**《Python程序设计》**

**课程设计报告**

　　　　　　　　　　（2018 —2019 学年第 一 学期）

题 目 简易计算器

学生姓名 杨 中 义

专业班级 信管1601

学生学号 311609030120

教师姓名 徐 文 鹏

成 绩**：**

评 语：

教师签名：

日期：

目录

**[一、 设计目的 1](#_Toc17765_WPSOffice_Level1)**

[1.1课程设计教学目的 1](#_Toc15177_WPSOffice_Level2)

1.2本课程设计目的 1

**[二、 设计任务和要求 2](#_Toc13063_WPSOffice_Level1)**

[2.1课程设计教学任务和要求 2](#_Toc13063_WPSOffice_Level2)

[2.2本课程设计具体任务和要求 2](#_Toc6151_WPSOffice_Level2)

**[三、总体设计 2](#_Toc6151_WPSOffice_Level1)**

[3.1软件功能图 2](#_Toc9118_WPSOffice_Level2)

[3.2软件流程图 3](#_Toc15177_WPSOffice_Level2)

3.3 任务分配 3

**[四、设计实现 4](#_Toc9118_WPSOffice_Level1)**

[4.1最终实现结果 4](#_Toc18018_WPSOffice_Level2)

[4.2实现结果评价 5](#_Toc27255_WPSOffice_Level2)

**[五、详细设计 6](#_Toc15177_WPSOffice_Level1)**

[5.1输入输出分析 6](#_Toc18018_WPSOffice_Level2)

5.2程序流程图 6

5.3 有关算法的详细说明 7

**[六、调试和测试 7](#_Toc18018_WPSOffice_Level1)**

**[七、设计总结 9](#_Toc27255_WPSOffice_Level1)**

**[八、附录 1](#_Toc15116_WPSOffice_Level1)0**

1. **设计目的：**  
    1.1课程设计教学目的

本课程设计是本专业的一门重要实践性教学环节。在学习了专业基础课和《Python程序设计》课程的基础上，本课程设计旨在加深对Python程序设计的认识，对Python语言及其语言生态有一个进一步的掌握和应用，学会运用Python标准库及外接相关库来解决实际问题的基本能力，培养和提高学生分析问题、解决问题的能力，尤其是提高学生使用Python为开发语言来进行问题描述、交流与思考的能力，为毕业设计和以后的工程实践打下良好的基础。  
  1.2本课程设计目的

Python是一种面向对象、直译式计算机程序设计语言，也是一种功能强大而完善的通用型语言，已经具有十多年的发展历史，成熟且稳定。这种语言具有非常简捷而清晰的语法特点，适合完成各种高层任务，几乎可以在所有的操作系统中运行。目前，基于这种语言的相关技术正在飞速的发展，用户数量急剧扩大，相关的资源非常多。

在本学期的学习过程中，先后学习了python的基本语句、函数、模块、类以及异常，对python语言有了一个初步的了解，并对面向对象编程有了一个整体的认识。在这里，利用课堂上所学习的python知识，编写程序客户端与服务器的运作流程。

本次课程设计简易计算器的制作是由我们小组三个人经过商议之后，综合考虑自身的能力后共同决定的，因为计算器在我们的生活中非常常见并且很有用，当代的人类生活中，我们经常会遇到数字数据比较大且数据复杂的数字运算，心算笔算起来很复杂困难，这时候就要想到用到计算器。通过本程序设计将要实现一个简单的计算器。计算生活中的数字已经成了必不可少的一段生活过程了,而随着生活水平的提高，更加繁琐的计算的已经成了必不可少的一部分了,现在生活的一部分仅仅靠自己的心算来计算已经是满足不了的。所以我们设计了这个简易计算器程序并通过python的打包功能使普通的电脑也可以使用这个软件。并且通过这个简易程序的设计使我们对python的功能及用法有更多的了解和掌握，对于我们以后的编程发展也有很大的好处。虽然我们的课程设计作品相对简单，但是同样锻炼了我们的编程能力和团队合作能力。为我们今后的学习和工作积累了经验。

1. **设计任务和要求**

2.1课程设计教学任务和要求

本课程设计主要任务是以Python为开发语言完成一个100~300行左右规模的程序项目开发。

课程设计的基本要求是：在课程设计的各个阶段严格、规范地完成相关的文档，例如在初期按时完成设计目的、设计要求、总体设计，在后期完成详细设计、调试与测试、设计总结等。要求能完成所选题目的主要功能，程序运行健壮、正确，代码要有详细注释，可读性好；所写文档结构合理、内容完整、叙述清晰。更高要求是：有创意、系统界面美观。

由于课程设计项目具有一定的综合性，鼓励具有不同特长和不同能力的学生互相组队。项目小组自己推荐一名组长，实行“组长负责制”。组长组织组员进行项目选题、任务分配、方案确定、方案设计、系统调试测试，组员分工协作。小组成员开展项目讨论，互相支持，形成协作意识。

2.2本课程设计具体任务和要求

经过小组成员的讨论，为满足Python课程教学任务与要求，锻炼小组成员利用Python编写程序与自主学习的能力，我们小组各成员决定选择利用Python语言实现一个简易的计算器。

所要实现的简易计算器的功能如下：

（1）程序能实现简单的加减乘除取余等计算功能；

（2）程序由鼠标来输入，具有良好的用户界面；

（3）利用pyinstaller工具将程序打包成exe文件。

**三、总体设计**

3.1软件功能图

简单计算器

加减乘除取余运算

退格

清零

3.2软件流程图

开始

按下数字按钮

显示数字在文本框内

加减乘除运算

在文本框中显示结果

结束

3.3任务分配

设计内容：能实现简单的加减乘除取模等简单计算，可以使用鼠标来输入，用户界面美观简洁。能够使用pyinstaller工具将程序打包为exe文件。

任务分配情况：

余自立负责实现程序基本运算功能；

袁畅负责实现鼠标点击输入功能和程序打包；

杨中义负责用户图形界面设计和代码修改等工作。

**四、设计实现**

4.1最终实现结果

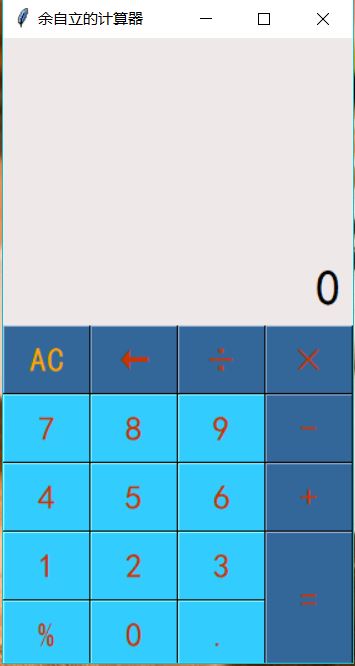


图1.程序运行初始界面 图2.计算器执行加操作



图3.计算器执行减操作 图4. 计算器执行乘操作



图5.计算器执行除操作 图6.计算器执行除操作

4.2实现结果评价

首先，在拿到了老师给的课程设计指导书后，我们小组仔细、认真的研读，综合考虑自身的编程能力，实事求是选定课程设计备选题目中的第一题就开始着手准备。然而，题目虽然简单但是想要做好课程设计也不是那么简单，我们使用IDLE编写Python代码。在百度、google和查阅相关书籍一番后，并对简单的程序编写和调试，其次，在课程设计中，有很多的、复杂的数据计算，这就必须要求我心细谨慎。更重要的是，在对软件编写程序的时候，每一个字母，每一个标点符号都要求做到百分之百。因此，我也由一开始的热情高涨变得浮躁，开始抓狂。在同学们的鼓励支持下，我又重获自信，细化目标，一步一步的沿着小小的目标进发，逐步地检查计算结果和编写，修改，调试程序。最终，完善课程设计中的任务。

再次，我们在课程设计中，锻炼了我们的编程能力和团队合作能力。我认为此次课程设计不仅磨练了我们的耐心，还增加了我们对自动控制学习的兴趣，虽然有汗水，有心酸，但我知道这一切都值得的，而且我还会继续坚持的走下去。同时也要感谢老师对我们的帮助和同学们对我们的鼓励。

设计完成结果小组自我评价：良好

**五、详细设计**

我主要负责的是用户图形界面设计和代码修改等工作，在进行初始的界面设计时，我查找了一些python关于设计图形界面的第三方库，最终决定采用tkinter标准库来进行图形界面的设计，tkinter库是python关于图形界面设计的非常常用的第三方库，对于python的图形界面设计非常有用。选定使用的第三方库后，我就开始构思计算器主界面的设计，刚开始我也想过把计算器设计成正方形或者其他形状，但之后我无意间看到了手机上的计算器，我发现手机上的计算器模型看着很不错，于是就觉得按照手机上计算器模型来设计简易计算器的整体框架，在整体的框架确定之后，我就开始具体的界面的设计，经过和小组的成员进行讨论之后，我们决定设计两个显示面板，一个用于显示计算过程，一个用于显示计算后的结果，然后就是数字按键和字符按键的设置，经过讨论，我们也决定了下来。因为是简易计算器的设置，所以我们也没有决定弄得太过复杂，关于字符的设置，我们就简单的设计了一些用于计算加减乘除的字符。用于简易计算。

5.1输入输出分析**：**运行程序后，打开的界面就是一个简易计算器，在显示计算结果的界面上会显示0，这是我进行程序设计时加入的一个默认值。当你要进行计算时，只需要用鼠标点击要计算的数字和运算符号进行运算即可。比如要计算”8+9”,只需要以此点击“8”,”+”,和”9”即可，然后在用于显示结果的显示板上就会显示数字17，而在用于显示计算过程的面板上则会显示运算过程即”8+9”。

**5.2**程序流程图如下：

程序界面设置

主界面

显示面板2

显示面板1

字符面板

数

字

面

板

5.3有关算法的详细说明：在我负责的代码部分，在代码的开始就是import tkinter，这就引入了python的第三方库tkinter，Tkinter 是 Python 的标准 GUI 库，**Tkinter中的** Tkinter 模块(Tk 接口)是 Python 的标准 Tk GUI 工具包的接口 .Tk 和 Tkinter 可以在大多数的 Unix 平台下使用,同样可以应用在 Windows 和 Macintosh 系统里。Tk8.0 的后续版本可以实现本地窗口风格,并良好地运行在绝大多数平台中。Python 使用 Tkinter 可以快速的创建 GUI 应用程序,图形化用户界面。然后进行主窗口的设置，包括主窗口的生成，窗口大小的设置以及主窗口名字即计算器名字的设置。

在具体界面布局的设置中，我主要使用了StringVar方法，StringVar是Tk库内部定义的字符串变量类型，在这里用于管理部件上面的字符；不过一般用在按钮button上。改变StringVar，按钮上的文字也随之改变.然后对显示面板进行具体的设置，包括大小，字体，背景色等的设置。

最后对数字字符和运算字符进行位置的设置，通过使用place()函数固定各个数字和运算字符的位置及高度和宽度的大小。place()函数是tkinter中一个非常常用的函数，它能够对使得各个组件按照一定的次序依次排列，这对于我们学习python的图形化界面管理非常有用，而且非常方便。这些就是我对我负责部分的算法具体说明。

**六、调试和测试**

在程序的调试阶段，因为我们设计的简易计算器总体上来说是比较简单的，因此大部分是在python的IDLE环境下调试的。IDLE是开发 python 程序的基本IDE（[集成开发环境](https://baike.baidu.com/item/%E9%9B%86%E6%88%90%E5%BC%80%E5%8F%91%E7%8E%AF%E5%A2%83" \t "_blank)），具备基本的IDE的功能，是非商业Python开发的不错的选择。当安装好python以后，IDLE就自动安装好了，不需要另外去找。同时，使用Eclipse这个强大的框架式IDLE也可以非常方便的调试Python程序。基本功能：语法加亮；段落缩进；基本文本编辑；TABLE键控制；调试程序。

当我将主界面和显示面板的代码写好后进行调试后，计算器的显示总体上符合我的要求，大小和位置也都比较合适。调试时的一些关键代码如下:

import tkinter

root=tkinter.Tk('red')

root.minsize(280,500)

root.title('余自立的计算器')

result = tkinter.StringVar()

result.set(0)

result2 = tkinter.StringVar()

result2.set('')

label = tkinter.Label(root,font = ('黑体',20),bg = '#EEE9E9',bd ='9',fg = '#828282',anchor = 'se',textvariable = result2)# Anchor 标签中文本的位置 顶对齐'n',底对齐's',左'w',右'e'；background(bg)背景色； foreground(fg)前景色；borderwidth(bd)边框宽度；textvariable 可变文本，与StringVar等配合着用

label.place(width = 280,height = 170)；

label2 = tkinter.Label(root,font = ('黑体',30),bg = '#EEE9E9',bd ='9',fg = 'black',anchor = 'se',textvariable = result)

label2.place(y = 170,width = 280,height = 60)#y:

有了总体设计以后，我就开始对具体的小模块进行具体的设置，就拿显示面板来说吧，除了要设计具体的大小，我还要对里面具体的显示部分进行详细设计，包括字体的大小设置，字体的背景色设置以及字体前景色等的设置，一些具体的代码如下：

btn0.place(x = 70,y = 450,width = 70,height = 55)

btn1.place(x = 0,y = 395,width = 70,height = 55)

btn2.place(x = 70,y = 395,width = 70,height = 55)

btn3.place(x = 140,y = 395,width = 70,height = 55)

btn4.place(x = 0,y = 340,width = 70,height = 55)

btn5.place(x = 70,y = 340,width = 70,height = 55)

btn6.place(x = 140,y = 340,width = 70,height = 55)

btn7.place(x = 0,y = 285,width = 70,height = 55)

btn8.place(x = 70,y = 285,width = 70,height = 55)

btn9.place(x = 140,y = 285,width = 70,height = 55)

在具体的设计中也曾出现一些问题，在进行字体的设置中，因为字体需要的属性比较多，比如背景色、前景色等，这些属性中有些是我以前学过的，用起来比较简单，但有一些是我没有听过的，这就给我造成了比较大的困扰，但是通过查阅书籍资料以及上网查找资料等方法最终解决了这个不大不小的问题。这个事情也给了我一些感悟，在我们的学习的过程中没有过不去的坎，只要我们肯努力，肯去学习，不管多难的问题总会找到解决的方法的。

**七、设计总结**

此次的课程设计对我们三个的收获都非常的大，我们用我们所学习的python知识共同设计了一个计算器，虽然它仅仅是一个非常简易的计算器，只能计算一些非常简单的计算，但这是凝聚了我们的心血和包含了我们的努力的作品，这让我们感到非常的骄傲和自豪。这次做课程设计使我受益良多。一方面,通过这次实验的完成,我对Python有了更深刻的理解,能够更加熟练的运用这门语言,更加理解到了编程的魅力。而另一方面,使我明白了认真做事的重要性。使我体会到完成一件事的喜悦。以后面对困难的时候,都能有直面困难并克服它的决心。本设计能够顺利的完成,也感谢老师的辅导和同学们的互相帮助学习,使我能够很好的掌握和运用专业知识,并在设计中得以体现。在以后的学习生活中我一定会能有更好的发挥,克服自己的缺点,不断改进,不断成长!

此次的python课程设计中，我通过与同学们的探讨和自己不懈的努力，设计并实现了人生中的第一个计算器，由于刚刚上完python的课程，此次的动手实践确实加深了我对python这门课程的理解，让我感受到了python那独有的魅力，这让我深陷其中，但同时我也发现了许多平常忽略的问题。比如平时看书上的一些知识感觉十分简单，但自己设计的时候却发现需要考虑许多细节，稍有疏忽就会产生许多棘手的问题。从而产生许许多多的错误，这更让我感到平时努力的重要性，对于第一次设计计算器的我来说刚开始的时候确实感觉到无从下手，很难把问题考虑全面，在学习这门课的过程中，在对python的了解过程中，慢慢对python有了感观。python这一词并不是很难想象，并不是像外人看来很神奇，那么高大上，作为计算机专业的学生，这样的专业术语或者专业知识是最基本的。学习的时候没有想象中的那么难，只要上课能听懂就基本还可以。但是问题还是出在书本有点厚，有的时候上课的内容都要找很久才能找到，甚至有的时候老师讲的知识书本上是找不到的，是另外补充而且是相当重要的内容。有的时候开小差，没有听到老师讲的知识点，这就导致了以后的学习无法顺利进行，使得学习起来十分困难。所以在数据库这门课的学习中，上课一定要听牢，就像老师说的那样，这样的专业课如果想凭考试前几天突击是行不通的，必须是日积月累的知识才能取得好成绩

课程设计的不足：

让我感到比较遗憾的一点是我们的计算器的大小是固定的，因为设计的过程中我已经对主界面的大小进行了限定，导致放大后计算器的界面大小也没有改变，我在设计之出本来是想使计算器的主界面的大小随放缩大小的改变而改变的，但我在设计之初发现如果这样设计，里面具体的部件设计就没法进行了，因为当界面的大小发生改变时，里面的具体部件的位置就会发生错位，整洁有序的界面会变得混乱不堪，因为时间的缘故，在经过和小组成员的商量之后，我们决定放弃这个选择，而改用具体大小的界面布局，这个是在设计时未解决的问题，也是让我非常遗憾的一方面。

**八、附录**

import tkinter #导入tkinter模块,Tkinter 是 Python 的标准 GUI 库。Python 使用 Tkinter 可以快速的创建 GUI 应用程序,图形化用户界面。

root=tkinter.Tk('red')#生成root主窗口

root.minsize(280,500)#设置窗口大小

root.title('余自立的计算器')#计算机的名字

#1.界面布局

#显示面板

result = tkinter.StringVar()#StringVar是Tk库内部定义的字符串变量类型，在这里用于管理部件上面的字符；不过一般用在按钮button上。改变StringVar，按钮上的文字也随之改变。

result.set(0) #显示面板显示结果1，用于显示默认数字0

result2 = tkinter.StringVar() #显示面板显示结果2，用于显示计算过程

result2.set('')

#显示板

label = tkinter.Label(root,font = ('黑体',20),bg = '#EEE9E9',bd ='9',fg = '#828282',anchor = 'se',textvariable = result2)# Anchor 标签中文本的位置 顶对齐'n',底对齐's',左'w',右'e'；background(bg)背景色； foreground(fg)前景色；borderwidth(bd)边框宽度；textvariable 可变文本，与StringVar等配合着用

label.place(width = 280,height = 170) #width: 组件的宽度；

label2 = tkinter.Label(root,font = ('黑体',30),bg = '#EEE9E9',bd ='9',fg = 'black',anchor = 'se',textvariable = result)

label2.place(y = 170,width = 280,height = 60)#y: 组件右上角的y坐标；

#数字键按钮

btn0 = tkinter.Button(root,text = '0',font = ('黑体',20),fg = ('#cc3300'),bd = 0.5,bg = ('#33ccff'),command = lambda : pressNum('0')) #command指定按钮消息的回调函数；

btn0.place(x = 70,y = 450,width = 70,height = 55)

btn1 = tkinter.Button(root,text = '1',font = ('黑体',20),fg = ('#cc3300'),bd = 0.5,bg = ('#33ccff'),command = lambda : pressNum('1'))

btn1.place(x = 0,y = 395,width = 70,height = 55)

btn2 = tkinter.Button(root,text = '2',font = ('黑体',20),fg = ('#cc3300'),bd = 0.5,bg = ('#33ccff'),command = lambda : pressNum('2'))

btn2.place(x = 70,y = 395,width = 70,height = 55)

btn3 = tkinter.Button(root,text = '3',font = ('黑体',20),fg = ('#cc3300'),bd = 0.5,bg = ('#33ccff'),command = lambda : pressNum('3'))

btn3.place(x = 140,y = 395,width = 70,height = 55)

btn4 = tkinter.Button(root,text = '4',font = ('黑体',20),fg = ('#cc3300'),bd = 0.5,bg = ('#33ccff'),command = lambda : pressNum('4'))

btn4.place(x = 0,y = 340,width = 70,height = 55)

btn5 = tkinter.Button(root,text = '5',font = ('黑体',20),fg = ('#cc3300'),bd = 0.5,bg = ('#33ccff'),command = lambda : pressNum('5'))

btn5.place(x = 70,y = 340,width = 70,height = 55)

btn6 = tkinter.Button(root,text = '6',font = ('黑体',20),fg = ('#cc3300'),bd = 0.5,bg = ('#33ccff'),command = lambda : pressNum('6'))

btn6.place(x = 140,y = 340,width = 70,height = 55)

btn7 = tkinter.Button(root,text = '7',font = ('黑体',20),fg = ('#cc3300'),bd = 0.5,bg = ('#33ccff'),command = lambda : pressNum('7'))

btn7.place(x = 0,y = 285,width = 70,height = 55)

btn8 = tkinter.Button(root,text = '8',font = ('黑体',20),fg = ('#cc3300'),bd = 0.5,bg = ('#33ccff'),command = lambda : pressNum('8'))

btn8.place(x = 70,y = 285,width = 70,height = 55)

btn9 = tkinter.Button(root,text = '9',font = ('黑体',20),fg = ('#cc3300'),bd = 0.5,bg = ('#33ccff'),command = lambda : pressNum('9'))

btn9.place(x = 140,y = 285,width = 70,height = 55)

#运算符号按钮

btnac = tkinter.Button(root,text = 'AC',bd = 0.5,font = ('黑体',20),fg = 'orange',bg = ('#336699'),command = lambda :pressCompute('AC'))

btnac.place(x = 0,y = 230,width = 70,height = 55)

btnback = tkinter.Button(root,text = '←',font = ('黑体',20),fg = '#cc3300',bg = ('#336699'),bd = 0.5,command = lambda:pressCompute('b'))

btnback.place(x = 70,y = 230,width = 70,height = 55)

btndivi = tkinter.Button(root,text = '÷',font = ('黑体',20),fg = '#cc3300',bg = ('#336699'),bd = 0.5,command = lambda:pressCompute('/'))

btndivi.place(x = 140,y = 230,width = 70,height = 55)

btnmul = tkinter.Button(root,text ='×',font = ('黑体',20),fg = "#cc3300",bd = 0.5,bg = ('#336699'),command = lambda:pressCompute('\*'))

btnmul.place(x = 210,y = 230,width = 70,height = 55)

btnsub = tkinter.Button(root,text = '-',font = ('黑体',20),fg = ('#cc3300'),bd = 0.5,bg = ('#336699'),command = lambda:pressCompute('-'))

btnsub.place(x = 210,y = 285,width = 70,height = 55)

btnadd = tkinter.Button(root,text = '+',font = ('黑体',20),fg = ('#cc3300'),bd = 0.5,bg = ('#336699'),command = lambda:pressCompute('+'))

btnadd.place(x = 210,y = 340,width = 70,height = 55)

btnequ = tkinter.Button(root,text = '=',bg = ('#336699'),font = ('黑体',20),fg = ('#cc3300'),bd = 0.5,command = lambda :pressEqual())

btnequ.place(x = 210,y = 395,width = 70,height = 110)

btnper = tkinter.Button(root,text = '%',font = ('黑体',20),fg = ('#cc3300'),bd = 0.5,bg = ('#33ccff'),command = lambda:pressCompute('%'))

btnper.place(x = 0,y = 450,width = 70,height = 55)

btnpoint = tkinter.Button(root,text = '.',font = ('黑体',20),fg = ('#cc3300'),bd = 0.5,bg = ('#33ccff'),command = lambda:pressCompute('.'))

btnpoint.place(x = 140,y = 450,width = 70,height = 55)

#操作函数

lists = [] #设置一个变量 保存运算数字和符号的列表

isPressSign = False #添加一个判断是否按下运算符号的标志,假设默认没有按下按钮

isPressNum = False

#数字函数

def pressNum(num): #设置一个数字函数 判断是否按下数字 并获取数字将数字写在显示版上

global lists #全局化lists和按钮状态isPressSign

global isPressSign

if isPressSign == False:

pass

else: #重新将运算符号状态设置为否

result.set(0) #输入栏初始显示为0

isPressSign = False

#判断界面的数字是否为0

oldnum = result.get() #第一步

if oldnum =='0': #如果界面上数字为0 则获取按下的数字

result.set(num)

else: #如果界面上的而数字不是0 则链接上新按下的数字

newnum = oldnum + num

result.set(newnum) #将按下的数字写到面板中

#运算函数

def pressCompute(sign):

global lists

global isPressSign

num = result.get() #获取界面数字

lists.append(num) #保存界面获取的数字到列表中

result.set(sign) #将输入的运算符写在面板中

lists.append(sign) #将按下的运算符号保存到列表中

isPressSign = True

if sign =='AC':#如果按下的是'AC'按键，则清空列表内容，将屏幕上的数字键设置为默认数字0

lists.clear()

result.set(0)

if sign =='b': #如果按下的是退格‘’，则选取当前数字第一位到倒数第二位

a = num[0:-1]

lists.clear()

result.set(a)

#获取运算结果函数

def pressEqual():

global lists

global isPressSign

curnum = result.get() #设置当前数字变量，并获取添加到列表

lists.append(curnum)

computrStr = ''.join(lists) #讲列表内容用join命令将字符串链接起来

endNum = eval(computrStr) #用eval命令运算字符串中的内容

result.set(endNum) #讲运算结果显示到屏幕1

result2.set(computrStr) #将运算过程显示到屏幕2

lists.clear() #清空列表内容

root.mainloop()#进入消息循环（必需组件）