实例 2：电子计算器

从早期的算盘、算筹到如今的计算器，计算工具的不断升级使得人类的计算能力从速度与计算位数上都得到了质的提升。随着智能设备的发展，计算器从一个独立的机器成为了电子设备中的一个附加功能。一个常规的电子计算器如图 1 所示。

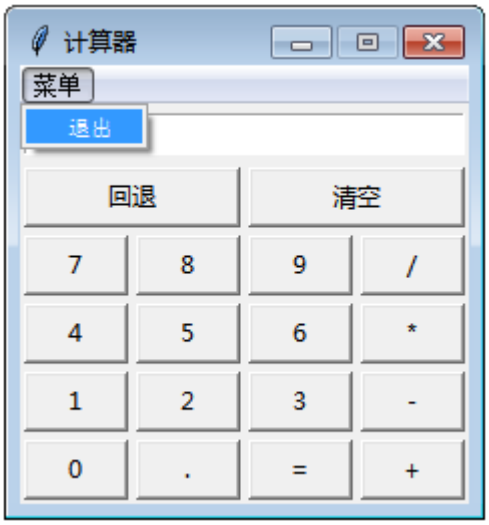

图1 电子计算器

图 1 所示的电子计算器不仅支持“+”、“-”、“*”、“/”运算，还支持“回退”、“清空”与“退出”功能。本实例要求使用 tkinter 实现如图 1 所示的电子计算器。

实例目标

- 掌握 tkinter 基础组件的使用
- 熟练地创建菜单组件

实例分析

电子计算器的界面主要包含以下元素：

- 根窗口：大小不可拉伸。
- 窗口标题：内容为计算器。
- 顶级菜单：只包含“菜单”选项。
- 下拉菜单：只包含“退出”选项。
- 文本框：可同时接收键盘输入和按钮输入。
- 回退与清空按钮：两者占据一整行，大小相同。
- 16 个按钮：所有的数字、小数点与运算符号。

由于回退和清空按钮的样式相同，其它按钮的样式相同，为了将相关组件使用框架统一

管理，因此这里可创建两个框架分组排布组件。

文本框中的数据是一个字符串，它里面只能包含小数、整数、运算符，且是不可变的，因此这里需要先将字符串转换成列表，再将列表转换为字符串进行返回。

代码实现

（1）由于实例中需创建两个框架，这里可将这两个框架共有的属性作为函数的默认值，以简化创建框架的过程。定义一个创建框架的函数 frame()，具体代码如下。

```
from tkinter import *

def frame(master):

    """将共同的属性作为默认值，以简化 Frame 创建过程"""

    w = Frame(master)

    w.pack(side=TOP, expand=YES, fill=BOTH)

    return w
```

（2）界面中也包含多个按钮组件，同样可以提取这些组件共有的属性作为函数的默认值，以简化创建按钮的过程。定义一个创建按钮的函数 btn()，具体代码如下。

```
def btn(master, text, command):

    """提取共同的属性作为默认值，使 Button 创建过程简化"""

    w = Button(master, text=text, command=command, width=6)

    w.pack(side=LEFT, expand=YES, fill=BOTH, padx=2, pady=2)

    return w
```

（3）定义一个根据文本框内容计算的函数 calc()，该函数中会尝试捕获处理文本框中的字符串：若 separator_flag 等于 0，则调用 del_separator()函数删除千位分隔符，否则调用 add_separator()函数添加千位分隔符。一旦出现 SyntaxError、ZeroDivisionError、NameError 中的任一种异常便会在文本框中返回“Error”，具体代码如下。

```
def calc(text_data):

    """用 eval 方法计算表达式字符串"""

    try:

        if (separator_flag.get() == 0):

            return eval(del_separator(text_data))

        else:

            return add_separator(str(eval(del_separator(text_data))))

    except (SyntaxError, ZeroDivisionError, NameError):
```

```
        return 'Error'
```

（4）定义一个删除文本框中末尾字符的函数 back()，该函数中需接收文本框中的字符串，返回删除末尾字符后的字符串，具体代码如下。

```
def back(text_data):
    """将 text_data 最末的字符删除并返回"""
    if len(text_data) > 0:
        return text_data[:-1]
    else:
        return text_data
```

（5）定义一个添加千位分隔符的函数 add_separator()，该函数中需接收字符串，先获取该字符串中小数点所在的位置，并依据不同的位置将小数点之前与之后的字符分隔为两部分 text_head 与 text_tail，然后为 text_head 插入千位分隔符，并将插入分隔符之后的 text_head 与 text_tail 合并转换成新的字符串返回，具体代码如下。

```
def add_separator(text_data):
    """向参数传入的数字串中添加千位分隔符
    这里考虑了三种情况：无整数部份，无小数部份，同时有整数和小数部份
    由于字符串是不可改变的，这里由字符串生成列表以便执行 insert 操作和
    extend 操作，操作完成后最由列表生成字符串返回
    """
    dot_index = text_data.find('.')
    if dot_index > 0:
        text_head = text[:dot_index]
        text_tail = text[dot_index:]
    elif dot_index < 0:
        text_head = text_data
        text_tail = ''
    else:
        text_head = ''
        text_tail = text_data
    list_ = [char for char in text_head]
    length = len(list_)
```

```
    tmp_index = 3
    while length - tmp_index > 0:
        list_.insert(length - tmp_index, ',')
        tmp_index += 3
    list_.extend(text_tail)
    new_text = ''
    for char in list_:
        new_text += char
    return new_text
```

（6）定义一个删除千位分隔符的函数 del_separator ()，具体代码如下。

```
def del_separator(text_data):
    """删除数字串中所有的千位分隔符"""
    return text_data.replace(',', '')
```

（7）创建根窗口，然后往根窗口中逐个添加界面元素，具体代码如下。

```
# 开始界面的实现
init_root = Tk()
init_root.resizable(width=False, height=False)
init_root.title("计算器")    # 添加标题
main_menus = Menu()          # 创建最上层主菜单
# 创建计算器菜单，并加入到主菜单
calc_menu = Menu(main_menus, tearoff=0)
calc_menu.add_command(label='退出', command=lambda: exit())
main_menus.add_cascade(label='菜单', menu=calc_menu)
text = StringVar()
separator_flag = IntVar()
separator_flag.set(0)
view_menu = Menu(main_menus, tearoff=0)
init_root['menu'] = main_menus  # 将主菜单与 root 绑定
# 创建文本框
Entry(init_root, textvariable=text).pack(expand=YES,
```

```
        fill=BOTH, padx=2, pady=4)
# 创建第一行三个按钮
first_line = frame(init_root)
btn(first_line, '回退', lambda t=text: t.set(back(t.get())))
btn(first_line, '清空', lambda t=text: t.set(''))
# 每行四个，创建其余四行按钮
for key in ('789/', '456*', '123-', '0.=+'):
    others = frame(init_root)
    for char in key:
        if char == '=':
            btn(others, char,
                lambda data=text: data.set(calc(data.get())))
        else:
            btn(others, char,
                lambda data=text, c=char: data.set(data.get() + c))
```

（8）通过根窗口调用 mainloop()函数监听事件循环，具体代码如下。

```
if __name__ == '__main__':
    init_root.mainloop()
```

代码测试

运行程序，在弹出的计算器的输入框中输入“11+22-33”，单击“=”按钮之前与之后的界面如图 2 所示。

图2 电子计算器-计算

在图 2 中的输入框中输入“10/0”， 单击“=”按钮之前与之后的界面如图 3 所示。

图3 电子计算器-错误