```
######################
# Thomas Guldentops  #
# TFE - Programmation#
# Juin 2018          #
######################

from tkinter import *
from tkinter.messagebox import showerror
from math import *
from Notepad_project_tfe import *

class Calculatrice(object):
    """ Création d'une application type calculatrice basique & scientifique """
    def __init__(self, fen_largeur=400, fen_hauteur=300):
        """ Définition de la méthode constructeur"""
        self.fen = Tk()
        self.fen.title("Calculatrice")
        self.fen.iconbitmap("images/image2.ico")
        self.fen.configure(bg = "#282828")

        self.fen.resizable(width=False, height=False)
        self.fen.geometry("{0}x{1}".format(str(fen_largeur), str(fen_hauteur)))
        self.fen_largeur = fen_largeur
        self.fen_hauteur = fen_hauteur

        self.original_background = self.fen.cget("background")
        self.couleur = "#000000"
        self.entree = Entry(self.fen, width = 45, fg=self.couleur)
        self.entree.bind("<Return>", self.bouttonEgale)
        self.entree.grid(row=0,column=0,columnspan=6,pady=3)
        self.entree.focus_set()

        menubar = Menu(self.fen)
        self.fen.config(menu=menubar)
        menufichier= Menu(menubar,tearoff=0)
        menubar.add_cascade(label="Affichage", menu=menufichier)
        menufichier.add_command(label="Standard ", command = self.creationBouttons)
        menufichier.add_command(label="Scientifique ", command = self.calculatriceScientifique)
        menufichier.add_separator()
        menufichier.add_command(label="Options ", command = self.options)


        self.fen.config(menu= menubar)
        menuNotepad = Menu(menubar, tearoff=0)
        menubar.add_cascade(label="Bloc Note", menu=menuNotepad)
        menuNotepad.add_command(label="Bloc Note ", command = Notepad)

        self.creationBouttons()
        self.fen.mainloop()


    def toucheClavier(self,touche):
        """ création de la valeur des touches"""
        return self.entree.insert('end',touche)

    def bouttonEgale(self, *event):
        """ Création du résultat des entrées."""
        self.solution = self.entree.get()
        self.entree.delete(0, 'end')
        try:
            self.solution = self.entree.insert(0, str(eval(self.solution)))
        except:
            self.entree.insert(0, "Syntax Error")
            self.fen.after(500, self.bouttonToutEffacer)
        return self.solution

    def bouttonEffacerUn(self):
        """ Permet de effacer un chiffre entrée"""
        self.effacer = self.entree.get()[:-1]
        self.entree.delete(0,'end')
        self.entree.insert(0, self.effacer)
        return self.effacer

    def bouttonToutEffacer(self):
        """ Permet d'effacer tout se qu'il se trouve dans l'entry"""
        self.supprimer = self.entree.delete(0, 'end')
        return self.supprimer

    def racineCarree(self):
        self.racine = self.entree.get()
        self.entree.delete(0, 'end')
        try:
            self.racine = sqrt(float(self.racine))
            self.entree.insert(0, self.racine)
        except:
            self.entree.insert(0, "Syntax Error")
            self.fen.after(500, self.bouttonToutEffacer)
        return self.racine

    def puissanceDeux(self):
        self.puissance = self.entree.get()
        self.entree.delete(0, 'end')
        try:
            self.puissance = float(self.puissance)**2
            self.entree.insert(0, self.puissance)
        except:
            self.entree.insert(0, "Syntax Error")
```

```
            self.fen.after(500, self.bouttonToutEffacer)
        return self.puissance

    def creationBouttons(self, tailleBouttons = 8, couleurTouche = "#FDF1B8"):
        """ Creation des Bouttons"""
        self.tailleBouttons = tailleBouttons
        self.couleurTouche = couleurTouche
        try:
            self.toucheRacineCarree.destroy()
            self.touchePuissanceDeux.destroy()
        except:
            #Touches de 0 à  9
            self.touche1 = Button(self.fen, text='1', command=lambda:self.toucheClavier(1), relief='flat',width = self.tailleBouttons,
            self.touche1.grid(row=4, column=0,padx=5, pady=5 )
            self.original_button_background = self.touche1.cget("background")
            self.touche2 = Button(self.fen, text='2', command=lambda:self.toucheClavier(2), relief='flat',width = self.tailleBouttons,
            self.touche2.grid(row=4, column=1,padx=5, pady=5)
            self.touche3 = Button(self.fen, text='3', command=lambda:self.toucheClavier(3), relief='flat',width = self.tailleBouttons,
            self.touche3.grid(row=4, column=2,padx=5, pady=5)
            self.touche4 = Button(self.fen, text='4', command=lambda:self.toucheClavier(4), relief='flat',width = self.tailleBouttons,
            self.touche4.grid(row=3, column=0,padx=5, pady=5)
            self.touche5 = Button(self.fen, text='5', command=lambda:self.toucheClavier(5), relief='flat',width = self.tailleBouttons,
            self.touche5.grid(row=3, column=1,padx=5, pady=5)
            self.touche6 = Button(self.fen, text='6', command=lambda:self.toucheClavier(6), relief='flat',width = self.tailleBouttons,
            self.touche6.grid(row=3, column=2,padx=5, pady=5)
            self.touche7 = Button(self.fen, text='7', command=lambda:self.toucheClavier(7), relief='flat',width = self.tailleBouttons,
            self.touche7.grid(row=2, column=0,padx=5, pady=5)
            self.touche8 = Button(self.fen, text='8', command=lambda:self.toucheClavier(8), relief='flat',width = self.tailleBouttons,
            self.touche8.grid(row=2, column=1,padx=5, pady=5)
            self.touche9 = Button(self.fen, text='9', command=lambda:self.toucheClavier(9), relief='flat',width = self.tailleBouttons,
            self.touche9.grid(row=2, column=2,padx=5, pady=5)
            self.touche0 = Button(self.fen, text='0', command=lambda:self.toucheClavier(0), relief='flat',width = self.tailleBouttons
            self.touche0.grid(row=5, column=0, columnspan = 2)

            #Touches opérations
            self.toucheM = Button(self.fen, text="-", command=lambda:self.toucheClavier('-'), relief='flat',width = self.tailleBouttor
            self.toucheM.grid(row=3, column=3,padx=5, pady=5)
            self.toucheP = Button(self.fen, text="+", command=lambda:self.toucheClavier('+'), relief='flat',width = self.tailleBouttor
            self.toucheP.grid(row=4, column=3,padx=5, pady=5)
            self.toucheF = Button(self.fen, text="x", command=lambda:self.toucheClavier('*'), relief='flat',width = self.tailleBouttor
            self.toucheF.grid(row=2, column=3,padx=5, pady=5)
            self.toucheD = Button(self.fen, text="/", command=lambda:self.toucheClavier('/'), relief='flat',width = self.tailleBouttor
            self.toucheD.grid(row=1, column=3,padx=5, pady=5)

            self.touchePo = Button(self.fen, text=".", command=lambda:self.toucheClavier('.'), relief='flat',width = self.tailleBoutto
            self.touchePo.grid(row=5, column=2,padx=5, pady=5)
            self.toucheE = Button(self.fen, text='=', command=lambda:self.bouttonEgale(), relief='flat', width = self.tailleBouttons,h
            self.toucheE.grid(row=5, column=3)
            self.toucheEfTout = Button(self.fen, text='AC',command=lambda:self.bouttonToutEffacer(), relief='flat',width = self.taille
            self.toucheEfTout.grid(row=1, column=0,padx=5, pady=5)
            self.toucheEfUn = Button(self.fen, text='C',command=lambda:self.bouttonEffacerUn(), relief='flat',width = self.tailleBoutt
            self.toucheEfUn.grid(row=1, column=1,padx=5, pady=5)

    # Une fonction lambda est une fonction qui prend un nombre quelconque d'arguments (y compris des arguments optionnels) et retourne
    # Les fonctions lambda ne peuvent pas contenir de commandes et elles ne peuvent contenir plus d'une expression.
    # N'essayez pas de mettre trop de choses dans une fonction lambda, si vous avez besoin de quelque chose de complexe, définissez pl

    def calculatriceScientifique(self, tailleBouttons = 8, couleurTouche = "#FDF1B8"):
        self.tailleBouttons = tailleBouttons
        self.couleurTouche = couleurTouche

        self.toucheRacineCarree = Button(self.fen, text='√',command=lambda:self.racineCarree(), relief='flat',width = self.tailleBoutt
        self.toucheRacineCarree.grid(row=2, column=4,padx=5, pady=5)

        self.touchePuissanceDeux = Button(self.fen, text='²',command=lambda:self.puissanceDeux(), relief='flat',width = self.tailleBou
        self.touchePuissanceDeux.grid(row=3, column=4,padx=5, pady=5)


    ##########################
    # Options                #
    ##########################
    def options(self):
        """ Menu options. Je n'ai pas trouver d'option utile à rajouter dans une calculatrice. J'avais mit avant une personnalisation
        self.fenOption = Toplevel(self.fen)
        self.fenOption.title("Options")
        self.fenOption.configure(bg = "#282828")
        self.fenOption.geometry("{0}x{1}".format(str(self.fen_largeur), str(self.fen_hauteur)))
        self.fenOption.resizable(width=False, height=False)



if __name__== "__main__":
    Calculatrice()
```