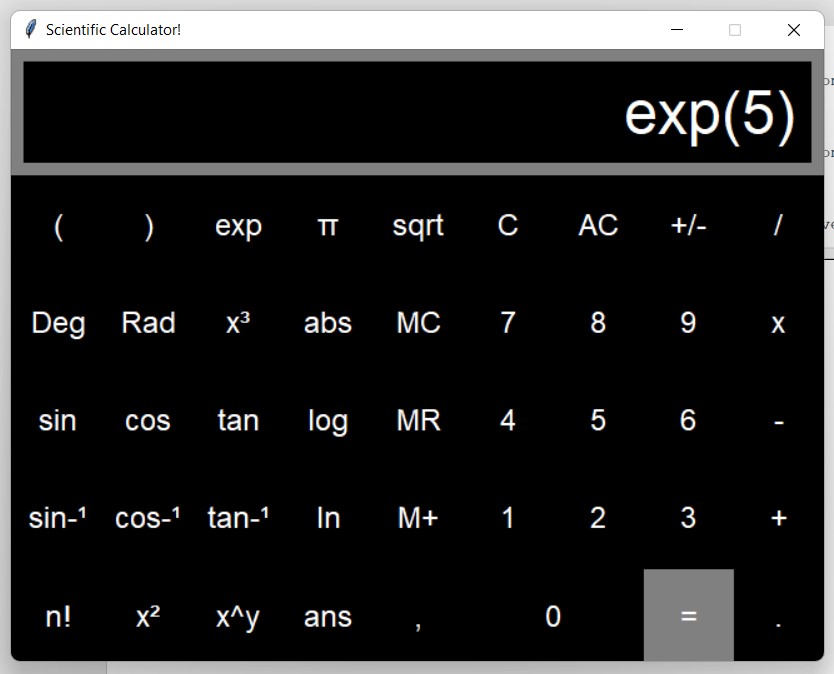
# PI

PI is a mathematical constant of the ratio of a circle's circumference to its diameter.

**SCREENSHOTS:**



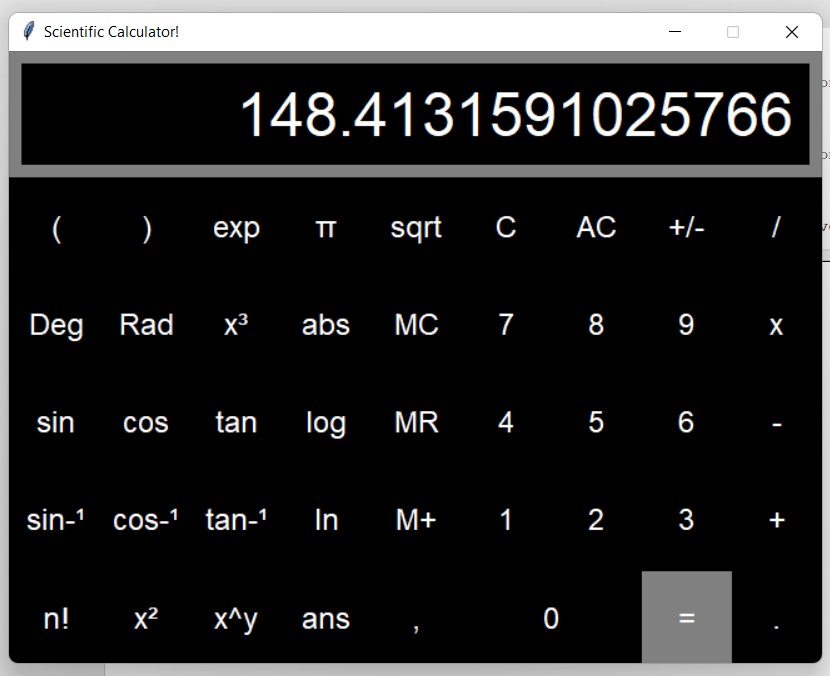
**Pressing exp Button to find exponential of a**



**number**

**.**

**Finding Exponential of 5 using exp function.**



**To get output we have to click “=” button.**

**FUTURE SCOPE:**

Our project will be able to implement in future after making some changes and modifications as we make our project at very low level. So the modifications that can be done in our project are:

To make it screen touch so no need to touch key buttons and one more change which can we made is to add snaps of the persons who use it.

**CONCLUSION:**

We built a Scientific Calculator in Python using Tkinter and defined what happens when a user interacts with a GUI element.

Using Frame widgets to create containers/windows in which to arrange UI components. Positioning Button widgets within a Frame widget using Tkinter’s Layout Manager. The grid layout manager was chosen for the Calculator application due to its precision compared to alternative layout techniques.

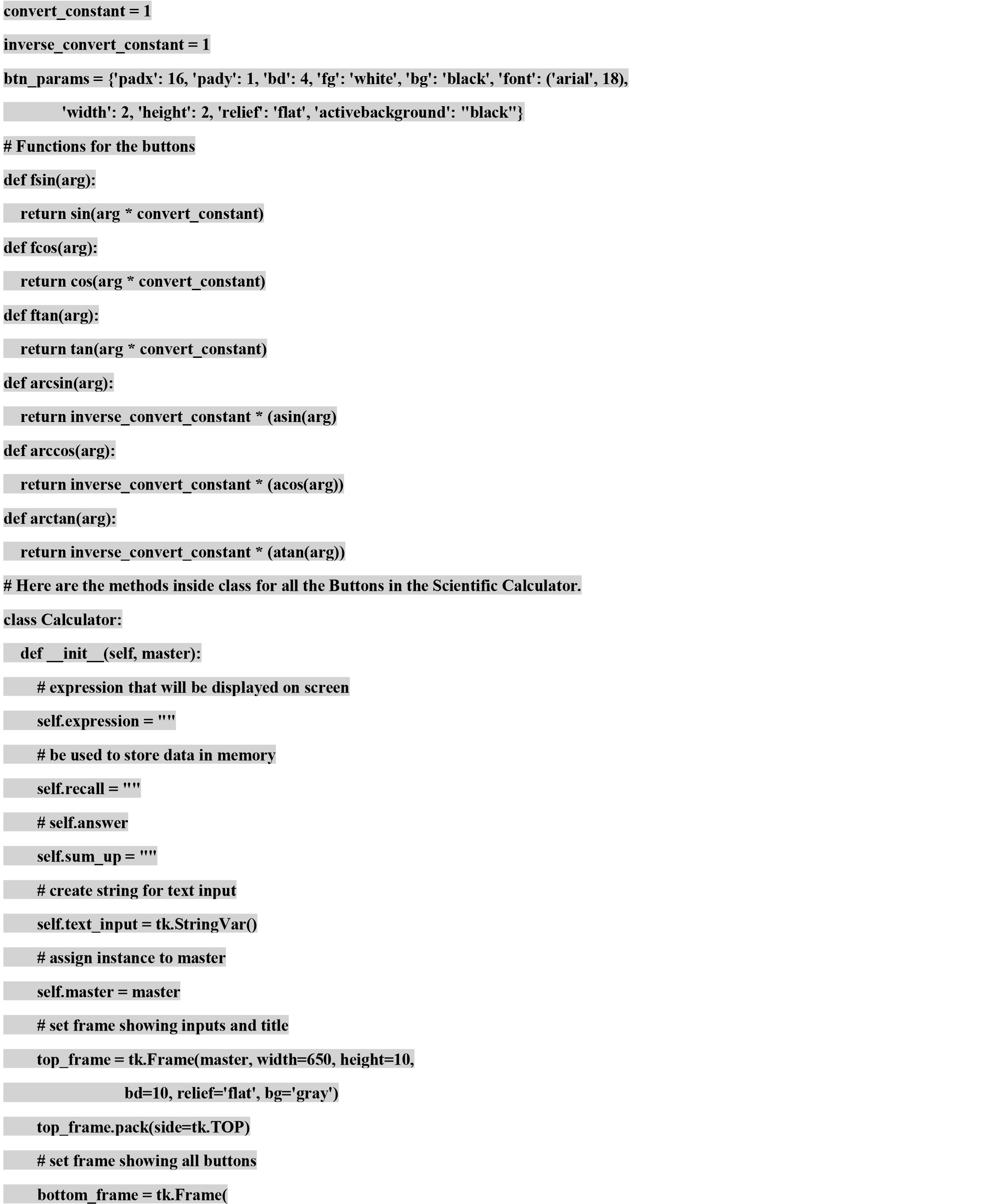
**REFERENCE:**

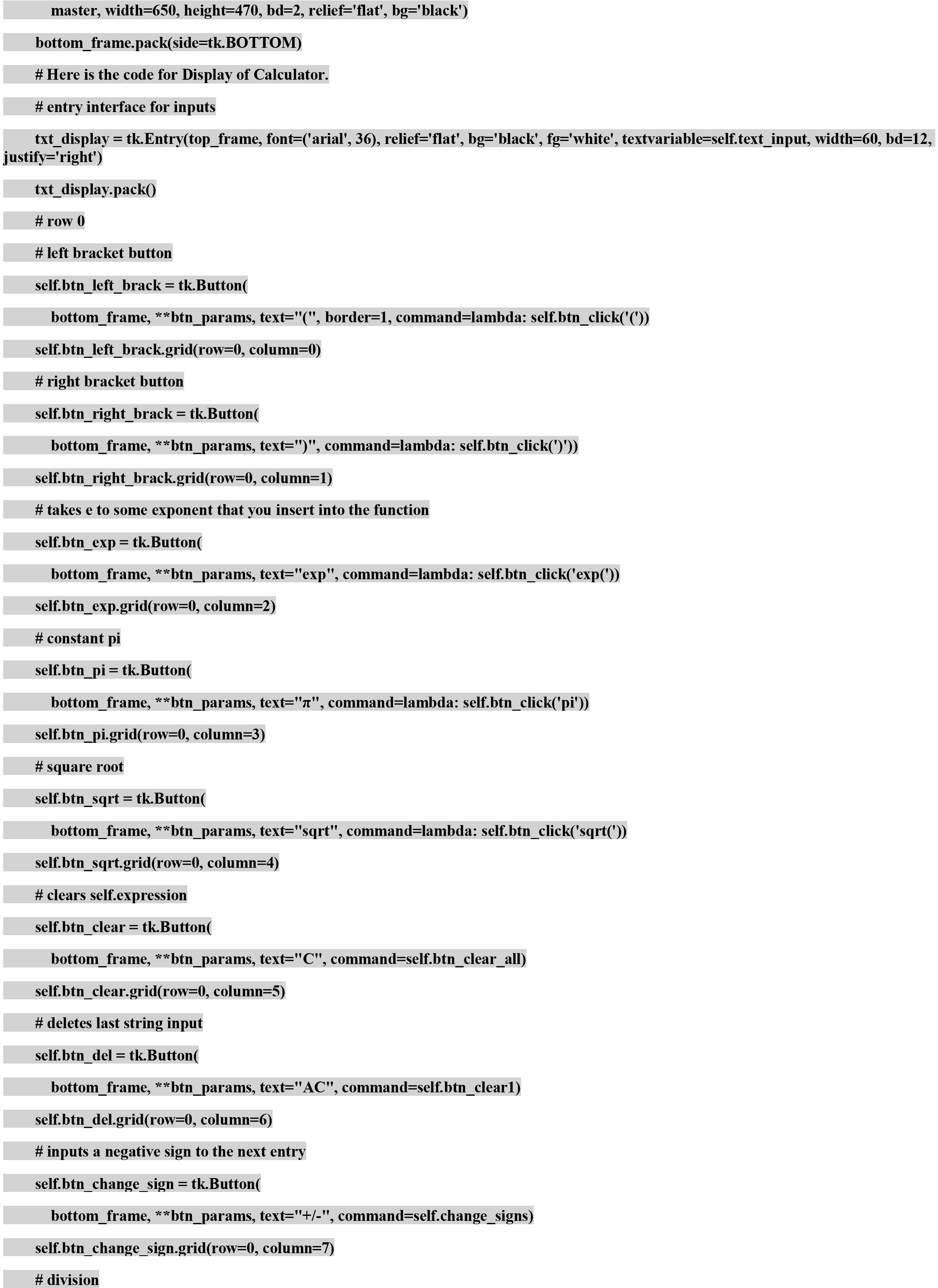
* <https://www.geeksforgeeks.org/python-gui-tkinter/>
* [https://www.tutorialspoint.com/python/python\_gui\_programming.htm #:~:text=Tkinter%20Programming&text=Import%20the%20Tkinter% 20module.,event%20triggered%20by%20the%20user.](https://www.tutorialspoint.com/python/python_gui_programming.htm#:~:text=Tkinter%20Programming&text=Import%20the%20Tkinter%20module.,event%20triggered%20by%20the%20user)
* <https://www.javatpoint.com/python-tkinter>
* <https://realpython.com/python-gui-tkinter/>
* [https://www.academia.edu/41687743/PROJECT\_REPORT\_Scientific \_Calculator\_SUBMITTED\_BY\_GUIDED\_BY\_NAME\_D\_Murali\_kri shna\_ATMAJA\_MADAM\_BATCH\_ITAP\_203\_CRANES\_VARSIT Y\_CRANES\_VARSITY\_BENGALURU\_BENGALURU](https://www.academia.edu/41687743/PROJECT_REPORT_Scientific_Calculator_SUBMITTED_BY_GUIDED_BY_NAME_D_Murali_krishna_ATMAJA_MADAM_BATCH_ITAP_203_CRANES_VARSITY_CRANES_VARSITY_BENGALURU_BENGALURU)

**ANNEXURE-A:**

**import tkinter as tk from math import \***

**# used to switch between units of rad, and deg**





**self.btn\_div = tk.Button(**

**bottom\_frame, \*\*btn\_params, text="/", command=lambda: self.btn\_click('/'))**

**self.btn\_div.grid(row=0, column=8)**

**# row**

**1**

**# changes trig function outputs to degre**

**es**

**self.btn\_Deg = tk.Button(bottom\_frame, \*\*btn\_params, activeforeground='gray', text="Deg", command=self.convert\_deg)**

**self.btn\_Deg.grid(row=1, column=0)**

**# changes trig function outputs to default back to radians**

**self.btn\_Rad = tk.Butt**

**on(bottom\_frame, \*\*btn\_params, foreground='white', activeforeground='Gray', text="Rad",**

**command=self.convert\_rad**

**)**

**)**

**self.btn\_Rad.grid(row=1, column=1**

**e**

**# cubes a valu**

**self.cube = tk.Button(bottom\_frame, \***

**\*btn\_params, text=u"x**

**\**

**u00B3", command=lambda: self.btn\_click('\*\*3'))**

**self.cube.grid(row=1, column=2)**

**# takes the absolute value of an expression**

**(**

**self.btn\_abs = tk.Button**

**params, text="abs", command=l**

**bottom\_frame, \*\*btn\_**

**ambda: self.btn\_click('abs' + '('))**

**column=3)**

**self.btn\_abs.grid(row=1,**

**# 'memory clear' button.**

**Wipes self**

**.recall to an empty string**

**(**

**self.btn\_MC = tk.Button**

**bottom\_frame, \*\*btn\_params, te**

**xt="MC", command=sel**

**f.memory\_clear)**

**self.btn\_MC.grid(row=1, column=4)**

**# seven**

**self.btn\_7 = tk.Button(bottom\_frame, \*\*btn\_params, text="**

**7**

**", command=lambda: self.btn\_click**

**(7))**

**self.btn**

**\_7.configure(activebackground="black", bg='black')**

**self.btn\_7.grid(row=1, column=5)**

|  |  |
| --- | --- |
| **# eight** |  |
| **self.btn\_8 = tk.Button(bottom\_frame, \*\*btn\_params, text="8", command=lambda: self.btn\_click(8))** | |

**self.btn\_8.configure(activebackgroun**

**)**

**d="black", bg='black'**

**)**

**self.btn\_8.grid(row=1, column=6**

**e**

**# nin**

**(9))**

**9**

**ame, \*\*btn\_params, text="**

**self.btn\_9 = tk.Button(bottom\_fr**

**", command=lambda: self.btn\_click**

**ound="black", bg='black')**

**self.btn\_9.configure(activebackgr**

**self.btn\_9.grid(row=1, column=7**

**)**

**# multiplication**

**self.btn\_mult = tk.Button(bottom\_frame, \*\*btn\_params, text="x", command=lambda: self.btn\_click('\*'))**

**self.btn\_mult.grid(row=1, column=8)**

**# row**

**2**

**# sin functio**

**n that returns value from**

**-**

**1**

**to 1 by defaul**

**t**

**self.btn\_sin = tk.Button**

**(**

**bottom\_frame, \*\*btn\_params, text="sin", command=lambda: self.btn\_click('fsin('))**

**self.btn\_sin.grid(row=2, column=0)**

**# cos function that returns value**

**from**

**-**

**1**

**to 1 by defaul**

**t**

**self.btn\_cos = tk.Button**

**(**

**bottom\_frame, \*\*btn\_params, text="cos", command=lambda: self.btn\_click('fcos('))**

**self.btn\_cos.grid(row=2, column=1)**

|  |  |
| --- | --- |
| **# tan function** |  |
| **self.btn\_tan = tk.Button(bottom\_frame, \*\*btn\_params, text="tan", command=lambda: self.btn\_click('ftan('))** | |

**self.btn\_tan.grid(row=2, column=2)**

**#**

**ame, \*\*btn\_params, text="log", command=lambda: self.btn\_click('log('))**

**self.btn\_log = tk.Button(bottom\_fr**

**self.b**

**)**

**tn\_log.grid(row=2, column=3**

**# outputs what is in self.recall**

**self.btn\_MR = tk.Button(bottom\_frame, \*\*btn\_params, text="MR", command=self.memory\_recall)**

**self.btn\_MR.grid(row=2, column=4)**

**r**

**# fou**

**tn\_4 = tk.Button**

**self.b**

**(**

**bottom\_frame, \*\*btn\_params, text="4", command=lambda: self.btn\_click**

**(4))**

**tn\_4.configure(activebackground="black", bg='black')**

**self.b**

**self.b**

**)**

**tn\_4.grid(row=2, column=5**

**# fiv**

**e**

**self.btn\_5 = tk.Button(bottom\_frame, \*\*btn\_params, text="5", command=lambda: self.btn\_click(5))**

**self.btn\_5.configure(activebackground="black", bg='black')**

**self.btn\_5.grid(row=2, column=6**

**)**

**# si**

**x**

**self.btn\_6 = tk.Button**

**ame, \*\*btn\_params, text="6", command=lambda: self.btn\_click**

**bottom\_fr**

**(**

**(6))**

**ound="black", bg='black')**

**self.btn\_6.configure(activebackgr**

**)**

**self.btn\_6.grid(row=2, column=7**

**# subtractio**

**n**

**self.btnSub = tk.Button(bottom\_frame, \*\*btn\_**

**params, text="**

**-**

**", command=lambda: self.btn\_click('**

**-**

**'))**

**self.btnSub.grid(row=2, column=8**

**)**

**# row**

**3**

**# sin inverse function**

**self.btn\_sin\_inverse**

**=**

**tk.Button(bottom\_frame, \*\*btn\_params, text=u"sin**

**-**

**\**

**u00B9", command=lambda: self.bt**

**)**

**n\_click('arcsin(')**

**self.btn\_sin\_inverse.**

**grid(row=3, column=0)**

**# cos inverse function**

**self.btn\_cos\_inverse = tk.Button(bottom\_frame, \*\*btn\_params, text=u"cos**

**-**

**\**

**u00B9", command=lambda: self.btn\_click('arccos('))**

**self.btn\_cos\_i**

**nverse.grid(row=3, column=1)**

**# tan inverse function**

**self.btn\_tan\_inverse**

**=**

**tk.Button(bottom\_frame, \*\*btn\_params, text=u"tan**

**-**

**\**

**u00B9", command=lambda: self.btn\_click('arctan('))**

**self.btn\_tan\_inverse.g**

**rid(row=3, column=2)**

**# tak**

**es the natural lo**

**g**

**self.btn\_ln = tk.Button(bottom\_frame, \*\*btn\_params, text="ln", command=lambda: self.btn\_click('log1p('))**

**self.btn\_ln.grid(row=3, column=3)**

**# adds current self.expression to self.recall string**

**self.btn\_M\_pl**

**us = tk.Button(bottom\_frame, \*\*btn\_params, text="M+", command=self.memory\_add)**

**self.btn\_M\_plus.grid(row=3, column=4)**

**# one**

**self.btn\_1 = tk.Button(bottom\_frame, \*\*btn\_params, text="1", command=lambda: self.btn\_click(1))**

**self.btn\_1.configure(activebackground="black", bg='black')**

**self.btn\_1.grid(row=3, column=5**

**)**

**# tw**

**o**

**self.btn\_2 = tk.Button(bottom\_frame, \*\*btn\_params, text="2", command=lambda: self.btn\_click(2))**

**self.btn\_2.configure(a**

**ctivebackground="black", bg='black')**

**self.btn\_2.grid(row=3, column=6)**

|  |  |
| --- | --- |
| **# three** |  |
| **self.btn\_3 = tk.Button(bottom\_frame, \*\*btn\_params, text="3", command=lambda: self.btn\_click(3))** | |

**self.btn\_3.configure(activebackground="black", bg=**

**)**

**'black'**

**self.btn\_3.grid(row=3, column=7**

**)**

**# additio**

**n**

**self.btn\_add = tk.Button**

**(**

**bottom\_frame, \*\*btn\_params, text="+", command=lambda: self.btn\_click('+'))**

**self.btn\_add.grid(row=3, column=8)**

**# row 4**

**n**

**# factorial functio**

**self.btn\_fact = tk.Button(bottom\_frame, \*\*btn\_params, text="n!", command=lambda: self.btn\_click('factorial('))**

**self.btn\_fact.grid(row=4, column=0)**

**n**

**# square functio**

**self.btn\_sqr = tk.Button(botto**

**ame, \*\*btn\_params, text=u"x**

**m\_fr**

**\**

**u00B2", command=lambda: self.btn\_click('\*\*2'))**

**self.btn\_sqr.grid(row=4, column=1)**

**# to the power of function**

**self.btn\_power = tk.Button(bottom\_frame, \*\*btn\_params, text="x^y", command=lambda: self.btn**

**)**

**\_click('\*\*')**

**self.btn\_power.grid(row=4, column=2)**

**# stores previous expression as an answer value**

**self.btn\_ans = tk.Button(bottom\_frame, \*\*btn\_**

**params, text="ans", command=self.answer)**

**self.btn\_ans.grid(row=4, column=3)**

**# comma to allow for more than one parameter**

**!**

**self.btn\_comma = tk.Button(bottom\_frame, \*\*btn\_params, text=",", command=lambda: self.btn\_click(','))**

**self.btn\_comma.grid(row=4, column=4)**

**# zero**

**self.btn\_0 = tk.Button(b**

**ottom\_frame, \*\*btn\_params, text="0", command=lambda: self.btn\_click(0))**

**self.btn\_0.configure(activebackground="black", bg='black', width=7, bd=5)**

**self.btn\_0.grid(row=4, column=5, columnspan=2)**

**# equals button**

**self.btn\_eq = t**

**k.Button(bottom\_frame, \*\*btn\_params, text="=", command=self.btn\_equal)**

**self.btn\_eq.configure(bg='Gray', activebackground='#009999')**

**self.btn\_eq.grid(row=4, column=7)**

**# decimal to convert to float**

**self.btn\_dec = tk.Button(bot**

**tom\_frame, \*\*btn\_params, text=".", command=lambda: self.btn\_click('.'))**

**self.btn\_dec.grid(row=4, column=8)**

**# functions**

**# allows button you click to be put into self.expression**

**def btn\_click(self, expression\_val):**

**if len(self.exp**

**ression) >= 23**

**:**

**self.expression = self.expression**

**self.text\_input.set(self.expression)**

|  |  |
| --- | --- |
| **else:** |  |
| **self.expression = self.expression + str(expression\_val)** | |

**self.text\_input.set(self.expression)**

**g**

**# clears last item in strin**

**:**

**def btn\_clear1(self)**

**self.expression = self.expression[:**

**-**

**1**

**]**

**self.text\_input.set(self.expression)**

**# adds in a negative sig**

**n**

**def change\_signs(self):**

**self.expression = self.expression + '**

**-**

**'**

**self.text\_input.set(self.expression)**

**l**

**# clears memory\_recal**

**:**

**def memory\_clear(self)**

**"**

**self.recall = "**

**# adds whatever is on the screen to self.recall**

**def memory\_add(self):**

**self.recall = self.recall + '+' + self.expression**

**# uses whatever is stored in memory\_recall**

**:**

**def answer(self)**

**self.answer = self.sum\_u**

**p**

**self.expression = self.expression + self.an**

**swer**

**self.text\_input.set(self.expression)**

**# uses whatever is stored in memory\_recall**

**def memory\_recall(self):**

**if self.expression == ""**

**:**

**self.text\_input.set('0' + self.expression + self.recall)**

**else**

**:**

**self.text\_input.set(self.expression + self.recall)**

**# changes self.convert\_constant to a string that allows degree conversion when button is clicked**

**:**

**def convert\_deg(self)**

**t**

**global convert\_constan**

**global inverse\_convert\_constant**

**0**

**convert\_constant = pi / 18**

**inverse\_convert\_constant = 180 / pi**

**self.btn\_Rad["foreground"] = 'white**

**'**

**self.btn\_Deg["foreground"] = 'RED'**

**def convert\_rad(self)**

**:**

**t**

**global convert\_constan**

**global inverse\_convert\_constant**

**convert\_constant**

**=**

**1**

**inverse\_convert\_constant = 1**

**'**

**self.btn\_Rad["foreground"] = 'RED**

**self.btn\_Deg["foreground"] = 'white'**

**# clears self.expression**

**def btn\_clear\_all(self):**

**self.expression = "**

**"**

**self.text\_input.set(""**

**)**

**# converts self.expression into a mathematical expression and evaluates it**

**def btn\_equal(self)**

**:**

**self.sum\_up = str(eval(self.expression))**

**self.text\_input.set(self.sum\_up**

**)**

**self.expression = self.sum\_u**

**p**

**# tkinter layout**

**root =**

**tk.Tk(**

**)**

**b = Calculator(root**

**)**

**root.title("Scientific Calculator!")**

**root.geometry("650x490+50+50")**

**root.resizable(False, False)**

**root.mainloop**