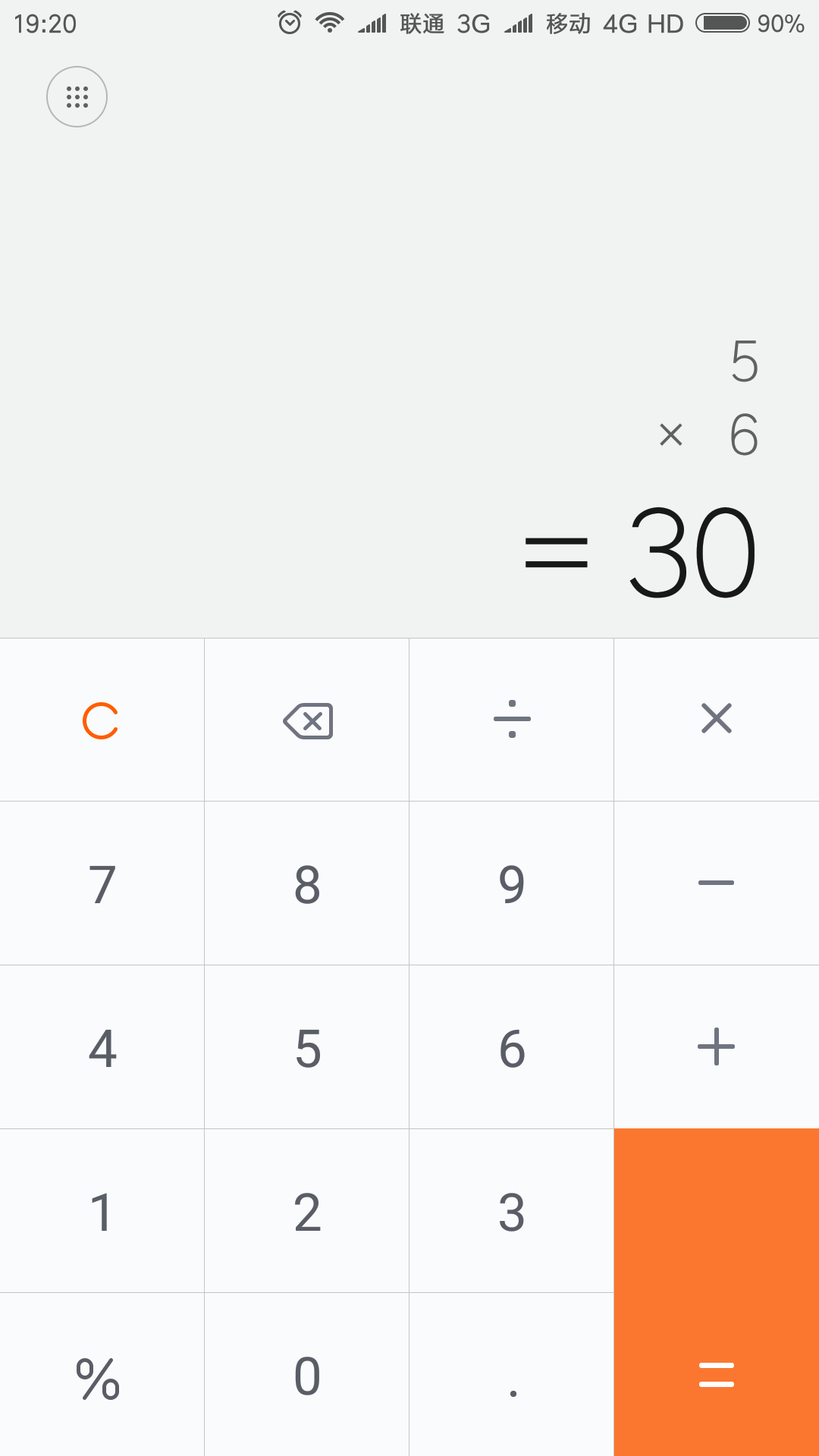
小朋友学Python（27）：实现带UI的简易计算器

#一、需求

实现一个简易计算器，用户界面类似小米手机的计算器。



#二、编程环境

语言：Python 3

操作系统：Win/Mac

#三、实现过程

##（一）界面构成

首先，咱们看一下上面整个页面的构成：

有两个Label，上面那个height较大的Label用来显示计算过程，下面那个height较小的Label用来显示计算结果。

再下面是19个Button，排列成五行四列（等号占了两行）。

##（二）数值定义

界面的宽度和高度通常用像素来表示，像素英文为pixel，简写为px。

按纽的宽和高设为70px \* 60px，则整个界面的宽度为70px \* 4 = 280px。

显示计算过程的Label高设为120px，显示计算结果的Label高设为60px，则整个界面的高度为120 + 60 + 60 \* 5 = 480px。

这些数值以及字体、字号、按钮颜色、按纽边框粗细，都可以定义为全局变量:

```

w = 280 #整个界面的宽度

h = 480 #整个界面的高度

processH = 120 #存放计算过程的Label的高度

resultH = 60 #存放计算结果的Label的高度

btnWidth = 70 #按纽宽度

btnHeight = 60 #按纽高度

msfont = '微软雅黑' #微软雅黑字体

fontSize = 20 #字体大小

btnFgColor = '#4F4F4F' #按纽颜色

btnBorderWidth = 0.5 #按纽边框粗细

```

##（三）编写整个界面和两个Label

```

import tkinter #导入tkinter模块

#定义数值

w = 280 #整个界面的宽度

h = 480 #整个界面的高度

processH = 120 #存放计算过程的Label的高度

resultH = 60 #存放计算结果的Label的高度

btnWidth = 70 #按纽宽度

btnHeight = 60 #按纽高度

msfont = '微软雅黑' #微软雅黑字体

fontSize = 20 #字体大小

btnFgColor = '#4F4F4F' #按纽颜色

btnBorderWidth = 0.5 #按纽边框粗细

#面板设置

root = tkinter.Tk()

root.minsize(w,h) #面板大小的最小值

root.title('海天一树的计算器') #面板的标题

process = tkinter.StringVar() #字符串用于存放计算过程，初始状态为空

process.set('')

result = tkinter.StringVar() #字符串用于存放计算结果，初始状态为0

result.set(0)

#labelProcess用于显示计算过程，labelResult用于显示计算结果

labelProcess = tkinter.Label(root,font = (msfont,fontSize),bg = '#ff0000',bd ='9',fg = '#828282',anchor = 'se',textvariable = process)

labelProcess.place(width = w,height = processH)

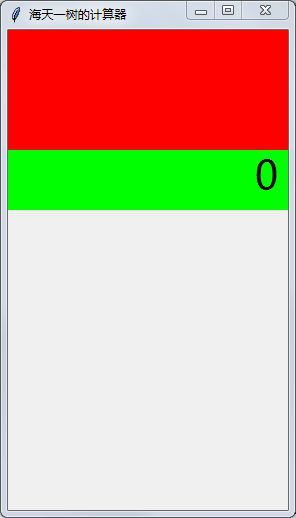
labelResult = tkinter.Label(root,font = (msfont,30),bg = '#00ff00',bd ='9',fg = 'black',anchor = 'se',textvariable = result)

labelResult.place(y = processH,width = w,height = resultH)

root.mainloop()

```

运行结果：



说明：

（1）tkinter.Label的数据结构为：

|  |  |
| --- | --- |
| 属性名称 | 属性含义 |
| anchor | 指示文字会被放在控件的什么位置, 可选项有 N, NE, E, SE, S, SW, W, NW, CENTER. 默认为 CENTER. |
| aspect | 控件的宽高比, 即 width/height , 以百分比形式表示. 默认为 150, 即 Message 控件宽度比其高度大 50%. 注意: 如果显式的指定了控件宽度, 则该属性将被忽略. |
| background | 控件的背景色. 默认值为当前窗口系统的默认背景色. |
| bg | 同 background. |
| borderwidth | 边框宽度. 默认为2. |
| bd | 同 borderwidth. |
| cursor | 定义鼠标移动到 Message 上时的光标样式. 默认为系统标准样式. |
| font | 文字字体. 默认为当前系统默认. |
| foreground | 文字颜色. 默认为当前系统默认. |
| fg | 同 foreground |
| highlightbackground | 和 highlightcolor, highlightthickness 这两个属性一起指定了控件如何绘制高亮区域. |
| highlightcolor | 参考 highlightbackground. |
| highlightthickness | 参考 highlightbackground. |
| justify | 指示多行文本如何对齐. 可选项为 LEFT, RIGHT, CENTER. 默认为 LEFT. 注意: 该属性指示文字对齐方式. 如果要指定文字在控件内的位置, 请使用 anchor 属性. |
| padx | 水平方向的边距. 默认为 -1, 即无边距. |
| pady | 垂直方向的边距. 默认为 -1, 即无边距. |
| relief | 边框样式. 默认为 FLAT. 其他可选项为 SUNKEN, RAISED, GROOVE, RIDGE. |
| takefocus | 如果设置为 True, 控件将可以获取焦点. 默认为 False. |
| text | 文字内容. 控件将根据宽度自动对文字进行断行. |
| textvariable | 关联一个 Tkinter variable 对象, 通常为 StringVar 对象. 控件文本将在该对象改变时跟着改变. |
| width | 控件宽度, 单位为 charater units (不是像素). 如果未指定该选项, 将根据 aspect 属性自动设置宽度. |

（2）颜色的RGB表示法

R--Red

G--Green

B--Blue

R, G, B的取值范围为0x00~0xff，对应于十进制的0~255

下表列举了几个常见的RGB值所对应的颜色

|  |  |
| --- | --- |
| RGB值 | 颜色 |
| #ff0000 | 红色 |
| #00ff00 | 绿色 |
| #0000ff | 蓝色 |
| #ffff00 | 黄色 |
| #ff00ff | 粉色 |
| #00ffff | 浅蓝色 |
| #000000 | 黑色 |
| #ffffff | 白色 |
| #555555 | 深灰色 |
| #cccccc | 浅灰色 |

##（四）添加数字按纽

```

#数字按键

btn7 = tkinter.Button(root,text = '7',font = (msfont,fontSize),fg = (btnFgColor),bd = btnBorderWidth,command = lambda : clickNum('7'))

btn7.place(x = 0,y = processH + resultH + btnHeight,width = btnWidth,height = btnHeight)

btn8 = tkinter.Button(root,text = '8',font = (msfont,fontSize),fg = (btnFgColor),bd = btnBorderWidth,command = lambda : clickNum('8'))

btn8.place(x = btnWidth,y = processH + resultH + btnHeight,width = btnWidth,height = btnHeight)

btn9 = tkinter.Button(root,text = '9',font = (msfont,fontSize),fg = (btnFgColor),bd = btnBorderWidth,command = lambda : clickNum('9'))

btn9.place(x = 2 \* btnWidth,y = processH + resultH + btnHeight,width = btnWidth,height = btnHeight)

btn4 = tkinter.Button(root,text = '4',font = (msfont,fontSize),fg = (btnFgColor),bd = btnBorderWidth,command = lambda : clickNum('4'))

btn4.place(x = 0,y = processH + resultH + 2 \* btnHeight,width = btnWidth,height = btnHeight)

btn5 = tkinter.Button(root,text = '5',font = (msfont,fontSize),fg = (btnFgColor),bd = btnBorderWidth,command = lambda : clickNum('5'))

btn5.place(x = btnWidth,y = processH + resultH + 2 \* btnHeight,width = btnWidth,height = btnHeight)

btn6 = tkinter.Button(root,text = '6',font = (msfont,fontSize),fg = (btnFgColor),bd = btnBorderWidth,command = lambda : clickNum('6'))

btn6.place(x = 2 \* btnWidth,y = processH + resultH + 2 \* btnHeight,width = btnWidth,height = btnHeight)

btn1 = tkinter.Button(root,text = '1',font = (msfont,fontSize),fg = (btnFgColor),bd = btnBorderWidth,command = lambda : clickNum('1'))

btn1.place(x = 0,y = processH + resultH + 3 \* btnHeight,width = btnWidth,height = btnHeight)

btn2 = tkinter.Button(root,text = '2',font = (msfont,fontSize),fg = (btnFgColor),bd = btnBorderWidth,command = lambda : clickNum('2'))

btn2.place(x = btnWidth,y = processH + resultH + 3 \* btnHeight,width = btnWidth,height = btnHeight)

btn3 = tkinter.Button(root,text = '3',font = (msfont,fontSize),fg = (btnFgColor),bd = btnBorderWidth,command = lambda : clickNum('3'))

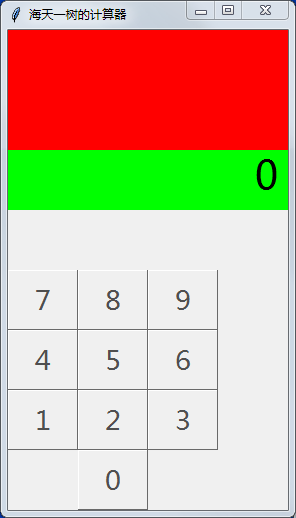
btn3.place(x = 2 \* btnWidth,y = processH + resultH + 3 \* btnHeight,width = btnWidth,height = btnHeight)

btn0 = tkinter.Button(root,text = '0',font = (msfont,fontSize),fg = (btnFgColor),bd = btnBorderWidth,command = lambda : clickNum('0'))

btn0.place(x = btnWidth,y = processH + resultH + 4 \* btnHeight,width = btnWidth,height = btnHeight)

```

运行结果：



说明：

Button按纽属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| text | 显示按纽内容 |
| command | 指定Button的事件处理函数 |
| compound | 指定文本与图像的位置关系 |
| bitmap | 指定位图 |
| focus\_set | 设置当前组件得到的焦点 |
| master | 代表了父窗口 |
| bg | 设置背景颜色 |
| fg | 设置前景颜色 |
| font | 设置字体大小 |
| height | 设置显示高度、如果未设置此项，其大小以适应内容标签 |
| relief | 指定外观装饰边界附近的标签,默认是平的,可以设置的参数有flat、groove、raised、ridge、solid、sunken |
| width | 设置显示宽度，如果未设置此项，其大小以适应内容标签 |
| wraplength | 将此选项设置为所需的数量限制每行的字符,数默认为0 |
| state | 设置组件状态;正常(normal),激活(active),禁用(disabled) |
| anchor | 设置Button文本在控件上的显示位置  可取值:n(north),s(south),w(west),e(east),和ne,nw,se,sw |
| bd | 设置Button的边框大小;bd(bordwidth)缺省为1或2个像素 |

##（五）添加符号按纽

```

btnac = tkinter.Button(root,text = 'AC',bd = btnBorderWidth,font = ('黑体',fontSize),fg = 'orange',command = lambda :clickSign('AC'))

btnac.place(x = 0,y = processH + resultH,width = btnWidth,height = btnHeight)

btnback = tkinter.Button(root,text = '←',font = (msfont,fontSize),fg = btnFgColor,bd = btnBorderWidth,command = lambda:clickSign('b'))

btnback.place(x = btnWidth,y = processH + resultH,width = btnWidth,height = btnHeight)

btndiv = tkinter.Button(root,text = '÷',font = (msfont,fontSize),fg = btnFgColor,bd = btnBorderWidth,command = lambda:clickSign('/'))

btndiv.place(x = 2 \* btnWidth,y = processH + resultH,width = btnWidth,height = btnHeight)

btnmul = tkinter.Button(root,text ='×',font = (msfont,fontSize),fg = btnFgColor,bd = btnBorderWidth,command = lambda:clickSign('\*'))

btnmul.place(x = 3 \* btnWidth,y = processH + resultH,width = btnWidth,height = btnHeight)

btnsub = tkinter.Button(root,text = '-',font = (msfont,fontSize),fg = (btnFgColor),bd = btnBorderWidth,command = lambda:clickSign('-'))

btnsub.place(x = 3 \* btnWidth,y = processH + resultH + btnHeight,width = btnWidth,height = btnHeight)

btnadd = tkinter.Button(root,text = '+',font = (msfont,fontSize),fg = (btnFgColor),bd = btnBorderWidth,command = lambda:clickSign('+'))

btnadd.place(x = 3 \* btnWidth,y = processH + resultH + 2 \* btnHeight,width = btnWidth,height = btnHeight)

btnequ = tkinter.Button(root,text = '=',bg = 'orange',font = (msfont,fontSize),fg = (btnFgColor),bd = btnBorderWidth,command = lambda :clickEqual())

btnequ.place(x = 3 \* btnWidth,y = processH + resultH + 3 \* btnHeight,width = btnWidth,height = 2 \* btnHeight)

btnper = tkinter.Button(root,text = '%',font = (msfont,fontSize),fg = (btnFgColor),bd = btnBorderWidth,command = lambda:clickSign('%'))

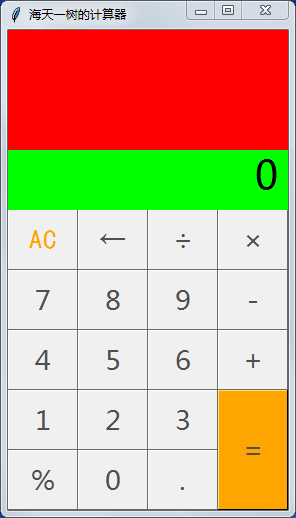
btnper.place(x = 0,y = processH + resultH + 4 \* btnHeight,width = btnWidth,height = btnHeight)

btnpoint = tkinter.Button(root,text = '.',font = (msfont,fontSize),fg = (btnFgColor),bd = btnBorderWidth,command = lambda:clickSign('.'))

btnpoint.place(x = 2 \* btnWidth,y = processH + resultH + 4 \* btnHeight,width = btnWidth,height = btnHeight)

```

运行结果：



到这一步，界面都已经绘制完了。下一步要开始做功能。

##（六）实现点击数字按钮的功能

```

def clickNum(num):

global lists #将lists变量设为全局变量

global isSignClicked #将isSignClicked变量设为全局变量

global isEqualClicked

if True == isSignClicked:

result.set(0) #如果上次按过运算符，那么这次要把原先的数字清空，否则会加在上次那个数的末尾

isSignClicked = False

if True == isEqualClicked:

result.set(0)

isEqualClicked = True

#判断界面的数字是否为0

oldnum = result.get() #第一步

if '0' == oldnum: #如果界面上数字为0 则获取按下的数字

result.set(num)

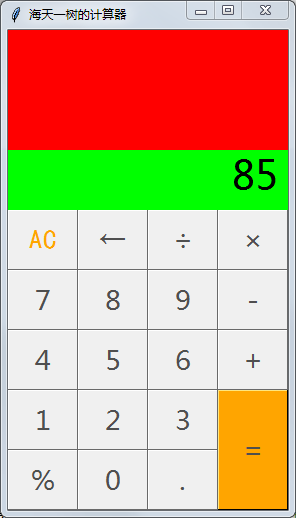
else: #如果界面上的数字不为0 则链接上新按下的数字

newnum = oldnum + num

result.set(newnum) #将按下的数字写到面板中

```

运行结果：



##（七）实现点击符号（等号除外）按纽的功能

```

#点击运算符号（等号除外）

def clickSign(sign):

global lists

global isSignClicked

num = result.get() #获取界面数字

if 0 == num.find('='): #在连续计算时，若得到的是“=15”，要去掉等号，取其数字

num = num[1:len(num)]

lists.append(num) #将界面获取的数字保存列表中

lists.append(sign) #将运算符号保存到列表中

isSignClicked = True

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

lists.clear()

process.set('')

result.set(0)

if 'b' == sign: #如果按下的是退格按纽，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)

if 0 == len(a): #如果是最后一个数，退格后变成0

result.set(0)

```

##（八）实现点击“=”的功能

```

#获取运算结果

def clickEqual():

global lists

global isEqualClicked

isEqualClicked = True

curval = result.get() #将界面上的数字添加到列表中

lists.append(curval)

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

res = eval(proc) #用eval命令运算字符串中的内容

res = '='+str(res) #给运算结果前添加一个 ‘=’ 显示

res = res[0:10] #所有的运算结果取9位数

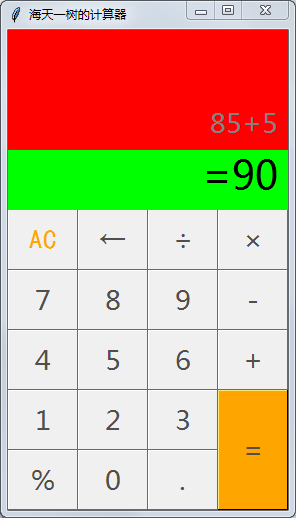
process.set(proc) #显示运算过程

result.set(res) #显示运算结果

lists.clear() #清空lists数组中的内容

```

运行结果：



##（九）调整两个Label的颜色

过程Label和结果Label的背景色很不好看，将两个Label的背景色都调为：

bg = ‘#EEE9E9’

运行效果：

