**实验7 GUI程序设计**

1. **实验目的**
2. 理解掌握GUI程序设计的基本思想和方法。
3. 学会使用Python tkinter标准图形库。
4. 学会使用Matplotlib 进行简单的数据分析和数据可视化展示。
5. **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:

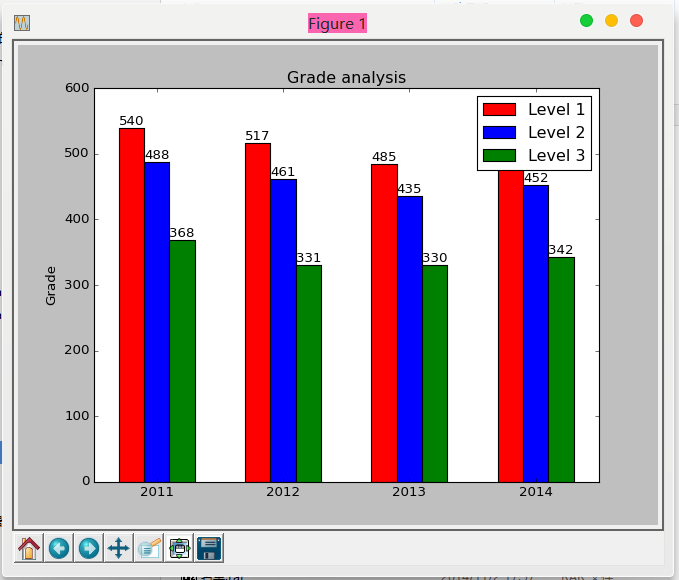


图1. 陕西省近四年理科本科分数线统计图

**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() |