```
import tkinter as tk
from math import *

# used to switch between units of rad, and deg
convert_constant = 1
inverse_convert_constant = 1

btn_params = {
    'padx': 16,
    'pady': 1,
    'bd': 4,
    'fg': 'white',
    'bg': '#666666',
    'font': ('arial', 18),
    'width': 2,
    'height': 2,
    'relief': 'flat',
    'activebackground': "#666666"
}


def fsin(arg):
    return sin(arg * convert_constant)
```

```
def fcos(arg):
    return cos(arg * convert_constant)



def ftan(arg):
    return tan(arg * convert_constant)



def arcsin(arg):
    return inverse_convert_constant * (asin(arg))



def arccos(arg):
    return inverse_convert_constant * (acos(arg))



def arctan(arg):
    return inverse_convert_constant * (atan(arg))



class Calculator:
    def __init__(self, master):
        # expression that will be displayed on screen
        self.expression = ""
        # be used to store data in memory
```

```
        self.recall = ""
        # self.answer
        self.sum_up = ""
        # create string for text input
        self.text_input = tk.StringVar()
        # assign instance to master
        self.master = master
        # set frame showing inputs and title
        top_frame = tk.Frame(master, width=650, height=20, bd=4, relief='flat', bg='#666666')
        top_frame.pack(side=tk.TOP)
        # set frame showing all buttons
        bottom_frame = tk.Frame(master, width=650, height=470, bd=4, relief='flat', bg='#666666')
        bottom_frame.pack(side=tk.BOTTOM)
        # name of calculator
        my_item = tk.Label(top_frame, text="Simple Scientific Calculator",
                           font=('arial', 14), fg='white', width=26, bg='#666666')
        my_item.pack()
        # entry interface for inputs
        txt_display = tk.Entry(top_frame, font=('arial', 36), relief='flat',
                               bg='#666666', fg='white', textvariable=self.text_input, width=60, bd=4, justify='right')
        txt_display.pack()


        # row 0
        # left bracket button
        self.btn_left_brack = tk.Button(bottom_frame, **btn_params, text="(", command=lambda:
self.btn_click('('))
```

```
        self.btn_left_brack.grid(row=0, column=0)

        # right bracket button

        self.btn_right_brack = tk.Button(bottom_frame, **btn_params, text=")", command=lambda:
self.btn_click(')'))

        self.btn_right_brack.grid(row=0, column=1)

        # takes e to some exponent that you insert into the function

        self.btn_exp = tk.Button(bottom_frame, **btn_params, text="exp", command=lambda:
self.btn_click('exp('))

        self.btn_exp.grid(row=0, column=2)

        # constant pi

        self.btn_pi = tk.Button(bottom_frame, **btn_params, text="π", command=lambda:
self.btn_click('pi'))

        self.btn_pi.grid(row=0, column=3)

        # clears self.expression

        self.btn_clear = tk.Button(bottom_frame, **btn_params, text="C", command=self.btn_clear_all)

        self.btn_clear.grid(row=0, column=4)

        # deletes last string input

        self.btn_del = tk.Button(bottom_frame, **btn_params, text="del", command=self.btn_clear1)

        self.btn_del.grid(row=0, column=5)

        # inputs a negative sign to the next entry

        self.btn_change_sign = tk.Button(bottom_frame, **btn_params, text="+/-",
command=self.change_signs)

        self.btn_change_sign.grid(row=0, column=6)

        # division

        self.btn_div = tk.Button(bottom_frame, **btn_params, text="/", command=lambda:
self.btn_click('/'))

        self.btn_div.grid(row=0, column=7)
```

```
        # square root
        self.btn_sqrt = tk.Button(bottom_frame, **btn_params, text="sqrt", command=lambda:
self.btn_click('sqrt('))
        self.btn_sqrt.grid(row=0, column=8)
        # row 1
        # changes trig function outputs to degrees
        self.btn_Deg = tk.Button(bottom_frame, **btn_params, activeforeground='orange', 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.Button(bottom_frame, **btn_params, foreground='orange',
activeforeground='orange', text="Rad",
                                 command=self.convert_rad)
        self.btn_Rad.grid(row=1, column=1)
        # cubes a value
        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(bottom_frame, **btn_params, text="abs", command=lambda:
self.btn_click('abs' + '('))
        self.btn_abs.grid(row=1, column=3)
        # seven
        self.btn_7 = tk.Button(bottom_frame, **btn_params, text="7", command=lambda: self.btn_click(7))
        self.btn_7.configure(activebackground="#4d4d4d", bg='#4d4d4d')
        self.btn_7.grid(row=1, column=4)
        # eight
```

```
        self.btn_8 = tk.Button(bottom_frame, **btn_params, text="8", command=lambda: self.btn_click(8))

        self.btn_8.configure(activebackground="#4d4d4d", bg='#4d4d4d')

        self.btn_8.grid(row=1, column=5)

        # nine

        self.btn_9 = tk.Button(bottom_frame, **btn_params, text="9", command=lambda: self.btn_click(9))

        self.btn_9.configure(activebackground="#4d4d4d", bg='#4d4d4d')

        self.btn_9.grid(row=1, column=6)

        # multiplication

        self.btn_mult = tk.Button(bottom_frame, **btn_params, text="x", command=lambda:
self.btn_click('*'))

        self.btn_mult.grid(row=1, column=7)

        # 'memory clear' button. Wipes self.recall to an empty string

        self.btn_MC = tk.Button(bottom_frame, **btn_params, text="MC", command=self.memory_clear)

        self.btn_MC.grid(row=1, column=8)

        # row 2

        # sin function that returns value from -1 to 1 by default

        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 default

        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)
```

```
        #
        self.btn_log = tk.Button(bottom_frame, **btn_params, text="log", command=lambda:
self.btn_click('log('))
        self.btn_log.grid(row=2, column=3)
        # four
        self.btn_4 = tk.Button(bottom_frame, **btn_params, text="4", command=lambda: self.btn_click(4))
        self.btn_4.configure(activebackground="#4d4d4d", bg='#4d4d4d')
        self.btn_4.grid(row=2, column=4)
        # five
        self.btn_5 = tk.Button(bottom_frame, **btn_params, text="5", command=lambda: self.btn_click(5))
        self.btn_5.configure(activebackground="#4d4d4d", bg='#4d4d4d')
        self.btn_5.grid(row=2, column=5)
        # six
        self.btn_6 = tk.Button(bottom_frame, **btn_params, text="6", command=lambda: self.btn_click(6))
        self.btn_6.configure(activebackground="#4d4d4d", bg='#4d4d4d')
        self.btn_6.grid(row=2, column=6)
        # subtraction
        self.btnSub = tk.Button(bottom_frame, **btn_params, text="-", command=lambda: self.btn_click('-
'))
        self.btnSub.grid(row=2, column=7)
        # 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=8)
        # row 3
        # sin inverse function
        self.btn_sin_inverse = tk.Button(bottom_frame, **btn_params, text=u"sin-\u00B9",
```

```
                    command=lambda: self.btn_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_inverse.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.grid(row=3, column=2)
        # takes the natural log
        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)
        # one
        self.btn_1 = tk.Button(bottom_frame, **btn_params, text="1", command=lambda: self.btn_click(1))
        self.btn_1.configure(activebackground="#4d4d4d", bg='#4d4d4d')
        self.btn_1.grid(row=3, column=4)
        # two
        self.btn_2 = tk.Button(bottom_frame, **btn_params, text="2", command=lambda: self.btn_click(2))
        self.btn_2.configure(activebackground="#4d4d4d", bg='#4d4d4d')
        self.btn_2.grid(row=3, column=5)
        # three
        self.btn_3 = tk.Button(bottom_frame, **btn_params, text="3", command=lambda: self.btn_click(3))
        self.btn_3.configure(activebackground="#4d4d4d", bg='#4d4d4d')
        self.btn_3.grid(row=3, column=6)
```

```
        # addition

        self.btn_add = tk.Button(bottom_frame, **btn_params, text="+", command=lambda:
self.btn_click('+'))

        self.btn_add.grid(row=3, column=7)

        # adds current self.expression to self.recall string

        self.btn_M_plus = tk.Button(bottom_frame, **btn_params, text="M+",
command=self.memory_add)

        self.btn_M_plus.grid(row=3, column=8)

        # row 4

        # factorial function

        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)

        # square function

        self.btn_sqr = tk.Button(bottom_frame, **btn_params, text=u"x\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)

        # zero

        self.btn_0 = tk.Button(bottom_frame, **btn_params, text="0", command=lambda: self.btn_click(0))

        self.btn_0.configure(activebackground="#4d4d4d", bg='#4d4d4d', width=7, bd=5)
```

```
        self.btn_0.grid(row=4, column=4, columnspan=2)

        # equals button

        self.btn_eq = tk.Button(bottom_frame, **btn_params, text="=", command=self.btn_equal)

        self.btn_eq.configure(bg='#ff9980', activebackground='#ff9980')

        self.btn_eq.grid(row=4, column=6)

        # decimal to convert to float

        self.btn_dec = tk.Button(bottom_frame, **btn_params, text=".", command=lambda:
self.btn_click('.'))

        self.btn_dec.grid(row=4, column=7)

        # 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=8)


    # functions

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


    def btn_click(self, expression_val):

        if len(self.expression) >= 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)


    # clears last item in string
```

```
def btn_clear1(self):
    self.expression = self.expression[:-1]
    self.text_input.set(self.expression)

# adds in a negative sign

def change_signs(self):
    self.expression = self.expression + '-'
    self.text_input.set(self.expression)

# clears memory_recall

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_up
```

```
    self.expression = self.expression + self.answer
    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):
    global convert_constant
    global inverse_convert_constant
    convert_constant = pi / 180
    inverse_convert_constant = 180 / pi
    self.btn_Rad["foreground"] = 'white'
    self.btn_Deg["foreground"] = 'orange'

def convert_rad(self):
    global convert_constant
    global inverse_convert_constant
    convert_constant = 1
```

```
        inverse_convert_constant = 1
        self.btn_Rad["foreground"] = 'orange'
        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_up


# tkinter layout
root = tk.Tk()
b = Calculator(root)
root.title("Simple Scientific Calculator")
root.geometry("650x490+50+50")
root.resizable(False, False)
root.mainloop()
```