# [Scientific Calculator in Python](https://copyassignment.com/scientific-calculator-in-python-2/)

**Project Overview:**

|  |  |
| --- | --- |
| Project Name: | Scientific Calculator in Python using Tkinter |
| Abstract | It’s a GUI-based project used with the Tkinter module to organize all the elements that work under Scientific Calculator. |
| Language/s Used: | Python |
| IDE | Thonny and PyCharm (Recommended) |
| Python version (Recommended): | Python 3 |
| Database: | None |
| Type: | Desktop Application |

Python GUI Scientific Calculator with Code is a straightforward project created in Python. This project is based on a straightforward GUI and is very simple to comprehend and use. Additionally, the user may easily learn how to execute numerical computations with the help of this project. Like a typical calculator, the project has numbers, operators, and signs. In order to enter any number or simply click on the numbers they desire for the computations, the user needs only click. As a result, the user of this application can utilize a straightforward calculator.

## Features of Scientific Calculator:

* All basic operators like **addition, multiplication, subtraction,**and**division**.
* Dedicated buttons for **sin, cos, tan,**and **cot.**
* Functions for **log**
* Functions for **ratio**and **exponents.**
* Other buttons like **Pi, mod,**and **x!**

## Main Code:

Let's dive right into building a Scientific Calculator in Python using Tkinter. We'll provide a step-by-step explanation along with the source code to help you learn and create the project effectively.

### ****Basic library imports and window configuration:****

from tkinter import \*  
import math  
import tkinter.messagebox  
  
root = Tk()  
root.title("Scientific Calculator")  
root.configure(background='white')  
root.resizable(width=False, height=False)  
  
root.geometry("480x568+450+90")  
calc = Frame(root)  
calc.grid()

### ****Class definition for all the Scientific Calculator functions:****

class Calc():  
 def \_\_init\_\_(self):  
 self.total = 0  
 self.current = ''  
 self.input\_value = True  
 self.check\_sum = False  
 self.op = ''  
 self.result = False  
  
 def numberEnter(self, num):  
 self.result = False  
 firstnum = txtDisplay.get()  
 secondnum = str(num)  
 if self.input\_value:  
 self.current = secondnum  
 self.input\_value = False  
 else:  
 if secondnum == '.':  
 if secondnum in firstnum:  
 return  
 self.current = firstnum + secondnum  
 self.display(self.current)  
  
 def sum\_of\_total(self):  
 self.result = True  
 self.current = float(self.current)  
 if self.check\_sum == True:  
 self.valid\_function()  
 else:  
 self.total = float(txtDisplay.get())  
  
 def display(self, value):  
 txtDisplay.delete(0, END)  
 txtDisplay.insert(0, value)  
  
 def valid\_function(self):  
 if self.op == "add":  
 self.total += self.current  
 if self.op == "sub":  
 self.total -= self.current  
 if self.op == "multi":  
 self.total \*= self.current  
 if self.op == "divide":  
 self.total /= self.current  
 if self.op == "mod":  
 self.total %= self.current  
 self.input\_value = True  
 self.check\_sum = False  
 self.display(self.total)  
  
 def operation(self, op):  
 self.current = float(self.current)  
 if self.check\_sum:  
 self.valid\_function()  
 elif not self.result:  
 self.total = self.current  
 self.input\_value = True  
 self.check\_sum = True  
 self.op = op  
 self.result = False  
  
 def Clear\_Entry(self):  
 self.result = False  
 self.current = "0"  
 self.display(0)  
 self.input\_value = True  
  
 def All\_Clear\_Entry(self):  
 self.Clear\_Entry()  
 self.total = 0  
  
 def pi(self):  
 self.result = False  
 self.current = math.pi  
 self.display(self.current)  
  
 def tau(self):  
 self.result = False  
 self.current = math.tau  
 self.display(self.current)  
  
 def e(self):  
 self.result = False  
 self.current = math.e  
 self.display(self.current)  
  
 def mathPM(self):  
 self.result = False  
 self.current = -(float(txtDisplay.get()))  
 self.display(self.current)  
  
 def squared(self):  
 self.result = False  
 self.current = math.sqrt(float(txtDisplay.get()))  
 self.display(self.current)  
  
 def cos(self):  
 self.result = False  
 self.current = math.cos(math.radians(float(txtDisplay.get())))  
 self.display(self.current)  
  
 def cosh(self):  
 self.result = False  
 self.current = math.cosh(math.radians(float(txtDisplay.get())))  
 self.display(self.current)  
  
 def tan(self):  
 self.result = False  
 self.current = math.tan(math.radians(float(txtDisplay.get())))  
 self.display(self.current)  
  
 def tanh(self):  
 self.result = False  
 self.current = math.tanh(math.radians(float(txtDisplay.get())))  
 self.display(self.current)  
  
 def sin(self):  
 self.result = False  
 self.current = math.sin(math.radians(float(txtDisplay.get())))  
 self.display(self.current)  
  
 def sinh(self):  
 self.result = False  
 self.current = math.sinh(math.radians(float(txtDisplay.get())))  
 self.display(self.current)  
  
 def log(self):  
 self.result = False  
 self.current = math.log(float(txtDisplay.get()))  
 self.display(self.current)  
  
 def exp(self):  
 self.result = False  
 self.current = math.exp(float(txtDisplay.get()))  
 self.display(self.current)  
  
 def acosh(self):  
 self.result = False  
 self.current = math.acosh(float(txtDisplay.get()))  
 self.display(self.current)  
  
 def asinh(self):  
 self.result = False  
 self.current = math.asinh(float(txtDisplay.get()))  
 self.display(self.current)  
  
 def expm1(self):  
 self.result = False  
 self.current = math.expm1(float(txtDisplay.get()))  
 self.display(self.current)  
  
 def lgamma(self):  
 self.result = False  
 self.current = math.lgamma(float(txtDisplay.get()))  
 self.display(self.current)  
  
 def degrees(self):  
 self.result = False  
 self.current = math.degrees(float(txtDisplay.get()))  
 self.display(self.current)  
  
 def log2(self):  
 self.result = False  
 self.current = math.log2(float(txtDisplay.get()))  
 self.display(self.current)  
  
 def log10(self):  
 self.result = False  
 self.current = math.log10(float(txtDisplay.get()))  
 self.display(self.current)  
  
 def log1p(self):  
 self.result = False  
 self.current = math.log1p(float(txtDisplay.get()))  
 self.display(self.current)

### ****To display the text:****

added\_value = Calc()  
  
txtDisplay = Entry(calc, font=('Helvetica', 20, 'bold'),  
 bg='black', fg='white',  
 bd=30, width=28, justify=RIGHT)  
txtDisplay.grid(row=0, column=0, columnspan=4, pady=1)  
txtDisplay.insert(0, "0")

### ****Arranging the number pad:****

numberpad = "789456123"  
i = 0  
btn = []  
for j in range(2, 5):  
 for k in range(3):  
 btn.append(Button(calc, width=6, height=2,  
 bg='white', fg='Black',  
 font=('Helvetica', 20, 'bold'),  
 bd=4, text=numberpad[i]))  
 btn[i].grid(row=j, column=k, pady=1)  
 btn[i]["command"] = lambda x=numberpad[i]: added\_value.numberEnter(x)  
 i += 1

### ****Buttons to handle the Clear and Clear all operation:****

btnClear = Button(calc, text=chr(67), width=6,  
 height=2, bg='white',  
 font=('Helvetica', 20, 'bold')  
 , bd=4, command=added\_value.Clear\_Entry  
 ).grid(row=1, column=0, pady=1)  
  
btnAllClear = Button(calc, text=chr(67) + chr(69),  
 width=6, height=2,  
 bg='white',  
 font=('Helvetica', 20, 'bold'),  
 bd=4,  
 command=added\_value.All\_Clear\_Entry  
 ).grid(row=1, column=1, pady=1)

**Standard Calculator operations buttons:**

btnsq = Button(calc, text="\u221A", width=6, height=2,  
 bg='white', font=('Helvetica',  
 20, 'bold'),  
 bd=4, command=added\_value.squared  
 ).grid(row=1, column=2, pady=1)  
  
btnAdd = Button(calc, text="+", width=6, height=2,  
 bg='white',  
 font=('Helvetica', 20, 'bold'),  
 bd=4, command=lambda: added\_value.operation("add")  
 ).grid(row=1, column=3, pady=1)  
  
btnSub = Button(calc, text="-", width=6,  
 height=2, bg='white',  
 font=('Helvetica', 20, 'bold'),  
 bd=4, command=lambda: added\_value.operation("sub")  
 ).grid(row=2, column=3, pady=1)  
  
btnMul = Button(calc, text="x", width=6,  
 height=2, bg='white',  
 font=('Helvetica', 20, 'bold'),  
 bd=4, command=lambda: added\_value.operation("multi")  
 ).grid(row=3, column=3, pady=1)  
  
btnDiv = Button(calc, text="/", width=6,  
 height=2, bg='white',  
 font=('Helvetica', 20, 'bold'),  
 bd=4, command=lambda: added\_value.operation("divide")  
 ).grid(row=4, column=3, pady=1)  
  
btnDot = Button(calc, text=".", width=6,  
 height=2, bg='white', fg='Black',  
 font=('Helvetica', 20, 'bold'),  
 bd=4, command=lambda: added\_value.numberEnter(".")  
 ).grid(row=5, column=2, pady=1)  
  
btnZero = Button(calc, text="0", width=6,  
 height=2, bg='white', fg='Black',  
 font=('Helvetica', 20, 'bold'),  
 bd=4, command=lambda: added\_value.numberEnter(0)  
 ).grid(row=5, column=1, pady=1)  
  
btnPM = Button(calc, text=chr(177), width=6,  
 height=2, bg='white', font=('Helvetica', 20, 'bold'),  
 bd=4, command=added\_value.mathPM  
 ).grid(row=5, column=0, pady=1)  
  
btnEquals = Button(calc, text="=", width=6,  
 height=2, bg='white',  
 font=('Helvetica', 20, 'bold'),  
 bd=4, command=added\_value.sum\_of\_total  
 ).grid(row=5, column=3, pady=1)

### ****Buttons for row = 1****

btnPi = Button(calc, text="pi", width=6,  
 height=2, bg='white', fg='Black',  
 font=('Helvetica', 20, 'bold'),  
 bd=4, command=added\_value.pi  
 ).grid(row=1, column=4, pady=1)  
  
btnCos = Button(calc, text="Cos", width=6,  
 height=2, bg='white', fg='Black',  
 font=('Helvetica', 20, 'bold'),  
 bd=4, command=added\_value.cos  
 ).grid(row=1, column=5, pady=1)  
  
btntan = Button(calc, text="tan", width=6,  
 height=2, bg='white', fg='Black',  
 font=('Helvetica', 20, 'bold'),  
 bd=4, command=added\_value.tan  
 ).grid(row=1, column=6, pady=1)  
  
btnsin = Button(calc, text="sin", width=6,  
 height=2, bg='white', fg='Black',  
 font=('Helvetica', 20, 'bold'),  
 bd=4, command=added\_value.sin  
 ).grid(row=1, column=7, pady=1)

### ****Buttons for row - 2****

btn2Pi = Button(calc, text="2pi", width=6,  
 height=2, bg='white', fg='Black',  
 font=('Helvetica', 20, 'bold'),  
 bd=4, command=added\_value.tau  
 ).grid(row=2, column=4, pady=1)  
  
btnCosh = Button(calc, text="Cosh", width=6,  
 height=2, bg='white', fg='Black',  
 font=('Helvetica', 20, 'bold'),  
 bd=4, command=added\_value.cosh  
 ).grid(row=2, column=5, pady=1)  
  
btntanh = Button(calc, text="tanh", width=6,  
 height=2, bg='white', fg='Black',  
 font=('Helvetica', 20, 'bold'),  
 bd=4, command=added\_value.tanh  
 ).grid(row=2, column=6, pady=1)  
  
btnsinh = Button(calc, text="sinh", width=6,  
 height=2, bg='white', fg='Black',  
 font=('Helvetica', 20, 'bold'),  
 bd=4, command=added\_value.sinh  
 ).grid(row=2, column=7, pady=1)

### ****Buttons for row - 3****

btnlog = Button(calc, text="log", width=6,  
 height=2, bg='white', fg='Black',  
 font=('Helvetica', 20, 'bold'),  
 bd=4, command=added\_value.log  
 ).grid(row=3, column=4, pady=1)  
  
btnExp = Button(calc, text="exp", width=6, height=2,  
 bg='white', fg='Black',  
 font=('Helvetica',20, 'bold'),  
 bd=4, command=added\_value.exp  
 ).grid(row=3, column=5, pady=1)  
  
btnMod = Button(calc, text="Mod", width=6,  
 height=2, bg='white', fg='Black',  
 font=('Helvetica', 20, 'bold'),  
 bd=4, command=lambda: added\_value.operation("mod")  
 ).grid(row=3, column=6, pady=1)  
  
btnE = Button(calc, text="e", width=6,  
 height=2, bg='white', fg='Black',  
 font=('Helvetica', 20, 'bold'),  
 bd=4, command=added\_value.e  
 ).grid(row=3, column=7, pady=1)

### ****Buttons for row - 4****

btnlog10 = Button(calc, text="log10", width=6,  
 height=2, bg='white', fg='Black',  
 font=('Helvetica', 20, 'bold'),  
 bd=4, command=added\_value.log10  
 ).grid(row=4, column=4, pady=1)  
  
btncos = Button(calc, text="log1p", width=6,  
 height=2, bg='white', fg='Black',  
 font=('Helvetica', 20, 'bold'),  
 bd=4, command=added\_value.log1p  
 ).grid(row=4, column=5, pady=1)  
  
btnexpm1 = Button(calc, text="expm1", width=6,  
 height=2, bg='white', fg='Black',  
 font=('Helvetica', 20, 'bold'),  
 bd=4, command=added\_value.expm1  
 ).grid(row=4, column=6, pady=1)  
  
btngamma = Button(calc, text="gamma", width=6,  
 height=2, bg='white', fg='Black',  
 font=('Helvetica', 20, 'bold'),  
 bd=4, command=added\_value.lgamma  
 ).grid(row=4, column=7, pady=1)

### ****Buttons for row - 5****

btnlog2 = Button(calc, text="log2", width=6,  
 height=2, bg='white', fg='Black',  
 font=('Helvetica', 20, 'bold'),  
 bd=4, command=added\_value.log2  
 ).grid(row=5, column=4, pady=1)  
  
btndeg = Button(calc, text="deg", width=6,  
 height=2, bg='white', fg='Black',  
 font=('Helvetica', 20, 'bold'),  
 bd=4, command=added\_value.degrees  
 ).grid(row=5, column=5, pady=1)  
  
btnacosh = Button(calc, text="acosh", width=6,  
 height=2, bg='white', fg='Black',  
 font=('Helvetica', 20, 'bold'),  
 bd=4, command=added\_value.acosh  
 ).grid(row=5, column=6, pady=1)  
  
btnasinh = Button(calc, text="asinh", width=6,  
 height=2, bg='white', fg='Black',  
 font=('Helvetica', 20, 'bold'),  
 bd=4, command=added\_value.asinh  
 ).grid(row=5, column=7, pady=1)  
  
lblDisplay = Label(calc, text="Scientific Calculator",  
 font=('Times New Roman', 38, 'bold','underline'),  
 bg='black', fg='white', justify=CENTER)  
  
lblDisplay.grid(row=0, column=4, columnspan=4)

### ****Menu Bar Options:****

def iExit():  
 iExit = tkinter.messagebox.askyesno("Scientific Calculator",  
 "Do you want to exit ?")  
 if iExit > 0:  
 root.destroy()  
 return  
  
  
def Scientific():  
 root.resizable(width=False, height=False)  
 root.geometry("944x568")  
  
  
def Standard():  
 root.resizable(width=False, height=False)  
 root.geometry("480x568")  
  
  
menubar = Menu(calc)

### ****Sub-Menu Bar Options:****

# ManuBar 1 :  
filemenu = Menu(menubar, tearoff=0)  
menubar.add\_cascade(label='File', menu=filemenu)  
filemenu.add\_command(label="Standard", command=Standard)  
filemenu.add\_command(label="Scientific", command=Scientific)  
filemenu.add\_separator()  
filemenu.add\_command(label="Exit", command=iExit)  
  
# ManuBar 2 :  
editmenu = Menu(menubar, tearoff=0)  
menubar.add\_cascade(label='Edit', menu=editmenu)  
editmenu.add\_command(label="Cut")  
editmenu.add\_command(label="Copy")  
editmenu.add\_separator()  
editmenu.add\_command(label="Paste")  
  
root.config(menu=menubar)  
  
root.mainloop()

***Thank You.,***