实验 7 GUI 程序设计

1、 实验目的

（1） 理解掌握 GUI 程序设计的基本思想和方法。

（2） 学会使用 Python tkinter 标准图形库。

（3） 学会使用 Matplotlib 进行简单的数据分析和数据可视化展示。

2、 实验任务

实验任务 7-1 柱形图绘制

要求：我们知道，展示、分析数据的时候，往往图表比表格更能体现数据的特点、走势。请查阅你所在的省（市）近四年的理科一本、文本、三本的分数线，用 Python 的 Matplotlib 图形库绘制柱形图，展示你查到的数据。如下图所示:

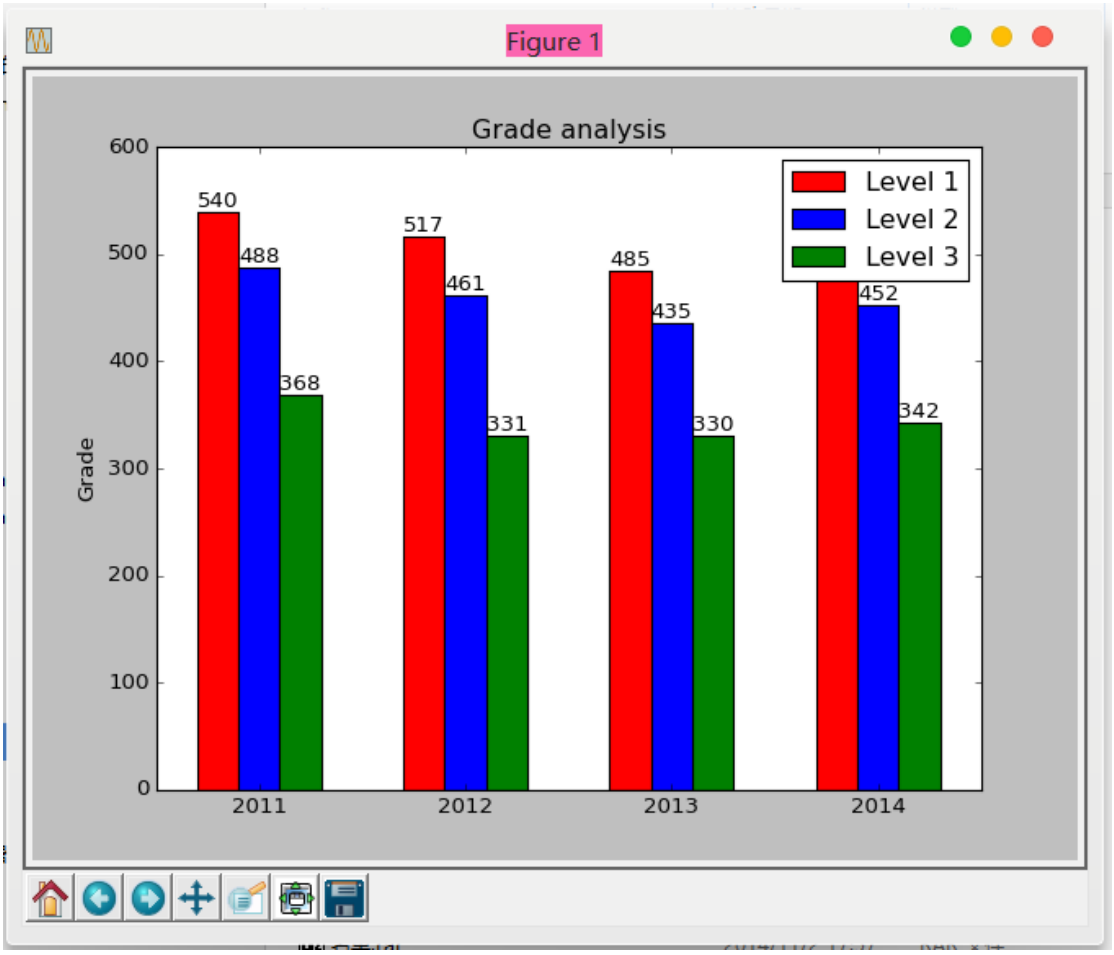

图 1. 陕西省近四年理科本科分数线统计图

实验目的：

掌握 matplotlib 的基本用法；

对数据分析和数据的可视化展示有一个较为直观的认识。

实验指导：更多关于 pyplot 的用法，请参考文档：

http://matplotlib.org/api/pyplot_summary.html

参考代码：

```
#! /usr/bin/env python3
# -*- encoding: utf8 -*-

# 7-2

import numpy as np
import matplotlib.pyplot as plt

def main():
    N = 4

    ind = np.arange(N)  # 获得 x 轴的坐标序列

    width = 0.2       # the width of the bars


    fig, ax = plt.subplots() # 得到图表

    rects = []
    grade = ((540, 517, 485, 503), (488, 461, 435, 452), (368, 331, 330,
342))


    # 分别构造单个序列的数据对应的图形，并设置颜色

    rects.append(ax.bar(ind + width * 1, grade[0], width, color =
'red'))
    ____________________________________________________________________
    ____________________________________________________________________


    ax.set_ylabel('Grade') # 设置坐标轴标题

    ax.set_title('Grade analysis')
    ax.set_xticks(ind + 2.5 * width)
    ax.set_xticklabels(('2011', '2012', '2013', '2014'))


    ax._____________________________________ # 绘制图形（柱状图和图例）
```

```
    def autolabel(rects):                    # 定义函数，在每一个序列的柱形
图上显示数值
        # attach some text labels
        for rect in rects:
            height = _____________________ # 获得每一列的高度值
            ax.text_____________________ # 显示数值，并设定其显示位置

    for i  in range(0, 3):
        _____________________________________ # 调用函数，给每一列显示数值

    plt.show()                               # 显示图形

if __name__ == '__main__':
    main()
```

实验任务 7-2 计算器

要求：请使用 Python 标准库中的 tkinter 模块编写一个简单的图形化计算器。并尝试通过点击按钮输入算式和运算符，点击“=”号按钮后，测试你编写的计算器是否能够计算出正确的结果。你可以直接使用给出的 calcu(equation) 函数来计算你输入的算式的值，你仅仅需要编写函数实验一个图形化界面即可。

关于 tkinter 包中的函数的详细用法，请查阅 Python documentation 中的相关的内容. https://docs.python.org/3/library/tk.html

实验目的：

学会图形程序设计的基本方法；

掌握 tkinter 标准图形库的基本用法。

实验指导：

1、 创建主窗口；

2、 向窗口上添加表达式输入框中(Text)；

3、 向窗口上添加数字和运算符输入按钮(Button)，当你点击这些按钮式，应该调用 input_char(experssionview, char)函数将对应的数字或运算符追加到表达式输入框中。

4、 当点击”=”按钮是，请调用给出的 calcu(experssion)函数进行计算，同时将得到的结构显示在表达式输入框的等号后边。

参考代码：

```
#! /usr/bin/env python3
# -*- encoding: utf-8 -*-

# 7-1

import tkinter
from tkinter import *
from tkinter import messagebox

def main():
    root = _____________________________ # 创建 GUI 主程序
    calculator = _____________________________ # 创建主窗口
    calculator.pack(fill=BOTH, expand = 1)

    addWidgets(calculator)                # 调用函数，向窗口上添加功能按钮，以输入数字和运算符
```

```
    root.title('Calculator')
    root.wm_resizable(width = False, height = False) # 禁用窗口缩放
    root.mainloop()

def addWidgets(frame):
    expression = Text(frame, height = 2, width = 28) # 创建表达式输入窗口
    number_1 = Button(frame, text='1', width = 5,
            command = lambda: input_char('1', expression))
    ____________________________________ # 创建数字按钮
    ____________________________________
    btn_add = Button(frame, text = '+', width = 5,
            command = lambda: input_char('+', expression))
    ____________________________________ # 创建运算符输入按钮

    # 使用 grid_configure 函数将组件添加到主窗口上
    expression.grid_configure(column = 1, row = 2, columnspan = 4, rowspan = 1)
    number_1.grid_configure(column = 1, row = 4, columnspan = 1, rowspan = 1)
    ____________________________________
    ____________________________________
    ____________________________________

def input_char(char, expressionview):
    expressionview.insert('1.end', char) # 输入按钮对应的字符

def calcu(expressionview):
    '''
    计算表达式的值，返回结果的字符串形式。
    如果表达式不合法，返回错误信息
    '''
    equal_str = expressionview.get('1.0', '1.end')
    expressionview.insert('1.end', '=')

    s = []
    i = 0
    equal_list = []
    while i < equal_str.__len__():
        if equal_str[i] <= '9' and equal_str[i] >= '0':
            tmp = 0
            while i < equal_str.__len__() and
                    equal_str[i] <= '9' and equal_str[i] >= '0':
                tmp = tmp * 10 + (ord(equal_str[i]) - ord('0'))
                i = i + 1
            equal_list.append(tmp)
            continue
```

```
        else:
            if equal_str[i] == '(':
                s.append(equal_str[i])
            elif equal_str[i] == ')':
                try:
                    while s[s.__len__()-1] != '(':
                        equal_list.append(s.pop())
                    s.pop()
                except IndexError:
                    return 'Invalid expersion!'
            elif equal_str[i] == '+' or equal_str[i] == '-':
                while s.__len__() > 0 and s[s.__len__()-1] != '(':
                    equal_list.append(s.pop())
                s.append(equal_str[i])
            elif equal_str[i] == '*' or equal_str[i] == '/':
                while s.__len__() > 0 and (s[s.__len__()-1] == '*'
                       or s[s.__len__() - 1] == '/'):
                    equal_list.append(s.pop())
                s.append(equal_str[i])
            i += 1

    while s.__len__() > 0:
        equal_list.append(s.pop())

    # print(equal_list) # print the postfix expression

    cnt_num = 0
    cnt_op = 0
    for item in equal_list:
        if type(item) == type(0): # integer.
            cnt_num += 1
        else:
            cnt_op += 1
    if cnt_op != cnt_num - 1:
        return 'Invalid expersion!'

    ans = []
    for i in range(0, equal_list.__len__()):
        if equal_list[i] == '+':
            ans.append(ans.pop(ans.__len__()-2) + ans.pop(ans.__len__()-1))
        elif equal_list[i] == '-':
            ans.append(ans.pop(ans.__len__()-2) - ans.pop(ans.__len__()-1))
        elif equal_list[i] == '*':
            ans.append(ans.pop(ans.__len__()-2) * ans.pop(ans.__len__()-1))
```

```
        elif equal_list[i] == '/':
            try:
                ans.append(ans.pop(ans.__len__()-2) / ans.pop(ans.__len__()-1))
            except ZeroDivisionError:
                return 'ZeroDivisionError: division by zero'
        else:
            ans.append(equal_list[i])

    return str(ans[0])

if __name__ == '__main__':
    main()
```