

《程序设计（上机）》报告

（2021/2022 学年 第 一 学期）

题 目：**图书管理系统**

|  |  |
| --- | --- |
| **专 业** | **智能科学与技术** |
| **组长 学号姓名** | **B20040931谢俊哲** |
| **组员 学号姓名** | **B20040822罗梓淇** |
|  |  |
| **指 导 教 师** | **吴家皋** |
| **指 导 单 位** | **计算机学院、软件学院、网络空间安全学院** |
| **日 期** | **2021.12.03** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **成员分工** | | | | | | | | | |
| 组长（谢俊哲） | | 负责图书管理系统后端各部分代码的编写，实现修改图书书目，查找图书数据，对查找到的图书数据进行排序、分类、删除等要求，完成基础的个人图书馆与ISBN码的合法性与唯一性的提升要求。完成文档编写和绘画流程图等工作。 | | | | | | | |
| 组员（罗梓淇） | | 负责前端TK交互页面的编写，负责整合图书馆各部分功能代码逻辑与交互页面的整合，最终实现使用友好的交互页面来进行个人图书馆管理，完成流畅的图书馆管理体验。完成答辩PPT制作和文档编写工作。 | | | | | | | |
|  | |  | | | | | | | |
| **教师评价** | | | | | | | | | |
| 评价准则 | | | | 优秀 | 良好 | | 中等 | 合格 | 不合格 |
| 该课题组掌握程序设计开发的相关工程基础知识，并能够针对求解的工程问题，进行合理的分析与综合。 | | | |  |  | |  |  |  |
| 该课题组能够结合计算机软硬件资源，合理选用算法、数据结构、数据存储方式等技术手段，对求解的工程问题进行系统设计、建模及预测。 | | | |  |  | |  |  |  |
| 该课题组能够选择合适的程序设计语言与编程开发平台，对求解的工程问题进行系统模拟、仿真与预测。 | | | |  | √ | |  |  |  |
| 该课题组熟练掌握调试方法与工具，对程序开发过程中出现的问题进行分析、跟踪与调试，并在开发平台上对最终作品进行充分测试。 | | | |  |  | |  |  |  |
| 该课题组具备初步的开发能力，完成了课题要求的各项开发任务，功能实现全面。 | | | |  |  | |  |  |  |
| 该课题组具备初步的界面设计意识。用户使用界面布局合理，交互友好，操作简便。 | | | |  |  | |  |  |  |
| 该课题组正确、完整的回答了指导教师关于课题作品的提问，反映出其对课题内容，以及相关的工程基础知识具有较好的理解和掌握。 | | | |  |  | |  |  |  |
| 该课题组具备一定的表达能力与文字处理能力，能够熟练使用文字处理软件完成课题报告一份。报告完整记录了课题组的工作及总结，内容详实，格式规范。 | | | |  |  | |  |  |  |
| 该课题组具有较好的团队协作精神，能够努力沟通解决遇到的各种开发、工程问题。 | | | |  |  | |  |  |  |
| 该课题组具备一定自学能力与探索创新意识，能够充分利用各种网络与图书馆资源自学新知识与新技能。 | | | |  |  | |  |  |  |
| 该课题组在上机过程中遵守机房规章制度，出勤与平时表现规范。 | | | |  |  | |  |  |  |
| 最终作品和报告反映出该课题组具备较好的解决工程问题的能力。 | | | |  |  | |  |  |  |
| **本次程序设计能力达成评价** | | | | | | | | | |
| 学号姓名 | B20040931  谢俊哲 | | B20040822  罗梓淇 | | |  | | | |
| 贡献度及个人表现 | 50% | | 50% | | |  | | | |
| 最终成绩 |  | |  | | |  | | | |
| **备注：**  贡献度及个人表现：好、较好、中等、较差、差  最终成绩：优秀、良好、中等、及格、不及格 | | | | | | | | | |
| **指导教师： 年 月 日** | | | | | | | | | |

**图书管理系统**

**一、课题内容和要求**

该系统要求实现一个简单、实用的图书管理程序，主要功能包括图书数据的录入、查找、修改、删除、统计、排序等。所有图书数据都要利用文件系统保存，以备系统下次运行时使用。通过此课题，熟练掌握文件、数组、指针的各种操作，以及一些基本算法思想的应用。

**二、需求分析**

图书管理系统的功能框架图如图1所示。



图1 功能框架图

（1）提供可操作的图形化界面，用户可点击各按钮快速完成查找、修改图书、对图书排列等等各项图书操作。

（2）可根据用户提供的书籍的ISBN号、书名、作者等信息查找图书。

（3）可根据用户提供的书籍的ISBN号、书名、作者等信息修改与删除图书。

（4）可根据用户提供的ISBN号、书名、作者等信息对图书进行正、逆序排序。

（5）可显示所有的图书数据。

（6）可检验ISBN号的唯一性与合法性。

**三、概要设计**

1 主要存储结构

1. 用于存放在library.txt内提取的图书信息

number = [] #ISBN号

name = [] #书名

public\_address = [] #出版社信息

public\_time = [] #出版时间

price = [] #价格

author\_name = [] #作者名称

introduction = [] #介绍

lines\_str = [] # 所有书本信息

1. 用于修改图书信息的暂存信息字典

endtoend = {1: number, #ISBN号

2: name, #书名

3: public\_address, #出版社信息

4: public\_time, #出版时间

5: price, #价格

6: author\_name, #作者名名称

7: introduction #书本介绍

}

2 主要函数流程

（1）添加图书函数：该函数打开txt文件并向其中写入图书信息，用以保存图书信息。流程图如图1所示。



图1：添加图书函数

（2）全部图书信息显示函数：该函数遍历读取文件中所有文件信息并存入数组，遍历并打印数组中内容以显示图书信息。流程图如图二所示。



图2 显示全部图书信息

（3）图书信息查找函数：该函数遍历了所有图书信息并将其存入数组，并且根据用户提供的关键词信息锁定相关信息在数组中的索引；如果有重复，可将用户第一次查找到的信息存入新数组中，并在新数组的基础上继续提示用户继续输入关键词进行查找，最终筛选出唯一的图书信息。流程图如图（3）所示。



（3）图书信息查找函数

（4）图书信息修改函数：首先应用前面所述的图书查找程序找到唯一一本要修改图书所在处，然后输入要修改的图书的关键词，将其输入存放图书信息的列表中并覆盖原关键词信息所在位置；最后将其写入文件，修改图书信息成功。流程图如下所示。



（4）图书信息修改函数

（5）图书信息删除函数，首先应用前面所述的图书查找程序找到要删除的图书所在处，然后清空存放图书信息的数组中该图书所在位置处的信息。将清空后的图书信息数组写入文件。



（5）图书信息删除函数

(6)图书信息排序函数；将图书信息中每个类别的信息存为一个数组（ISBN号等等）与图书的整体信息存为一个数组；用户输入关键词类别；先对对应关键词类别的数组进行排序，然后按照排序后每个位置上对应的索引安排图书整体信息的排序（正序），用户输入判断是正序还是逆序，如果是逆序则将排序逆序。流程图如图（6）所示。



图（6） 图书排序函数

**四、源程序代码**

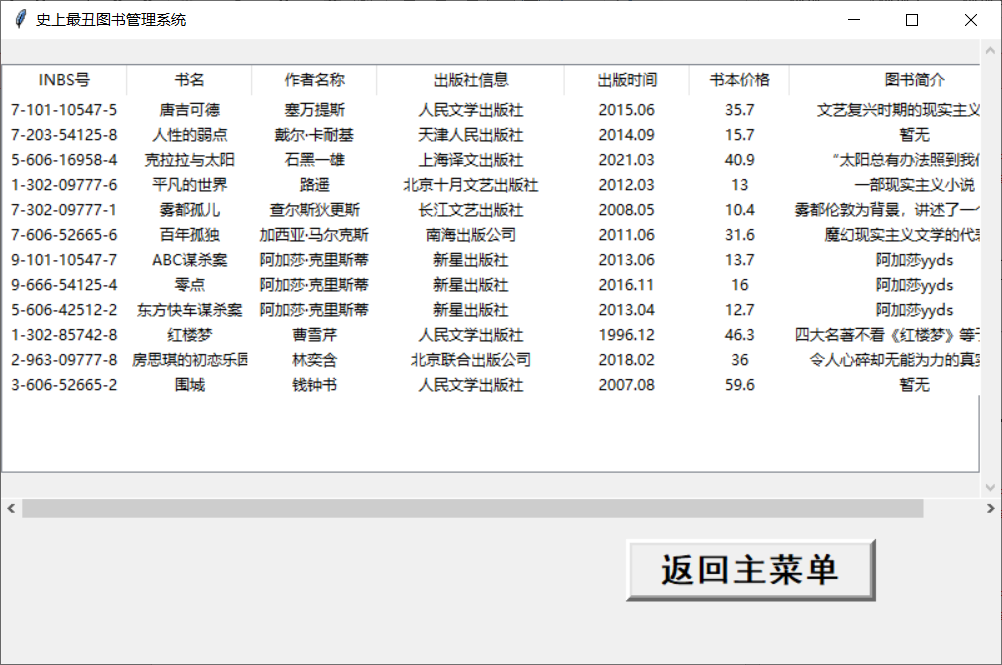
1. import tkinter as tk
2. from tkinter import ttk
3. import tkinter.messagebox
4. *# # 页面信息*
5. *# self.window = window*
6. *# self.window.title("图书管理系统")*
7. *# self.window.geometry("800x500")*
8. *#  宏定义*
9. *# 主页面*
10. but\_x = 290  *# 按钮横轴位置*
11. butwidth = 200  *# 按钮宽度*
12. butheight = 50  *# 按钮高度*
13. *# 增加界面*
14. labelstep = 40  *# 文本的垂直间隙*
15. entry\_x = 300  *# 输入框的横坐标*
16. *# 主页面删除控件函数*
17. def destroyall(self):
18. self.addbut.destroy()
19. self.changebut.destroy()
20. self.deletebut.destroy()
21. self.findbut.destroy()
22. self.showallbut.destroy()
23. self.quitbut.destroy()
24. self.labelname.destroy()
25. self.lzq.destroy()
26. self.xjz.destroy()
27. self.author.destroy()
28. *# 判断ISBN号合法性*
29. def cal(string):
30. if string[-2] != '-' or string[-8] != '-' or string[-12] != '-' or len("".join(string.split('-'))) != 10:
31. *# print('wrong')*
32. return 1
33. S = sum([i \* int(j) for i, j in zip(range(10, 1, -1), "".join(string.split('-'))[:-1])])
34. N = 11 - S % 11
35. if N == 10:
36. N = 'X'
37. if N == 11:
38. N = '0'
39. if "".join(string.split('-'))[-1] == str(N):
40. *# print('correct')*
41. return 0
42. else:
43. *# print('wrong')*
44. return 1
45. *# 主页面 OK*
46. class MainMenu:
47. def \_\_init\_\_(self, window):
48. *# 页面信息*
49. self.window = window
50. self.window.title("史上最丑图书管理系统")
51. self.window.geometry("800x500")
52. *# 文字标题*
53. self.labelname = tkinter.Label(window, text='图书管理系统', justify=tkinter.RIGHT, font=("华文行楷", 40), fg="blue",
54. width=100)
55. self.labelname.place(x=but\_x - 200, y=25, width=600, height=80)
56. *# 作者落款*
57. self.lzq = tkinter.Label(window, text='罗梓淇', justify=tkinter.RIGHT, font=("华文行楷", 15), fg="black", width=100)
58. self.lzq.place(x=but\_x - 250, y=400, width=100, height=80)
59. self.xjz = tkinter.Label(window, text='谢俊哲', justify=tkinter.RIGHT, font=("华文行楷", 15), fg="black", width=100)
60. self.xjz.place(x=but\_x - 250, y=350, width=100, height=80)
61. self.author = tkinter.Label(window, text='制作人：', justify=tkinter.RIGHT, font=("华文行楷", 15), fg="black", width=100)
62. self.author.place(x=but\_x - 250, y=300, width=100, height=80)
63. *# 增加图书*
64. self.addbut = tk.Button(self.window, text="增加图书", bd=5, font=('黑体', 20, 'bold'), command=self.toadd)
65. self.addbut.place(x=but\_x, y=100, width=butwidth, height=butheight)
66. *# 修改图书信息*
67. self.changebut = tk.Button(self.window, text="修改图书信息", bd=5, font=('黑体', 20, 'bold'), command=self.tochange)
68. self.changebut.place(x=but\_x, y=160, width=butwidth, height=butheight)
69. *# 删除图书信息*
70. self.deletebut = tk.Button(self.window, text="删除图书信息", bd=5, font=('黑体', 20, 'bold'), command=self.todelete)
71. self.deletebut.place(x=but\_x, y=220, width=butwidth, height=butheight)
72. *# 查找图书信息*
73. self.findbut = tk.Button(self.window, text="查找图书信息", bd=5, font=('黑体', 20, 'bold'), command=self.tofind)
74. self.findbut.place(x=but\_x, y=280, width=butwidth, height=butheight)
75. *# 显示图书信息*
76. self.showallbut = tk.Button(self.window, text="显示图书信息", bd=5, font=('黑体', 20, 'bold'), command=self.toshowall)
77. self.showallbut.place(x=but\_x, y=340, width=butwidth, height=butheight)
78. *# 退出按钮*
79. self.quitbut = tk.Button(self.window, text="退出", bd=5, font=('黑体', 20, 'bold'), command=self.quit)
80. self.quitbut.place(x=650, y=400, width=100, height=50)
81. def toadd(self):
82. destroyall(self)
83. AddMenu(root)
84. def tochange(self):
85. destroyall(self)
86. ChangeMenu(root)
87. def todelete(self):
88. destroyall(self)
89. DeleteMenu(root)
90. def tofind(self):
91. destroyall(self)
92. FindMenu(root)
93. def toshowall(self):
94. destroyall(self)
95. ShowAllMenu(root)
96. def quit(self):
97. root.destroy()
98. *# 增加图书页面 OK*
99. class AddMenu:  *# 增加图书页面*
100. def \_\_init\_\_(self, window):
101. self.window = window
102. self.window.title("史上最丑图书管理系统")
103. self.window.geometry("800x500")
104. *# self.window.config(bg="#0F375A")*
105. *# 返回主菜单按钮*
106. self.backbut = tk.Button(self.window, text="返回主菜单", bd=5, font=('黑体', 20, 'bold'), command=self.back)
107. self.backbut.place(x=550, y=400, width=200, height=50)
108. *# 添加图书按钮*
109. self.addbut = tk.Button(self.window, text="添加", bd=5, font=('黑体', 20, 'bold'), command=self.addbook)
110. self.addbut.place(x=300, y=80 + 7 \* labelstep, width=80, height=40)
111. *# 定义标签和输入框*
112. if 1:
113. *# 标签：INBS号，书名，作者名称，出版社信息，出版时间（格式为2020.03），书本价格，图书简介*
114. self.label1 = tk.Label(window, text='ISBN号:')
115. self.label1.place(x=150, y=80)
116. self.label2 = tk.Label(window, text='书名:')
117. self.label2.place(x=150, y=80 + 1 \* labelstep)
118. self.label3 = tk.Label(window, text='作者名称:')
119. self.label3.place(x=150, y=80 + 2 \* labelstep)
120. self.label4 = tk.Label(window, text='出版社信息:')
121. self.label4.place(x=150, y=80 + 3 \* labelstep)
122. self.label5 = tk.Label(window, text='出版时间（格式为2020.03）:')
123. self.label5.place(x=150, y=80 + 4 \* labelstep)
124. self.label6 = tk.Label(window, text='书本价格:')
125. self.label6.place(x=150, y=80 + 5 \* labelstep)
126. self.label7 = tk.Label(window, text='图书简介:')
127. self.label7.place(x=150, y=80 + 6 \* labelstep)
128. *# 输入框定义*
129. *# ISBN*
130. self.ISBN = tk.StringVar()
131. self.entry\_INBR = tk.Entry(window, textvariable=self.ISBN)
132. self.entry\_INBR.place(x=entry\_x, y=80)
133. *# 书名*
134. self.bookname = tk.StringVar()
135. self.entry\_bookname = tk.Entry(window, textvariable=self.bookname)
136. self.entry\_bookname.place(x=entry\_x, y=80 + 1 \* labelstep)
137. *# 作者名称*
138. self.author = tk.StringVar()
139. self.entry\_author = tk.Entry(window, textvariable=self.author)
140. self.entry\_author.place(x=entry\_x, y=80 + 2 \* labelstep)
141. *# 出版社信息*
142. self.press = tk.StringVar()
143. self.entry\_press = tk.Entry(window, textvariable=self.press)
144. self.entry\_press.place(x=entry\_x, y=80 + 3 \* labelstep)
145. *# 出版时间（格式为2020.03）*
146. self.pubtime = tk.StringVar()
147. self.entry\_pubtime = tk.Entry(window, textvariable=self.pubtime)
148. self.entry\_pubtime.place(x=entry\_x, y=80 + 4 \* labelstep)
149. *# 书本价格*
150. self.bookprice = tk.StringVar()
151. self.entry\_bookprice = tk.Entry(window, textvariable=self.bookprice)
152. self.entry\_bookprice.place(x=entry\_x, y=80 + 5 \* labelstep)
153. *# 图书简介*
154. self.bookintro = tk.StringVar()
155. self.entry\_bookintro = tk.Entry(window, textvariable=self.bookintro)
156. self.entry\_bookintro.place(x=entry\_x, y=80 + 6 \* labelstep)
157. def back(self):
158. *# 删除添加界面内的所有东西*
159. if 1:
160. self.backbut.destroy()
161. self.addbut.destroy()
162. self.label1.destroy()
163. self.label2.destroy()
164. self.label3.destroy()
165. self.label4.destroy()
166. self.label5.destroy()
167. self.label6.destroy()
168. self.label7.destroy()
169. self.entry\_INBR.destroy()
170. self.entry\_bookname.destroy()
171. self.entry\_author.destroy()
172. self.entry\_press.destroy()
173. self.entry\_pubtime.destroy()
174. self.entry\_bookprice.destroy()
175. self.entry\_bookintro.destroy()
176. MainMenu(root)
177. def addbook(self):
178. *# 输入框获取信息*
179. ISBN\_get = self.ISBN.get()
180. bookname\_get = self.bookname.get()
181. author\_get = self.author.get()
182. press\_get = self.press.get()
183. pubtime\_get = self.pubtime.get()
184. bookprice\_get = self.bookprice.get()
185. bookintro\_get = self.bookintro.get()
186. *# 判断图书信息是否完整*
187. if ISBN\_get == '' and bookname\_get == '' and press\_get == '' and author\_get == '' and pubtime\_get == '' and bookprice\_get == '' and bookintro\_get == '':
188. tk.messagebox.showinfo(title='消息提示框', message='请完善图书信息！')
189. elif cal(ISBN\_get):  *# 判断ISBN号合法性*
190. tk.messagebox.showinfo(title='消息提示框', message='ISBN号不合法！')
191. else:
192. ISBN\_onlyflag = 1  *# ISBN号唯一性，1为唯一， 0为有重复*
193. number = []
194. lines\_str = []
195. with open('library.txt', 'r') as f:
196. line = f.readline()
197. while line:
198. lines\_str.append(line.rstrip('\n'))
199. line = line.strip().split()
200. number.append(line[0])
201. line = f.readline()
202. f.close()
203. *# 判断ISBN号唯一性*
204. for i in number:
205. if i == ISBN\_get: ISBN\_onlyflag = 0
206. *# 往文件内写入信息*
207. if ISBN\_onlyflag == 1:
208. with open('library.txt', 'a') as f:
209. f.write(ISBN\_get + ' ')
210. f.write(bookname\_get + ' ')
211. f.write(author\_get + ' ')
212. f.write(press\_get + ' ')
213. f.write(pubtime\_get + ' ')
214. f.write(bookprice\_get + ' ')
215. f.write(bookintro\_get + '\n')
216. tk.messagebox.showinfo(title='消息提示框', message='添加成功！')
217. else:
218. tk.messagebox.showinfo(title='消息提示框', message='重复的ISBN号')
219. *# 修改图书页面*
220. class ChangeMenu:
221. def \_\_init\_\_(self, window):
222. *# 页面信息*
223. self.window = window
224. self.window.title("史上最丑图书管理系统")
225. self.window.geometry("800x500")
226. *# 定义容器 self.tabel\_frame*
227. self.tabel\_frame = tkinter.Frame(window)
228. self.tabel\_frame.pack()
229. *# 定义滚动条及其表格*
230. if 1:
231. xscroll = tk.Scrollbar(self.tabel\_frame, orient=tk.HORIZONTAL)
232. yscroll = tk.Scrollbar(self.tabel\_frame, orient=tk.VERTICAL)
233. columns = ['ISBN号', '书名', '作者名称', '出版社信息', '出版时间', '书本价格', '图书简介']
234. self.table = tk.ttk.Treeview(
235. master=self.tabel\_frame,  *# 父容器*
236. height=10,  *# 表格显示的行数,height行*
237. columns=columns,  *# 显示的列*
238. show='headings',  *# 隐藏首列*
239. xscrollcommand=xscroll.set,  *# x轴滚动条*
240. yscrollcommand=yscroll.set,  *# y轴滚动条*
241. )
242. *# 定义表头*
243. self.table.heading(column='ISBN号', text='INBS号', )  *# 点击表头出发lambda函数*
244. self.table.heading('书名', text='书名', )
245. self.table.heading('作者名称', text='作者名称', )
246. self.table.heading('出版社信息', text='出版社信息', )
247. self.table.heading('出版时间', text='出版时间', )
248. self.table.heading('书本价格', text='书本价格', )
249. self.table.heading('图书简介', text='图书简介', )  *# 点击表头出发lambda函数)*
250. *# 定义列*
251. self.table.column('ISBN号', width=100, minwidth=100, anchor='s', )
252. self.table.column('书名', width=100, minwidth=100, anchor='s')
253. self.table.column('作者名称', width=100, minwidth=100, anchor='s')
254. self.table.column('出版社信息', width=150, minwidth=150, anchor='s')
255. self.table.column('出版时间', width=100, minwidth=100, anchor='s')
256. self.table.column('书本价格', width=80, minwidth=80, anchor='s')
257. self.table.column('图书简介', width=200, minwidth=100, anchor='s')
258. *# config绑定滚动条为滚动事件 pack放置滚动条再最边边*
259. xscroll.config(command=self.table.xview)
260. xscroll.pack(side=tk.BOTTOM, fill=tk.X)
261. yscroll.config(command=self.table.yview)
262. yscroll.pack(side=tk.RIGHT, fill=tk.Y)
263. self.table.pack(pady=120)
264. *# 设计搜索方法和关键字搜索框*
265. if 1:
266. self.labelfindmethod = tkinter.Label(window, text='查找方式', justify=tkinter.RIGHT, width=50)
267. self.labelfindmethod.place(x=10, y=40, width=50, height=30)
268. self.labelenterkeywords = tkinter.Label(window, text='关键词', justify=tkinter.RIGHT, width=50)
269. self.labelenterkeywords.place(x=220, y=40, width=50, height=30)
270. *# 搜索方法元组*
271. finemethod = ('ISBN号', '书名', '作者名称', '出版社信息', '出版时间', '书本价格')
272. *# 查找方式组合框*
273. self.comboFM = tkinter.ttk.Combobox(window, values=finemethod, width=50)
274. self.comboFM.place(x=70, y=40, width=100, height=30)  *# 选择框的位置*
275. *# 绑定事件处理函数*
276. self.comboFM.bind('<<ComboboxSelected>> ', self.comboChange)
277. *# 输入框定义--关键字*
278. self.keywords = tk.StringVar()
279. self.entry\_keywords = tk.Entry(window, textvariable=self.keywords)
280. self.entry\_keywords.place(x=280, y=40, width=200, height=30)
281. *# 下面是关于修改内容的控件定义*
282. *# 修改信息种类的label*
283. self.labelcitype = tkinter.Label(window, text='修改信息种类', justify=tkinter.RIGHT, width=50)
284. self.labelcitype.place(x=10, y=360, width=80, height=30)
285. *# 修改信息种类的元组*
286. finemethod = ('ISBN号', '书名', '作者名称', '出版社信息', '出版时间', '书本价格', '图书简介')
287. *# 修改信息种类组合框*
288. self.comboCItype = tkinter.ttk.Combobox(window, values=finemethod, width=50)
289. self.comboCItype.place(x=100, y=360, width=100, height=30)  *# 选择框的位置*
290. *# 绑定事件处理函数*
291. self.comboCItype.bind('<<ComboboxSelected>> ', self.comboChangeCItype)
292. *# 文本定义*
293. self.lebelenterchangeinfo = tkinter.Label(window, text='输入修改信息', justify=tkinter.RIGHT, width=50)
294. self.lebelenterchangeinfo.place(x=220, y=360, width=100, height=30)
295. *# 输入框定义--修改内容*
296. self.changeinfo = tk.StringVar()
297. self.entry\_changeinfo = tk.Entry(window, textvariable=self.changeinfo)
298. self.entry\_changeinfo.place(x=330, y=360, width=150, height=30)
299. *# 定义按钮*
300. if 1:
301. *# 关键词查找按键*
302. self.findbut = tk.Button(self.window, text="关键词查找", bd=5, font=('黑体', 15, 'bold'), command=self.find)
303. self.findbut.place(x=500, y=40, width=200, height=30)
304. *# 修改图书信息按键*
305. self.changebut = tk.Button(self.window, text="修改图书信息", bd=5, font=('黑体', 15, 'bold'), command=self.change)
306. self.changebut.place(x=500, y=350, width=150, height=50)
307. *# 返回主菜单*
308. self.backbut = tk.Button(self.window, text="返回主菜单", bd=5, font=('黑体', 20, 'bold'), command=self.back)
309. self.backbut.place(x=550, y=400, width=200, height=50)
310. *# 帮助按钮*
311. self.helpbut = tk.Button(self.window, text="帮助", bd=5, font=('黑体', 10, 'bold'), command=self.help)
312. self.helpbut.place(x=60, y=420, width=50, height=30)
313. def help(self):
314. tk.messagebox.showinfo(title='消息提示框', message='请查询出你所需的图书，并选择修改信息的种类后修改')
315. *# 组合框：事件处理函数--关键字种类*
316. def comboChange(self, \*args, \*\*kwargs):
317. self.method = self.comboFM.get()
318. *# 组合框：事件处理函数--修改信息种类*
319. def comboChangeCItype(self, \*args, \*\*kwargs):
320. self.changeinfotype = self.comboCItype.get()
321. *# print(self.changeinfotype)*
322. *# finemethod = ('ISBN号', '书名', '作者名称', '出版社信息', '出版时间', '书本价格', '图书简介', '删除')*
323. if self.changeinfotype == 'ISBN号':
324. self.changeinfotype\_number = 1
325. elif self.changeinfotype == '书名':
326. self.changeinfotype\_number = 2
327. elif self.changeinfotype == '作者名称':
328. self.changeinfotype\_number = 3
329. elif self.changeinfotype == '出版社信息':
330. self.changeinfotype\_number = 4
331. elif self.changeinfotype == '出版时间':
332. self.changeinfotype\_number = 5
333. elif self.changeinfotype == '书本价格':
334. self.changeinfotype\_number = 6
335. elif self.changeinfotype == '图书简介':
336. self.changeinfotype\_number = 7
337. *# print(self.changeinfotype\_number)*
338. def back(self):
339. self.labelcitype.destroy()
340. self.lebelenterchangeinfo.destroy()
341. self.helpbut.destroy()
342. self.labelenterkeywords.destroy()
343. self.comboCItype.destroy()
344. self.entry\_changeinfo.destroy()
345. self.changebut.destroy()
346. self.backbut.destroy()
347. self.findbut.destroy()
348. self.comboFM.destroy()
349. self.entry\_keywords.destroy()
350. self.labelfindmethod.destroy()
351. self.tabel\_frame.destroy()
352. MainMenu(root)
353. *# 删除treeview所有*
354. def delTreeview(self, \*args, \*\*kwargs):
355. x = self.table.get\_children()
356. for item in x:
357. self.table.delete(item)
358. *# 关键词查找函数*
359. def find(self):
360. *# self.keywords\_get = self.keywords.get()  # 获取输入框内部的关键字*
361. *# columns = ['ISBN号', '书名', '作者名称', '出版社信息', '出版时间', '书本价格', '图书简介']*
362. number = []
363. name = []
364. public\_address = []
365. public\_time = []
366. price = []
367. author\_name = []
368. lines\_str = []  *# 字符串形式， 直接输出结果*
369. *# self.finalfind\_index = None   # 将找出来的书在库存里的序号放入finalfind\_index*
370. result = []
371. i = 0
372. *# 打开文件提取图书信息*
373. with open('library.txt', 'r') as f:
374. line = f.readline()
375. while line:
376. *# print(line.rstrip())  # 后面跟 ',' 将忽略换行符*
377. lines\_str.append(line.rstrip('\n'))
378. line = line.strip().split()
379. number.append(line[0])
380. name.append(line[1])
381. public\_address.append(line[2])
382. public\_time.append(line[3])
383. price.append(line[4])
384. author\_name.append(line[5])
385. line = f.readline()
386. f.close()
387. line2 = [t for t in lines\_str]
388. a = self.keywords.get() *#获取关键词输入框内文字*
389. if a == '':
390. tk.messagebox.showinfo(title='消息提示框', message='请输入关键词！')
391. else:
392. *# 关键词查找种类的判定*
393. if self.method == 'ISBN号':
394. index\_gather = [e for e, m in enumerate(number) if a in m]  *# 为了规避index只能查找一个*
395. copy = []  *# 专门针对重复情况而设置的， 正常情况下copy不会动为0， 重复情况可由i == 1 引出*
396. if len(lines\_str) > 0:  *# 这一个判断用于删除treeview表格内之前的数据，然后再打印出新的图书信息覆盖*
397. self.delTreeview(self)
398. for index\_ in index\_gather:
399. if i == 0:  *# 表示未重复初始输入状态*
400. result.append(lines\_str[index\_])
401. self.table.insert('', tk.END, values=lines\_str[index\_].split())  *# 添加数据到末尾*
402. if lines\_str[index\_] in result and i == 1:  *# 主要用于重复后第二次判断用的，如果不在里面，则代表是其他重复情况*
403. copy.append(lines\_str[index\_])
404. if i == 0:
405. length = len(result)
406. *# print(length)*
407. if i == 1:
408. result = [co for co in copy]
409. if len(result) == 0:
410. *# print('请输入其他有效信息！该信息在内部查不到任何内容！')*
411. tk.messagebox.showinfo(title='消息提示框', message='请输入其他有效信息！该信息在内部查不到任何内容！')
412. *# continue*
413. elif len(result) >= 1:
414. length = len(result)
415. if length > 1:
416. *# k = input('查阅的图书不止一本！要想继续按照关键词查找图书吗？(y/n)')*
417. tk.messagebox.showinfo(title='消息提示框', message='查阅的图书不止一本！请重新选择查找方式并按下列筛选图书输入新的关键词')
418. *# if k == "Y" or k == 'y':*
419. *#     print('继续查询！')*
420. *#     print('请在下方选择追加输入的条件！')  # 然后这里就会挑出循环到达while(1)开头*
421. i = 1
422. *#     #continue*
423. *# if k == 'N' or k == 'n':*
424. *#     print('返回初始图书查找页面！')*
425. *#     lines\_str = line2  # 为了防止图书查找有重复后再次查找后想继续查找别的；不然只会在result的范围内跑*
426. if length == 1:
427. *# k = input('以上可能是您要的图书！请问还要继续查询吗(y/n)')*
428. tk.messagebox.showinfo(title='消息提示框', message='以上可能是您要的图书！')
429. *# if k == 'y' or k == "Y":*
430. *#     print('继续查找图书！')*
431. result = []
432. i = 0
433. self.finalfind\_index = index\_  *# 将找出来的书在库存里的序号放入finalfind\_index*
434. *#     #continue*
435. *# if k == 'n' or k == "N":*
436. *#     print('查找结束，即将退出！')*
437. *#     #break*
438. if length == 0:
439. *# k = input('没有找到您所要的书本！请更换关键词或按照其他方式查找！请问还要继续查询吗？(y/n)')*
440. tk.messagebox.showinfo(title='消息提示框', message='没有找到您所要的书本！请更换关键词或按照其他方式查找！')
441. *# if k == 'y' or k == "Y":*
442. *#     print('继续查找图书！')*
443. *#     #continue*
444. *# if k == 'n' or k == "N":*
445. *#     print('查找结束，即将退出！')*
446. *#     #break*
447. if self.method == '书名':
448. index\_gather = [e for e, m in enumerate(name) if a in m]  *# 为了规避index只能查找一个*
449. print(name)
450. copy = []  *# 专门针对重复情况而设置的， 正常情况下copy不会动为0， 重复情况可由i == 1 引出*
451. if len(lines\_str) > 0:  *# 这一个判断用于删除treeview表格内之前的数据，然后再打印出新的图书信息覆盖*
452. self.delTreeview(self)
453. for index\_ in index\_gather:
454. if i == 0:  *# 表示未重复初始输入状态*
455. result.append(lines\_str[index\_])
456. self.table.insert('', tk.END, values=lines\_str[index\_].split())  *# 添加数据到末尾*
457. if lines\_str[index\_] in result and i == 1:  *# 主要用于重复后第二次判断用的，如果不在里面，则代表是其他重复情况*
458. copy.append(lines\_str[index\_])
459. if i == 0:
460. length = len(result)
461. *# print(length)*
462. if i == 1:
463. result = [co for co in copy]
464. if len(result) == 0:
465. tk.messagebox.showinfo(title='消息提示框', message='请输入其他有效信息！该信息在内部查不到任何内容！')
466. elif len(result) >= 1:
467. length = len(result)
468. if length > 1:
469. tk.messagebox.showinfo(title='消息提示框', message='查阅的图书不止一本！请重新选择查找方式并按下列筛选图书输入新的关键词')
470. i = 1
471. if length == 1:
472. tk.messagebox.showinfo(title='消息提示框', message='以上可能是您要的图书！')
473. result = []
474. i = 0
475. self.finalfind\_index = index\_  *# 将找出来的书在库存里的序号放入finalfind\_index*
476. if length == 0:
477. tk.messagebox.showinfo(title='消息提示框', message='没有找到您所要的书本！请更换关键词或按照其他方式查找！')
478. if self.method == '作者名称':
479. index\_gather = [e for e, m in enumerate(public\_address) if a in m]  *# 为了规避index只能查找一个*
480. *# print(public\_address) #实际上是作者名称 public address*
481. copy = []  *# 专门针对重复情况而设置的， 正常情况下copy不会动为0， 重复情况可由i == 1 引出*
482. if len(lines\_str) > 0:  *# 这一个判断用于删除treeview表格内之前的数据，然后再打印出新的图书信息覆盖*
483. self.delTreeview(self)
484. for index\_ in index\_gather:
485. if i == 0:  *# 表示未重复初始输入状态*
486. result.append(lines\_str[index\_])
487. self.table.insert('', tk.END, values=lines\_str[index\_].split())  *# 添加数据到末尾*
488. if lines\_str[index\_] in result and i == 1:  *# 主要用于重复后第二次判断用的，如果不在里面，则代表是其他重复情况*
489. copy.append(lines\_str[index\_])
490. if i == 0:
491. length = len(result)
492. *# print(length)*
493. if i == 1:
494. result = [co for co in copy]
495. if len(result) == 0:
496. tk.messagebox.showinfo(title='消息提示框', message='请输入其他有效信息！该信息在内部查不到任何内容！')
497. elif len(result) >= 1:
498. length = len(result)
499. if length > 1:
500. tk.messagebox.showinfo(title='消息提示框', message='查阅的图书不止一本！请重新选择查找方式并按下列筛选图书输入新的关键词')
501. i = 1
502. if length == 1:
503. tk.messagebox.showinfo(title='消息提示框', message='以上可能是您要的图书！')
504. result = []
505. i = 0
506. self.finalfind\_index = index\_  *# 将找出来的书在库存里的序号放入finalfind\_index*
507. if length == 0:
508. tk.messagebox.showinfo(title='消息提示框', message='没有找到您所要的书本！请更换关键词或按照其他方式查找！')
509. if self.method == '出版社信息':
510. index\_gather = [e for e, m in enumerate(public\_time) if a in m]  *# 为了规避index只能查找一个*
511. *# print(public\_time)  #实际上是出版社信息 public time*
512. copy = []  *# 专门针对重复情况而设置的， 正常情况下copy不会动为0， 重复情况可由i == 1 引出*
513. if len(lines\_str) > 0:  *# 这一个判断用于删除treeview表格内之前的数据，然后再打印出新的图书信息覆盖*
514. self.delTreeview(self)
515. for index\_ in index\_gather:
516. if i == 0:  *# 表示未重复初始输入状态*
517. result.append(lines\_str[index\_])
518. self.table.insert('', tk.END, values=lines\_str[index\_].split())  *# 添加数据到末尾*
519. if lines\_str[index\_] in result and i == 1:  *# 主要用于重复后第二次判断用的，如果不在里面，则代表是其他重复情况*
520. copy.append(lines\_str[index\_])
521. if i == 0:
522. length = len(result)
523. *# print(length)*
524. if i == 1:
525. result = [co for co in copy]
526. if len(result) == 0:
527. tk.messagebox.showinfo(title='消息提示框', message='请输入其他有效信息！该信息在内部查不到任何内容！')
528. elif len(result) >= 1:
529. length = len(result)
530. if length > 1:
531. tk.messagebox.showinfo(title='消息提示框', message='查阅的图书不止一本！请重新选择查找方式并按下列筛选图书输入新的关键词')
532. i = 1
533. if length == 1:
534. tk.messagebox.showinfo(title='消息提示框', message='以上可能是您要的图书！')
535. result = []
536. i = 0
537. self.finalfind\_index = index\_  *# 将找出来的书在库存里的序号放入finalfind\_index*
538. if length == 0:
539. tk.messagebox.showinfo(title='消息提示框', message='没有找到您所要的书本！请更换关键词或按照其他方式查找！')
540. if self.method == '出版时间':
541. index\_gather = [e for e, m in enumerate(price) if a in m]  *# 为了规避index只能查找一个*
542. *# print(price) #实际上是出版时间 price*
543. copy = []  *# 专门针对重复情况而设置的， 正常情况下copy不会动为0， 重复情况可由i == 1 引出*
544. if len(lines\_str) > 0:  *# 这一个判断用于删除treeview表格内之前的数据，然后再打印出新的图书信息覆盖*
545. self.delTreeview(self)
546. for index\_ in index\_gather:
547. if i == 0:  *# 表示未重复初始输入状态*
548. result.append(lines\_str[index\_])
549. self.table.insert('', tk.END, values=lines\_str[index\_].split())  *# 添加数据到末尾*
550. if lines\_str[index\_] in result and i == 1:  *# 主要用于重复后第二次判断用的，如果不在里面，则代表是其他重复情况*
551. copy.append(lines\_str[index\_])
552. if i == 0:
553. length = len(result)
554. *# print(length)*
555. if i == 1:
556. result = [co for co in copy]
557. if len(result) == 0:
558. tk.messagebox.showinfo(title='消息提示框', message='请输入其他有效信息！该信息在内部查不到任何内容！')
559. elif len(result) >= 1:
560. length = len(result)
561. if length > 1:
562. tk.messagebox.showinfo(title='消息提示框', message='查阅的图书不止一本！请重新选择查找方式并按下列筛选图书输入新的关键词')
563. i = 1
564. if length == 1:
565. tk.messagebox.showinfo(title='消息提示框', message='以上可能是您要的图书！')
566. result = []
567. i = 0
568. self.finalfind\_index = index\_  *# 将找出来的书在库存里的序号放入finalfind\_index*
569. if length == 0:
570. tk.messagebox.showinfo(title='消息提示框', message='没有找到您所要的书本！请更换关键词或按照其他方式查找！')
571. if self.method == '书本价格':
572. index\_gather = [e for e, m in enumerate(author\_name) if a in m]  *# 为了规避index只能查找一个*
573. *# print(author\_name) #实际上对应的是书本价格author name*
574. copy = []  *# 专门针对重复情况而设置的， 正常情况下copy不会动为0， 重复情况可由i == 1 引出*
575. if len(lines\_str) > 0:  *# 这一个判断用于删除treeview表格内之前的数据，然后再打印出新的图书信息覆盖*
576. self.delTreeview(self)
577. for index\_ in index\_gather:
578. if i == 0:  *# 表示未重复初始输入状态*
579. result.append(lines\_str[index\_])
580. self.table.insert('', tk.END, values=lines\_str[index\_].split())  *# 添加数据到末尾*
581. if lines\_str[index\_] in result and i == 1:  *# 主要用于重复后第二次判断用的，如果不在里面，则代表是其他重复情况*
582. copy.append(lines\_str[index\_])
583. if i == 0:
584. length = len(result)
585. *# print(length)*
586. if i == 1:
587. result = [co for co in copy]
588. if len(result) == 0:
589. tk.messagebox.showinfo(title='消息提示框', message='请输入其他有效信息！该信息在内部查不到任何内容！')
590. elif len(result) >= 1:
591. length = len(result)
592. if length > 1:
593. tk.messagebox.showinfo(title='消息提示框', message='查阅的图书不止一本！请重新选择查找方式并按下列筛选图书输入新的关键词')
594. i = 1
595. if length == 1:
596. tk.messagebox.showinfo(title='消息提示框', message='以上可能是您要的图书！')
597. result = []
598. i = 0
599. self.finalfind\_index = index\_  *# 将找出来的书在库存里的序号放入finalfind\_index*
600. if length == 0:
601. tk.messagebox.showinfo(title='消息提示框', message='没有找到您所要的书本！请更换关键词或按照其他方式查找！')
602. *# 修改图书信息函数*
603. def change(self):
604. *#print(self.finalfind\_index)*
605. if self.finalfind\_index != None:
606. *#print(self.finalfind\_index)*
607. number = []
608. name = []
609. public\_address = []
610. public\_time = []
611. price = []
612. author\_name = []
613. introduction = []
614. lines\_str = []  *# 字符串形式， 直接输出结果*
615. with open('library.txt', 'r') as f:
616. line = f.readline()
617. while line:
618. *# print(line.rstrip())  # 后面跟 ',' 将忽略换行符*
619. lines\_str.append(line.rstrip('\n'))
620. line = line.strip().split()
621. number.append(line[0])
622. name.append(line[1])
623. public\_address.append(line[2])
624. public\_time.append(line[3])
625. price.append(line[4])
626. author\_name.append(line[5])
627. introduction.append(line[6])
628. line = f.readline()
629. f.close()
630. endtoend = {1: number,
631. 2: name,
632. 3: public\_address,
633. 4: public\_time,
634. 5: price,
635. 6: author\_name,
636. 7: introduction
637. }
638. toto = [1, 2, 3, 4, 5, 6, 7]
639. *# k1 = input(*
640. *#     '请输入你所想要修改的图书的信息(INBS--1/book\_name--2/public\_address--3/public\_time--4/prcie--5/author\_name--6/introduction--7):')*
641. k1 = self.changeinfotype\_number
642. for to in toto:
643. if k1 == to and k1 != 8:
644. *# edition\_ = input('请输入你所想要修改的内容:')*
645. edition\_ = self.changeinfo.get()
646. if edition\_ == '':
647. tk.messagebox.showinfo(title='消息提示框', message='请输入修改信息！')
648. else:
649. endtoend[to][self.finalfind\_index] = edition\_  *# 这里的index一定为1*
650. with open('library.txt', 'w') as f:
651. for content\_ in range(len(number)):
652. f.write(number[content\_] + ' ')
653. f.write(name[content\_] + ' ')
654. f.write(public\_address[content\_] + ' ')
655. f.write(public\_time[content\_] + ' ')
656. f.write(price[content\_] + ' ')
657. f.write(author\_name[content\_] + ' ')
658. f.write(introduction[content\_] + '\n')
659. tk.messagebox.showinfo(title='消息提示框', message='修改成功！')
660. *# if k1 == 8:*
661. *#     del lines\_str[self.finalfind\_index]*
662. *#     with open('library.txt', 'w') as f:*
663. *#         for content2 in lines\_str:*
664. *#             f.write(content2 + '\n')*
665. *#     tk.messagebox.showinfo(title='消息提示框', message='删除成功！')*
666. else:
667. tk.messagebox.showinfo(title='消息提示框', message='请继续查找，确定修改书目！')
668. *# 修改完成后*
669. self.finalfind\_index = None
670. *# 删除图书页面*
671. class DeleteMenu:
672. def \_\_init\_\_(self, window):
673. *# 页面信息*
674. self.window = window
675. self.window.title("史上最丑图书管理系统")
676. self.window.geometry("800x500")
677. *# 定义容器 self.tabel\_frame*
678. self.tabel\_frame = tkinter.Frame(window)
679. self.tabel\_frame.pack()
680. *# 定义滚动条及其表格*
681. if 1:
682. xscroll = tk.Scrollbar(self.tabel\_frame, orient=tk.HORIZONTAL)
683. yscroll = tk.Scrollbar(self.tabel\_frame, orient=tk.VERTICAL)
684. columns = ['ISBN号', '书名', '作者名称', '出版社信息', '出版时间', '书本价格', '图书简介']
685. self.table = tk.ttk.Treeview(
686. master=self.tabel\_frame,  *# 父容器*
687. height=10,  *# 表格显示的行数,height行*
688. columns=columns,  *# 显示的列*
689. show='headings',  *# 隐藏首列*
690. xscrollcommand=xscroll.set,  *# x轴滚动条*
691. yscrollcommand=yscroll.set,  *# y轴滚动条*
692. )
693. *# 定义表头*
694. self.table.heading(column='ISBN号', text='INBS号', )  *# 点击表头出发lambda函数*
695. self.table.heading('书名', text='书名', )
696. self.table.heading('作者名称', text='作者名称', )
697. self.table.heading('出版社信息', text='出版社信息', )
698. self.table.heading('出版时间', text='出版时间', )
699. self.table.heading('书本价格', text='书本价格', )
700. self.table.heading('图书简介', text='图书简介', )  *# 点击表头出发lambda函数)*
701. *# 定义列*
702. self.table.column('ISBN号', width=100, minwidth=100, anchor='s', )
703. self.table.column('书名', width=100, minwidth=100, anchor='s')
704. self.table.column('作者名称', width=100, minwidth=100, anchor='s')
705. self.table.column('出版社信息', width=150, minwidth=150, anchor='s')
706. self.table.column('出版时间', width=100, minwidth=100, anchor='s')
707. self.table.column('书本价格', width=80, minwidth=80, anchor='s')
708. self.table.column('图书简介', width=200, minwidth=100, anchor='s')
709. *# config绑定滚动条为滚动事件 pack放置滚动条再最边边*
710. xscroll.config(command=self.table.xview)
711. xscroll.pack(side=tk.BOTTOM, fill=tk.X)
712. yscroll.config(command=self.table.yview)
713. yscroll.pack(side=tk.RIGHT, fill=tk.Y)
714. self.table.pack(pady=120)
715. *# 设计搜索方法和关键字搜索框*
716. if 1:
717. self.labelfindmethod = tkinter.Label(window, text='查找方式', justify=tkinter.RIGHT, width=50)
718. self.labelfindmethod.place(x=10, y=40, width=50, height=30)
719. self.labelenterkeywords = tkinter.Label(window, text='关键词', justify=tkinter.RIGHT, width=50)
720. self.labelenterkeywords.place(x=220, y=40, width=50, height=30)
721. *# 搜索方法元组*
722. finemethod = ('ISBN号', '书名', '作者名称', '出版社信息', '出版时间', '书本价格')
723. *# 查找方式组合框*
724. self.comboFM = tkinter.ttk.Combobox(window, values=finemethod, width=50)
725. self.comboFM.place(x=70, y=40, width=100, height=30)  *# 选择框的位置*
726. *# 绑定事件处理函数*
727. self.comboFM.bind('<<ComboboxSelected>> ', self.comboChange)
728. *# 输入框定义--关键字*
729. self.keywords = tk.StringVar()
730. self.entry\_keywords = tk.Entry(window, textvariable=self.keywords)
731. self.entry\_keywords.place(x=280, y=40, width=200, height=30)
732. *# 定义按钮*
733. if 1:
734. *# 关键词查找按键*
735. self.findbut = tk.Button(self.window, text="关键词查找", bd=5, font=('黑体', 15, 'bold'), command=self.find)
736. self.findbut.place(x=500, y=40, width=200, height=30)
737. *# 修改图书信息按键*
738. self.deletebut = tk.Button(self.window, text="删除图书信息", bd=5, font=('黑体', 15, 'bold'), command=self.delete)
739. self.deletebut.place(x=350, y=350, width=150, height=50)
740. *# 返回主菜单*
741. self.backbut = tk.Button(self.window, text="返回主菜单", bd=5, font=('黑体', 20, 'bold'), command=self.back)
742. self.backbut.place(x=550, y=400, width=200, height=50)
743. *# 帮助按钮*
744. self.helpbut = tk.Button(self.window, text="帮助", bd=5, font=('黑体', 10, 'bold'), command=self.help)
745. self.helpbut.place(x=60, y=420, width=50, height=30)
746. def help(self):
747. tk.messagebox.showinfo(title='消息提示框', message='请查询出你所需的图书，点击删除按钮删除')
748. *# 组合框：事件处理函数--关键字种类*
749. def comboChange(self, \*args, \*\*kwargs):
750. self.method = self.comboFM.get()
751. def back(self):
752. self.helpbut.destroy()
753. self.labelenterkeywords.destroy()
754. self.deletebut.destroy()
755. self.backbut.destroy()
756. self.findbut.destroy()
757. self.comboFM.destroy()
758. self.entry\_keywords.destroy()
759. self.labelfindmethod.destroy()
760. self.tabel\_frame.destroy()
761. MainMenu(root)
762. *# 删除treeview所有*
763. def delTreeview(self, \*args, \*\*kwargs):
764. x = self.table.get\_children()
765. for item in x:
766. self.table.delete(item)
767. *# 关键词查找函数*
768. def find(self):
769. *# self.keywords\_get = self.keywords.get()  # 获取输入框内部的关键字*
770. *# columns = ['ISBN号', '书名', '作者名称', '出版社信息', '出版时间', '书本价格', '图书简介']*
771. number = []
772. name = []
773. public\_address = []
774. public\_time = []
775. price = []
776. author\_name = []
777. lines\_str = []  *# 字符串形式， 直接输出结果*
778. *# self.finalfind\_index = None   # 将找出来的书在库存里的序号放入finalfind\_index*
779. result = []
780. i = 0
781. *# 打开文件提取图书信息*
782. with open('library.txt', 'r') as f:
783. line = f.readline()
784. while line:
785. *# print(line.rstrip())  # 后面跟 ',' 将忽略换行符*
786. lines\_str.append(line.rstrip('\n'))
787. line = line.strip().split()
788. number.append(line[0])
789. name.append(line[1])
790. public\_address.append(line[2])
791. public\_time.append(line[3])
792. price.append(line[4])
793. author\_name.append(line[5])
794. line = f.readline()
795. f.close()
796. line2 = [t for t in lines\_str]
797. a = self.keywords.get() *#获取关键词输入框内文字*
798. if a == '':
799. tk.messagebox.showinfo(title='消息提示框', message='请输入关键词！')
800. else:
801. *# 关键词查找种类的判定*
802. if self.method == 'ISBN号':
803. index\_gather = [e for e, m in enumerate(number) if a in m]  *# 为了规避index只能查找一个*
804. copy = []  *# 专门针对重复情况而设置的， 正常情况下copy不会动为0， 重复情况可由i == 1 引出*
805. if len(lines\_str) > 0:  *# 这一个判断用于删除treeview表格内之前的数据，然后再打印出新的图书信息覆盖*
806. self.delTreeview(self)
807. for index\_ in index\_gather:
808. if i == 0:  *# 表示未重复初始输入状态*
809. result.append(lines\_str[index\_])
810. self.table.insert('', tk.END, values=lines\_str[index\_].split())  *# 添加数据到末尾*
811. if lines\_str[index\_] in result and i == 1:  *# 主要用于重复后第二次判断用的，如果不在里面，则代表是其他重复情况*
812. copy.append(lines\_str[index\_])
813. if i == 0:
814. length = len(result)
815. *# print(length)*
816. if i == 1:
817. result = [co for co in copy]
818. if len(result) == 0:
819. *# print('请输入其他有效信息！该信息在内部查不到任何内容！')*
820. tk.messagebox.showinfo(title='消息提示框', message='请输入其他有效信息！该信息在内部查不到任何内容！')
821. *# continue*
822. elif len(result) >= 1:
823. length = len(result)
824. if length > 1:
825. *# k = input('查阅的图书不止一本！要想继续按照关键词查找图书吗？(y/n)')*
826. tk.messagebox.showinfo(title='消息提示框', message='查阅的图书不止一本！请重新选择查找方式并按下列筛选图书输入新的关键词')
827. *# if k == "Y" or k == 'y':*
828. *#     print('继续查询！')*
829. *#     print('请在下方选择追加输入的条件！')  # 然后这里就会挑出循环到达while(1)开头*
830. i = 1
831. *#     #continue*
832. *# if k == 'N' or k == 'n':*
833. *#     print('返回初始图书查找页面！')*
834. *#     lines\_str = line2  # 为了防止图书查找有重复后再次查找后想继续查找别的；不然只会在result的范围内跑*
835. if length == 1:
836. *# k = input('以上可能是您要的图书！请问还要继续查询吗(y/n)')*
837. tk.messagebox.showinfo(title='消息提示框', message='以上可能是您要的图书！')
838. *# if k == 'y' or k == "Y":*
839. *#     print('继续查找图书！')*
840. result = []
841. i = 0
842. self.finalfind\_index = index\_  *# 将找出来的书在库存里的序号放入finalfind\_index*
843. *#     #continue*
844. *# if k == 'n' or k == "N":*
845. *#     print('查找结束，即将退出！')*
846. *#     #break*
847. if length == 0:
848. *# k = input('没有找到您所要的书本！请更换关键词或按照其他方式查找！请问还要继续查询吗？(y/n)')*
849. tk.messagebox.showinfo(title='消息提示框', message='没有找到您所要的书本！请更换关键词或按照其他方式查找！')
850. *# if k == 'y' or k == "Y":*
851. *#     print('继续查找图书！')*
852. *#     #continue*
853. *# if k == 'n' or k == "N":*
854. *#     print('查找结束，即将退出！')*
855. *#     #break*
856. if self.method == '书名':
857. index\_gather = [e for e, m in enumerate(name) if a in m]  *# 为了规避index只能查找一个*
858. print(name)
859. copy = []  *# 专门针对重复情况而设置的， 正常情况下copy不会动为0， 重复情况可由i == 1 引出*
860. if len(lines\_str) > 0:  *# 这一个判断用于删除treeview表格内之前的数据，然后再打印出新的图书信息覆盖*
861. self.delTreeview(self)
862. for index\_ in index\_gather:
863. if i == 0:  *# 表示未重复初始输入状态*
864. result.append(lines\_str[index\_])
865. self.table.insert('', tk.END, values=lines\_str[index\_].split())  *# 添加数据到末尾*
866. if lines\_str[index\_] in result and i == 1:  *# 主要用于重复后第二次判断用的，如果不在里面，则代表是其他重复情况*
867. copy.append(lines\_str[index\_])
868. if i == 0:
869. length = len(result)
870. *# print(length)*
871. if i == 1:
872. result = [co for co in copy]
873. if len(result) == 0:
874. tk.messagebox.showinfo(title='消息提示框', message='请输入其他有效信息！该信息在内部查不到任何内容！')
875. elif len(result) >= 1:
876. length = len(result)
877. if length > 1:
878. tk.messagebox.showinfo(title='消息提示框', message='查阅的图书不止一本！请重新选择查找方式并按下列筛选图书输入新的关键词')
879. i = 1
880. if length == 1:
881. tk.messagebox.showinfo(title='消息提示框', message='以上可能是您要的图书！')
882. result = []
883. i = 0
884. self.finalfind\_index = index\_  *# 将找出来的书在库存里的序号放入finalfind\_index*
885. if length == 0:
886. tk.messagebox.showinfo(title='消息提示框', message='没有找到您所要的书本！请更换关键词或按照其他方式查找！')
887. if self.method == '作者名称':
888. index\_gather = [e for e, m in enumerate(public\_address) if a in m]  *# 为了规避index只能查找一个*
889. *# print(public\_address) #实际上是作者名称 public address*
890. copy = []  *# 专门针对重复情况而设置的， 正常情况下copy不会动为0， 重复情况可由i == 1 引出*
891. if len(lines\_str) > 0:  *# 这一个判断用于删除treeview表格内之前的数据，然后再打印出新的图书信息覆盖*
892. self.delTreeview(self)
893. for index\_ in index\_gather:
894. if i == 0:  *# 表示未重复初始输入状态*
895. result.append(lines\_str[index\_])
896. self.table.insert('', tk.END, values=lines\_str[index\_].split())  *# 添加数据到末尾*
897. if lines\_str[index\_] in result and i == 1:  *# 主要用于重复后第二次判断用的，如果不在里面，则代表是其他重复情况*
898. copy.append(lines\_str[index\_])
899. if i == 0:
900. length = len(result)
901. *# print(length)*
902. if i == 1:
903. result = [co for co in copy]
904. if len(result) == 0:
905. tk.messagebox.showinfo(title='消息提示框', message='请输入其他有效信息！该信息在内部查不到任何内容！')
906. elif len(result) >= 1:
907. length = len(result)
908. if length > 1:
909. tk.messagebox.showinfo(title='消息提示框', message='查阅的图书不止一本！请重新选择查找方式并按下列筛选图书输入新的关键词')
910. i = 1
911. if length == 1:
912. tk.messagebox.showinfo(title='消息提示框', message='以上可能是您要的图书！')
913. result = []
914. i = 0
915. self.finalfind\_index = index\_  *# 将找出来的书在库存里的序号放入finalfind\_index*
916. if length == 0:
917. tk.messagebox.showinfo(title='消息提示框', message='没有找到您所要的书本！请更换关键词或按照其他方式查找！')
918. if self.method == '出版社信息':
919. index\_gather = [e for e, m in enumerate(public\_time) if a in m]  *# 为了规避index只能查找一个*
920. *# print(public\_time)  #实际上是出版社信息 public time*
921. copy = []  *# 专门针对重复情况而设置的， 正常情况下copy不会动为0， 重复情况可由i == 1 引出*
922. if len(lines\_str) > 0:  *# 这一个判断用于删除treeview表格内之前的数据，然后再打印出新的图书信息覆盖*
923. self.delTreeview(self)
924. for index\_ in index\_gather:
925. if i == 0:  *# 表示未重复初始输入状态*
926. result.append(lines\_str[index\_])
927. self.table.insert('', tk.END, values=lines\_str[index\_].split())  *# 添加数据到末尾*
928. if lines\_str[index\_] in result and i == 1:  *# 主要用于重复后第二次判断用的，如果不在里面，则代表是其他重复情况*
929. copy.append(lines\_str[index\_])
930. if i == 0:
931. length = len(result)
932. *# print(length)*
933. if i == 1:
934. result = [co for co in copy]
935. if len(result) == 0:
936. tk.messagebox.showinfo(title='消息提示框', message='请输入其他有效信息！该信息在内部查不到任何内容！')
937. elif len(result) >= 1:
938. length = len(result)
939. if length > 1:
940. tk.messagebox.showinfo(title='消息提示框', message='查阅的图书不止一本！请重新选择查找方式并按下列筛选图书输入新的关键词')
941. i = 1
942. if length == 1:
943. tk.messagebox.showinfo(title='消息提示框', message='以上可能是您要的图书！')
944. result = []
945. i = 0
946. self.finalfind\_index = index\_  *# 将找出来的书在库存里的序号放入finalfind\_index*
947. if length == 0:
948. tk.messagebox.showinfo(title='消息提示框', message='没有找到您所要的书本！请更换关键词或按照其他方式查找！')
949. if self.method == '出版时间':
950. index\_gather = [e for e, m in enumerate(price) if a in m]  *# 为了规避index只能查找一个*
951. *# print(price) #实际上是出版时间 price*
952. copy = []  *# 专门针对重复情况而设置的， 正常情况下copy不会动为0， 重复情况可由i == 1 引出*
953. if len(lines\_str) > 0:  *# 这一个判断用于删除treeview表格内之前的数据，然后再打印出新的图书信息覆盖*
954. self.delTreeview(self)
955. for index\_ in index\_gather:
956. if i == 0:  *# 表示未重复初始输入状态*
957. result.append(lines\_str[index\_])
958. self.table.insert('', tk.END, values=lines\_str[index\_].split())  *# 添加数据到末尾*
959. if lines\_str[index\_] in result and i == 1:  *# 主要用于重复后第二次判断用的，如果不在里面，则代表是其他重复情况*
960. copy.append(lines\_str[index\_])
961. if i == 0:
962. length = len(result)
963. *# print(length)*
964. if i == 1:
965. result = [co for co in copy]
966. if len(result) == 0:
967. tk.messagebox.showinfo(title='消息提示框', message='请输入其他有效信息！该信息在内部查不到任何内容！')
968. elif len(result) >= 1:
969. length = len(result)
970. if length > 1:
971. tk.messagebox.showinfo(title='消息提示框', message='查阅的图书不止一本！请重新选择查找方式并按下列筛选图书输入新的关键词')
972. i = 1
973. if length == 1:
974. tk.messagebox.showinfo(title='消息提示框', message='以上可能是您要的图书！')
975. result = []
976. i = 0
977. self.finalfind\_index = index\_  *# 将找出来的书在库存里的序号放入finalfind\_index*
978. if length == 0:
979. tk.messagebox.showinfo(title='消息提示框', message='没有找到您所要的书本！请更换关键词或按照其他方式查找！')
980. if self.method == '书本价格':
981. index\_gather = [e for e, m in enumerate(author\_name) if a in m]  *# 为了规避index只能查找一个*
982. *# print(author\_name) #实际上对应的是书本价格author name*
983. copy = []  *# 专门针对重复情况而设置的， 正常情况下copy不会动为0， 重复情况可由i == 1 引出*
984. if len(lines\_str) > 0:  *# 这一个判断用于删除treeview表格内之前的数据，然后再打印出新的图书信息覆盖*
985. self.delTreeview(self)
986. for index\_ in index\_gather:
987. if i == 0:  *# 表示未重复初始输入状态*
988. result.append(lines\_str[index\_])
989. self.table.insert('', tk.END, values=lines\_str[index\_].split())  *# 添加数据到末尾*
990. if lines\_str[index\_] in result and i == 1:  *# 主要用于重复后第二次判断用的，如果不在里面，则代表是其他重复情况*
991. copy.append(lines\_str[index\_])
992. if i == 0:
993. length = len(result)
994. *# print(length)*
995. if i == 1:
996. result = [co for co in copy]
997. if len(result) == 0:
998. tk.messagebox.showinfo(title='消息提示框', message='请输入其他有效信息！该信息在内部查不到任何内容！')
999. elif len(result) >= 1:
1000. length = len(result)
1001. if length > 1:
1002. tk.messagebox.showinfo(title='消息提示框', message='查阅的图书不止一本！请重新选择查找方式并按下列筛选图书输入新的关键词')
1003. i = 1
1004. if length == 1:
1005. tk.messagebox.showinfo(title='消息提示框', message='以上可能是您要的图书！')
1006. result = []
1007. i = 0
1008. self.finalfind\_index = index\_  *# 将找出来的书在库存里的序号放入finalfind\_index*
1009. if length == 0:
1010. tk.messagebox.showinfo(title='消息提示框', message='没有找到您所要的关键词！请更换关键词或按照其他方式查找！')
1011. *# 修改图书信息函数*
1012. def delete(self):
1013. *#print(self.finalfind\_index)*
1014. if self.finalfind\_index != None:
1015. *#print(self.finalfind\_index)*
1016. number = []
1017. name = []
1018. public\_address = []
1019. public\_time = []
1020. price = []
1021. author\_name = []
1022. introduction = []
1023. lines\_str = []  *# 字符串形式， 直接输出结果*
1024. with open('library.txt', 'r') as f:
1025. line = f.readline()
1026. while line:
1027. *# print(line.rstrip())  # 后面跟 ',' 将忽略换行符*
1028. lines\_str.append(line.rstrip('\n'))
1029. line = line.strip().split()
1030. number.append(line[0])
1031. name.append(line[1])
1032. public\_address.append(line[2])
1033. public\_time.append(line[3])
1034. price.append(line[4])
1035. author\_name.append(line[5])
1036. introduction.append(line[6])
1037. line = f.readline()
1038. f.close()
1039. del lines\_str[self.finalfind\_index]
1040. with open('library.txt', 'w') as f:
1041. for content2 in lines\_str:
1042. f.write(content2 + '\n')
1043. tk.messagebox.showinfo(title='消息提示框', message='删除成功！')
1044. *# 修改完成后*
1045. self.finalfind\_index = None
1046. *# 查找图书页面 OK*
1047. class FindMenu:
1048. def \_\_init\_\_(self, window):
1049. *# 页面信息*
1050. self.window = window
1051. self.window.title("史上最丑图书管理系统")
1052. self.window.geometry("800x500")
1053. *# 定义容器 self.tabel\_frame*
1054. self.tabel\_frame = tkinter.Frame(window)
1055. self.tabel\_frame.pack()
1056. *# 定义滚动条及其表格*
1057. if 1:
1058. xscroll = tk.Scrollbar(self.tabel\_frame, orient=tk.HORIZONTAL)
1059. yscroll = tk.Scrollbar(self.tabel\_frame, orient=tk.VERTICAL)
1060. columns = ['ISBN号', '书名', '作者名称', '出版社信息', '出版时间', '书本价格', '图书简介']
1061. self.table = tk.ttk.Treeview(
1062. master=self.tabel\_frame,  *# 父容器*
1063. height=10,  *# 表格显示的行数,height行*
1064. columns=columns,  *# 显示的列*
1065. show='headings',  *# 隐藏首列*
1066. xscrollcommand=xscroll.set,  *# x轴滚动条*
1067. yscrollcommand=yscroll.set,  *# y轴滚动条*
1068. )
1069. *# 定义表头*
1070. self.table.heading(column='ISBN号', text='INBS号', )  *# 点击表头出发lambda函数*
1071. self.table.heading('书名', text='书名', )
1072. self.table.heading('作者名称', text='作者名称', )
1073. self.table.heading('出版社信息', text='出版社信息', )
1074. self.table.heading('出版时间', text='出版时间', )
1075. self.table.heading('书本价格', text='书本价格', )
1076. self.table.heading('图书简介', text='图书简介', )  *# 点击表头出发lambda函数)*
1077. *# 定义列*
1078. self.table.column('ISBN号', width=100, minwidth=100, anchor='s', )
1079. self.table.column('书名', width=100, minwidth=100, anchor='s')
1080. self.table.column('作者名称', width=100, minwidth=100, anchor='s')
1081. self.table.column('出版社信息', width=150, minwidth=150, anchor='s')
1082. self.table.column('出版时间', width=100, minwidth=100, anchor='s')
1083. self.table.column('书本价格', width=80, minwidth=80, anchor='s')
1084. self.table.column('图书简介', width=200, minwidth=100, anchor='s')
1085. *# config绑定滚动条为滚动事件 pack放置滚动条再最边边*
1086. xscroll.config(command=self.table.xview)
1087. xscroll.pack(side=tk.BOTTOM, fill=tk.X)
1088. yscroll.config(command=self.table.yview)
1089. yscroll.pack(side=tk.RIGHT, fill=tk.Y)
1090. self.table.pack(pady=120)
1091. *# 设计搜索方法和关键字搜索框*
1092. if 1:
1093. self.labelfindmethod = tkinter.Label(window, text='查找方式', justify=tkinter.RIGHT, width=50)
1094. self.labelfindmethod.place(x=10, y=40, width=50, height=30)
1095. self.labelenterkeywords = tkinter.Label(window, text='关键词', justify=tkinter.RIGHT, width=50)
1096. self.labelenterkeywords.place(x=220, y=40, width=50, height=30)
1097. *# 搜索方法元组*
1098. finemethod = ('ISBN号', '书名', '作者名称', '出版社信息', '出版时间', '书本价格')
1099. *# 查找方式组合框*
1100. self.comboFM = tkinter.ttk.Combobox(window, values=finemethod, width=50)
1101. self.comboFM.place(x=70, y=40, width=100, height=30)  *# 选择框的位置*
1102. *# 绑定事件处理函数*
1103. self.comboFM.bind('<<ComboboxSelected>> ', self.comboChange)
1104. *# 输入框定义--关键字*
1105. self.keywords = tk.StringVar()
1106. self.entry\_keywords = tk.Entry(window, textvariable=self.keywords)
1107. self.entry\_keywords.place(x=280, y=40, width=200, height=30)
1108. *# 定义按钮*
1109. if 1:
1110. *# 关键词查找按键*
1111. self.findbut = tk.Button(self.window, text="关键词查找", bd=5, font=('黑体', 15, 'bold'), command=self.find)
1112. self.findbut.place(x=500, y=40, width=200, height=30)
1113. *# 返回主菜单*
1114. self.backbut = tk.Button(self.window, text="返回主菜单", bd=5, font=('黑体', 20, 'bold'), command=self.back)
1115. self.backbut.place(x=550, y=400, width=200, height=50)
1116. *# 帮助按钮*
1117. self.helpbut = tk.Button(self.window, text="帮助", bd=5, font=('黑体', 10, 'bold'), command=self.help)
1118. self.helpbut.place(x=60, y=420, width=50, height=30)
1119. def help(self):
1120. tk.messagebox.showinfo(title='消息提示框', message='请选择搜索的关键词种类，输入关键词查询图书')
1121. *# 组合框：事件处理函数--关键字种类*
1122. def comboChange(self, \*args, \*\*kwargs):
1123. self.method = self.comboFM.get()
1124. *# 组合框：事件处理函数--修改信息种类*
1125. def back(self):
1126. self.helpbut.destroy()
1127. self.labelenterkeywords.destroy()
1128. self.backbut.destroy()
1129. self.findbut.destroy()
1130. self.comboFM.destroy()
1131. self.entry\_keywords.destroy()
1132. self.labelfindmethod.destroy()
1133. self.tabel\_frame.destroy()
1134. MainMenu(root)
1135. *# 删除treeview所有*
1136. def delTreeview(self, \*args, \*\*kwargs):
1137. x = self.table.get\_children()
1138. for item in x:
1139. self.table.delete(item)
1140. *# 关键词查找函数*
1141. def find(self):
1142. *# self.keywords\_get = self.keywords.get()  # 获取输入框内部的关键字*
1143. *# columns = ['ISBN号', '书名', '作者名称', '出版社信息', '出版时间', '书本价格', '图书简介']*
1144. number = []
1145. name = []
1146. public\_address = []
1147. public\_time = []
1148. price = []
1149. author\_name = []
1150. lines\_str = []  *# 字符串形式， 直接输出结果*
1151. *# self.finalfind\_index = None   # 将找出来的书在库存里的序号放入finalfind\_index*
1152. result = []
1153. i = 0
1154. *# 打开文件提取图书信息*
1155. with open('library.txt', 'r') as f:
1156. line = f.readline()
1157. while line:
1158. *# print(line.rstrip())  # 后面跟 ',' 将忽略换行符*
1159. lines\_str.append(line.rstrip('\n'))
1160. line = line.strip().split()
1161. number.append(line[0])
1162. name.append(line[1])
1163. public\_address.append(line[2])
1164. public\_time.append(line[3])
1165. price.append(line[4])
1166. author\_name.append(line[5])
1167. line = f.readline()
1168. f.close()
1169. line2 = [t for t in lines\_str]
1170. a = self.keywords.get()  *# 获取关键词输入框内文字*
1171. if a == '':
1172. tk.messagebox.showinfo(title='消息提示框', message='请输入关键词！')
1173. else:
1174. *# 关键词查找种类的判定*
1175. if self.method == 'ISBN号':
1176. index\_gather = [e for e, m in enumerate(number) if a in m]  *# 为了规避index只能查找一个*
1177. copy = []  *# 专门针对重复情况而设置的， 正常情况下copy不会动为0， 重复情况可由i == 1 引出*
1178. if len(lines\_str) > 0:  *# 这一个判断用于删除treeview表格内之前的数据，然后再打印出新的图书信息覆盖*
1179. self.delTreeview(self)
1180. for index\_ in index\_gather:
1181. if i == 0:  *# 表示未重复初始输入状态*
1182. result.append(lines\_str[index\_])
1183. self.table.insert('', tk.END, values=lines\_str[index\_].split())  *# 添加数据到末尾*
1184. if lines\_str[index\_] in result and i == 1:  *# 主要用于重复后第二次判断用的，如果不在里面，则代表是其他重复情况*
1185. copy.append(lines\_str[index\_])
1186. if i == 0:
1187. length = len(result)
1188. *# print(length)*
1189. if i == 1:
1190. result = [co for co in copy]
1191. if len(result) == 0:
1192. *# print('请输入其他有效信息！该信息在内部查不到任何内容！')*
1193. tk.messagebox.showinfo(title='消息提示框', message='请输入其他有效信息！该信息在内部查不到任何内容！')
1194. *# continue*
1195. elif len(result) >= 1:
1196. length = len(result)
1197. if length > 1:
1198. *# k = input('查阅的图书不止一本！要想继续按照关键词查找图书吗？(y/n)')*
1199. tk.messagebox.showinfo(title='消息提示框', message='查阅的图书不止一本！请重新选择查找方式并按下列筛选图书输入新的关键词')
1200. *# if k == "Y" or k == 'y':*
1201. *#     print('继续查询！')*
1202. *#     print('请在下方选择追加输入的条件！')  # 然后这里就会挑出循环到达while(1)开头*
1203. i = 1
1204. *#     #continue*
1205. *# if k == 'N' or k == 'n':*
1206. *#     print('返回初始图书查找页面！')*
1207. *#     lines\_str = line2  # 为了防止图书查找有重复后再次查找后想继续查找别的；不然只会在result的范围内跑*
1208. if length == 1:
1209. *# k = input('以上可能是您要的图书！请问还要继续查询吗(y/n)')*
1210. tk.messagebox.showinfo(title='消息提示框', message='以上可能是您要的图书！')
1211. *# if k == 'y' or k == "Y":*
1212. *#     print('继续查找图书！')*
1213. result = []
1214. i = 0
1215. self.finalfind\_index = index\_  *# 将找出来的书在库存里的序号放入finalfind\_index*
1216. *#     #continue*
1217. *# if k == 'n' or k == "N":*
1218. *#     print('查找结束，即将退出！')*
1219. *#     #break*
1220. if length == 0:
1221. *# k = input('没有找到您所要的书本！请更换关键词或按照其他方式查找！请问还要继续查询吗？(y/n)')*
1222. tk.messagebox.showinfo(title='消息提示框', message='没有找到您所要的书本！请更换关键词或按照其他方式查找！')
1223. *# if k == 'y' or k == "Y":*
1224. *#     print('继续查找图书！')*
1225. *#     #continue*
1226. *# if k == 'n' or k == "N":*
1227. *#     print('查找结束，即将退出！')*
1228. *#     #break*
1229. if self.method == '书名':
1230. index\_gather = [e for e, m in enumerate(name) if a in m]  *# 为了规避index只能查找一个*
1231. print(name)
1232. copy = []  *# 专门针对重复情况而设置的， 正常情况下copy不会动为0， 重复情况可由i == 1 引出*
1233. if len(lines\_str) > 0:  *# 这一个判断用于删除treeview表格内之前的数据，然后再打印出新的图书信息覆盖*
1234. self.delTreeview(self)
1235. for index\_ in index\_gather:
1236. if i == 0:  *# 表示未重复初始输入状态*
1237. result.append(lines\_str[index\_])
1238. self.table.insert('', tk.END, values=lines\_str[index\_].split())  *# 添加数据到末尾*
1239. if lines\_str[index\_] in result and i == 1:  *# 主要用于重复后第二次判断用的，如果不在里面，则代表是其他重复情况*
1240. copy.append(lines\_str[index\_])
1241. if i == 0:
1242. length = len(result)
1243. *# print(length)*
1244. if i == 1:
1245. result = [co for co in copy]
1246. if len(result) == 0:
1247. tk.messagebox.showinfo(title='消息提示框', message='请输入其他有效信息！该信息在内部查不到任何内容！')
1248. elif len(result) >= 1:
1249. length = len(result)
1250. if length > 1:
1251. tk.messagebox.showinfo(title='消息提示框', message='查阅的图书不止一本！请重新选择查找方式并按下列筛选图书输入新的关键词')
1252. i = 1
1253. if length == 1:
1254. tk.messagebox.showinfo(title='消息提示框', message='以上可能是您要的图书！')
1255. result = []
1256. i = 0
1257. self.finalfind\_index = index\_  *# 将找出来的书在库存里的序号放入finalfind\_index*
1258. if length == 0:
1259. tk.messagebox.showinfo(title='消息提示框', message='没有找到您所要的书本！请更换关键词或按照其他方式查找！')
1260. if self.method == '作者名称':
1261. index\_gather = [e for e, m in enumerate(public\_address) if a in m]  *# 为了规避index只能查找一个*
1262. *# print(public\_address) #实际上是作者名称 public address*
1263. copy = []  *# 专门针对重复情况而设置的， 正常情况下copy不会动为0， 重复情况可由i == 1 引出*
1264. if len(lines\_str) > 0:  *# 这一个判断用于删除treeview表格内之前的数据，然后再打印出新的图书信息覆盖*
1265. self.delTreeview(self)
1266. for index\_ in index\_gather:
1267. if i == 0:  *# 表示未重复初始输入状态*
1268. result.append(lines\_str[index\_])
1269. self.table.insert('', tk.END, values=lines\_str[index\_].split())  *# 添加数据到末尾*
1270. if lines\_str[index\_] in result and i == 1:  *# 主要用于重复后第二次判断用的，如果不在里面，则代表是其他重复情况*
1271. copy.append(lines\_str[index\_])
1272. if i == 0:
1273. length = len(result)
1274. *# print(length)*
1275. if i == 1:
1276. result = [co for co in copy]
1277. if len(result) == 0:
1278. tk.messagebox.showinfo(title='消息提示框', message='请输入其他有效信息！该信息在内部查不到任何内容！')
1279. elif len(result) >= 1:
1280. length = len(result)
1281. if length > 1:
1282. tk.messagebox.showinfo(title='消息提示框', message='查阅的图书不止一本！请重新选择查找方式并按下列筛选图书输入新的关键词')
1283. i = 1
1284. if length == 1:
1285. tk.messagebox.showinfo(title='消息提示框', message='以上可能是您要的图书！')
1286. result = []
1287. i = 0
1288. self.finalfind\_index = index\_  *# 将找出来的书在库存里的序号放入finalfind\_index*
1289. if length == 0:
1290. tk.messagebox.showinfo(title='消息提示框', message='没有找到您所要的书本！请更换关键词或按照其他方式查找！')
1291. if self.method == '出版社信息':
1292. index\_gather = [e for e, m in enumerate(public\_time) if a in m]  *# 为了规避index只能查找一个*
1293. *# print(public\_time)  #实际上是出版社信息 public time*
1294. copy = []  *# 专门针对重复情况而设置的， 正常情况下copy不会动为0， 重复情况可由i == 1 引出*
1295. if len(lines\_str) > 0:  *# 这一个判断用于删除treeview表格内之前的数据，然后再打印出新的图书信息覆盖*
1296. self.delTreeview(self)
1297. for index\_ in index\_gather:
1298. if i == 0:  *# 表示未重复初始输入状态*
1299. result.append(lines\_str[index\_])
1300. self.table.insert('', tk.END, values=lines\_str[index\_].split())  *# 添加数据到末尾*
1301. if lines\_str[index\_] in result and i == 1:  *# 主要用于重复后第二次判断用的，如果不在里面，则代表是其他重复情况*
1302. copy.append(lines\_str[index\_])
1303. if i == 0:
1304. length = len(result)
1305. *# print(length)*
1306. if i == 1:
1307. result = [co for co in copy]
1308. if len(result) == 0:
1309. tk.messagebox.showinfo(title='消息提示框', message='请输入其他有效信息！该信息在内部查不到任何内容！')
1310. elif len(result) >= 1:
1311. length = len(result)
1312. if length > 1:
1313. tk.messagebox.showinfo(title='消息提示框', message='查阅的图书不止一本！请重新选择查找方式并按下列筛选图书输入新的关键词')
1314. i = 1
1315. if length == 1:
1316. tk.messagebox.showinfo(title='消息提示框', message='以上可能是您要的图书！')
1317. result = []
1318. i = 0
1319. self.finalfind\_index = index\_  *# 将找出来的书在库存里的序号放入finalfind\_index*
1320. if length == 0:
1321. tk.messagebox.showinfo(title='消息提示框', message='没有找到您所要的书本！请更换关键词或按照其他方式查找！')
1322. if self.method == '出版时间':
1323. index\_gather = [e for e, m in enumerate(price) if a in m]  *# 为了规避index只能查找一个*
1324. *# print(price) #实际上是出版时间 price*
1325. copy = []  *# 专门针对重复情况而设置的， 正常情况下copy不会动为0， 重复情况可由i == 1 引出*
1326. if len(lines\_str) > 0:  *# 这一个判断用于删除treeview表格内之前的数据，然后再打印出新的图书信息覆盖*
1327. self.delTreeview(self)
1328. for index\_ in index\_gather:
1329. if i == 0:  *# 表示未重复初始输入状态*
1330. result.append(lines\_str[index\_])
1331. self.table.insert('', tk.END, values=lines\_str[index\_].split())  *# 添加数据到末尾*
1332. if lines\_str[index\_] in result and i == 1:  *# 主要用于重复后第二次判断用的，如果不在里面，则代表是其他重复情况*
1333. copy.append(lines\_str[index\_])
1334. if i == 0:
1335. length = len(result)
1336. *# print(length)*
1337. if i == 1:
1338. result = [co for co in copy]
1339. if len(result) == 0:
1340. tk.messagebox.showinfo(title='消息提示框', message='请输入其他有效信息！该信息在内部查不到任何内容！')
1341. elif len(result) >= 1:
1342. length = len(result)
1343. if length > 1:
1344. tk.messagebox.showinfo(title='消息提示框', message='查阅的图书不止一本！请重新选择查找方式并按下列筛选图书输入新的关键词')
1345. i = 1
1346. if length == 1:
1347. tk.messagebox.showinfo(title='消息提示框', message='以上可能是您要的图书！')
1348. result = []
1349. i = 0
1350. self.finalfind\_index = index\_  *# 将找出来的书在库存里的序号放入finalfind\_index*
1351. if length == 0:
1352. tk.messagebox.showinfo(title='消息提示框', message='没有找到您所要的书本！请更换关键词或按照其他方式查找！')
1353. if self.method == '书本价格':
1354. index\_gather = [e for e, m in enumerate(author\_name) if a in m]  *# 为了规避index只能查找一个*
1355. *# print(author\_name) #实际上对应的是书本价格author name*
1356. copy = []  *# 专门针对重复情况而设置的， 正常情况下copy不会动为0， 重复情况可由i == 1 引出*
1357. if len(lines\_str) > 0:  *# 这一个判断用于删除treeview表格内之前的数据，然后再打印出新的图书信息覆盖*
1358. self.delTreeview(self)
1359. for index\_ in index\_gather:
1360. if i == 0:  *# 表示未重复初始输入状态*
1361. result.append(lines\_str[index\_])
1362. self.table.insert('', tk.END, values=lines\_str[index\_].split())  *# 添加数据到末尾*
1363. if lines\_str[index\_] in result and i == 1:  *# 主要用于重复后第二次判断用的，如果不在里面，则代表是其他重复情况*
1364. copy.append(lines\_str[index\_])
1365. if i == 0:
1366. length = len(result)
1367. *# print(length)*
1368. if i == 1:
1369. result = [co for co in copy]
1370. if len(result) == 0:
1371. tk.messagebox.showinfo(title='消息提示框', message='请输入其他有效信息！该信息在内部查不到任何内容！')
1372. elif len(result) >= 1:
1373. length = len(result)
1374. if length > 1:
1375. tk.messagebox.showinfo(title='消息提示框', message='查阅的图书不止一本！请重新选择查找方式并按下列筛选图书输入新的关键词')
1376. i = 1
1377. if length == 1:
1378. tk.messagebox.showinfo(title='消息提示框', message='以上可能是您要的图书！')
1379. result = []
1380. i = 0
1381. self.finalfind\_index = index\_  *# 将找出来的书在库存里的序号放入finalfind\_index*
1382. if length == 0:
1383. tk.messagebox.showinfo(title='消息提示框', message='没有找到您所要的书本！请更换关键词或按照其他方式查找！')
1384. *# 显示图书页面 OK*
1385. class ShowAllMenu:
1386. def \_\_init\_\_(self, window):
1387. *# 页面信息*
1388. self.window = window
1389. self.window.title("史上最丑图书管理系统")
1390. self.window.geometry("800x500")
1391. *# 插入数据*
1392. def insert():
1393. lines = []  *# 储存了数组形式， 每个元素都是分开的*
1394. lines\_str = []  *# 字符串形式， 直接输出结果*
1395. *# 读取文件内数据*
1396. with open('library.txt', 'r') as f:
1397. line = f.readline()
1398. while line:
1399. *# print(line.rstrip())  # 后面跟 ',' 将忽略换行符*
1400. lines\_str.append(line.rstrip('\n'))
1401. line = line.strip().split()
1402. lines.append(line)
1403. line = f.readline()
1404. f.close()
1405. for index, data in enumerate(lines):
1406. self.table.insert('', tk.END, values=data)  *# 添加数据到末尾*
1407. *# 排序除价格外的书目*
1408. def treeview\_sort\_column(tv, col, reverse):  *# Treeview、列名、排列方式*
1409. l = [(tv.set(k, col), k) for k in tv.get\_children('')]
1410. *#print(l)*
1411. l.sort(reverse=reverse)  *# 排序方式*
1412. *# rearrange items in sorted positions*
1413. for index, (val, k) in enumerate(l):  *# 根据排序后索引移动*
1414. tv.move(k, '', index)
1415. tv.heading(col, command=lambda: treeview\_sort\_column(tv, col, not reverse))  *# 重写标题，使之成为再点倒序的标题*
1416. *# 定义框架放置表格和滚动条*
1417. if 1:
1418. self.tabel\_frame = tkinter.Frame(window)
1419. self.tabel\_frame.pack()
1420. *# 定义滚动条*
1421. xscroll = tk.Scrollbar(self.tabel\_frame, orient=tk.HORIZONTAL)
1422. yscroll = tk.Scrollbar(self.tabel\_frame, orient=tk.VERTICAL)
1423. columns = ['ISBN号', '书名', '作者名称', '出版社信息', '出版时间', '书本价格', '图书简介']
1424. self.table = tk.ttk.Treeview(
1425. master=self.tabel\_frame,  *# 父容器*
1426. height=15,  *# 表格显示的行数,height行*
1427. columns=columns,  *# 显示的列*
1428. show='headings',  *# 隐藏首列*
1429. xscrollcommand=xscroll.set,  *# x轴滚动条*
1430. yscrollcommand=yscroll.set,  *# y轴滚动条*
1431. )
1432. *# 定义表头*
1433. self.table.heading(column='ISBN号', text='INBS号',
1434. command=lambda \_col='ISBN号': treeview\_sort\_column(self.table, \_col,
1435. False))  *# 点击表头出发lambda函数*
1436. self.table.heading('书名', text='书名', command=lambda \_col='书名': treeview\_sort\_column(self.table, \_col, False))
1437. self.table.heading('作者名称', text='作者名称',
1438. command=lambda \_col='作者名称': treeview\_sort\_column(self.table, \_col, False))
1439. self.table.heading('出版社信息', text='出版社信息',
1440. command=lambda \_col='出版社信息': treeview\_sort\_column(self.table, \_col, False))
1441. self.table.heading('出版时间', text='出版时间',
1442. command=lambda \_col='出版时间': treeview\_sort\_column(self.table, \_col, False))
1443. self.table.heading('书本价格', text='书本价格',
1444. command=lambda \_col='书本价格': treeview\_sort\_column(self.table, \_col, False))
1445. self.table.heading('图书简介', text='图书简介', command=lambda \_col='图书简介': treeview\_sort\_column(self.table, \_col,
1446. False))  *# 点击表头出发lambda函数)*
1447. *# 定义列*
1448. self.table.column('ISBN号', width=100, minwidth=100, anchor='s', )
1449. self.table.column('书名', width=100, minwidth=100, anchor='s')
1450. self.table.column('作者名称', width=100, minwidth=100, anchor='s')
1451. self.table.column('出版社信息', width=150, minwidth=150, anchor='s')
1452. self.table.column('出版时间', width=100, minwidth=100, anchor='s')
1453. self.table.column('书本价格', width=80, minwidth=80, anchor='s')
1454. self.table.column('图书简介', width=200, minwidth=100, anchor='s')
1455. *# config绑定滚动条为滚动事件 pack放置滚动条再最边边*
1456. xscroll.config(command=self.table.xview)
1457. xscroll.pack(side=tk.BOTTOM, fill=tk.X)
1458. yscroll.config(command=self.table.yview)
1459. yscroll.pack(side=tk.RIGHT, fill=tk.Y)
1460. self.table.pack(pady=20)
1461. insert()  *# 插入library.txt内的数据*
1462. *# 返回主菜单按钮*
1463. self.backbut = tk.Button(self.window, text="返回主菜单", bd=5, font=('黑体', 20, 'bold'), command=self.back)
1464. self.backbut.place(x=500, y=400, width=200, height=50)
1465. def back(self):
1466. self.backbut.destroy()
1467. self.tabel\_frame.destroy()
1468. MainMenu(root)
1469. *# main*
1470. root = tk.Tk()  *# root加入界面主循环*
1471. mainmenu = MainMenu(root)
1472. root.mainloop()

**五、测试数据及其结果分析**

运行后，程序的主菜单界面如下。



1. 显示图书信息测试



（2）增加图书信息测试



如果输入图书信息不完整，则会提示完善图书信息：

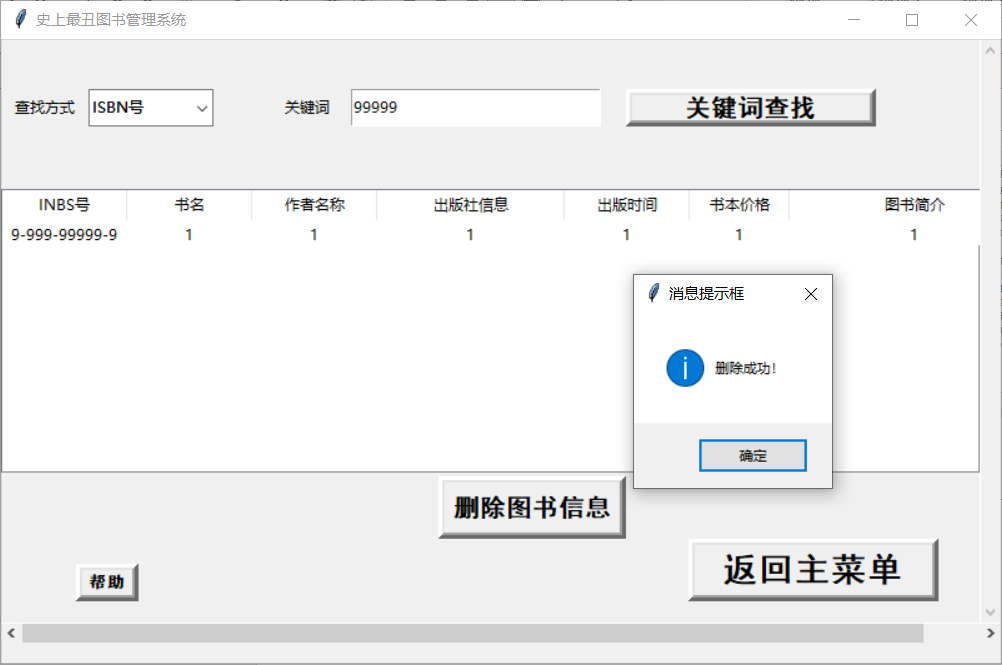


（3）客户排序测试

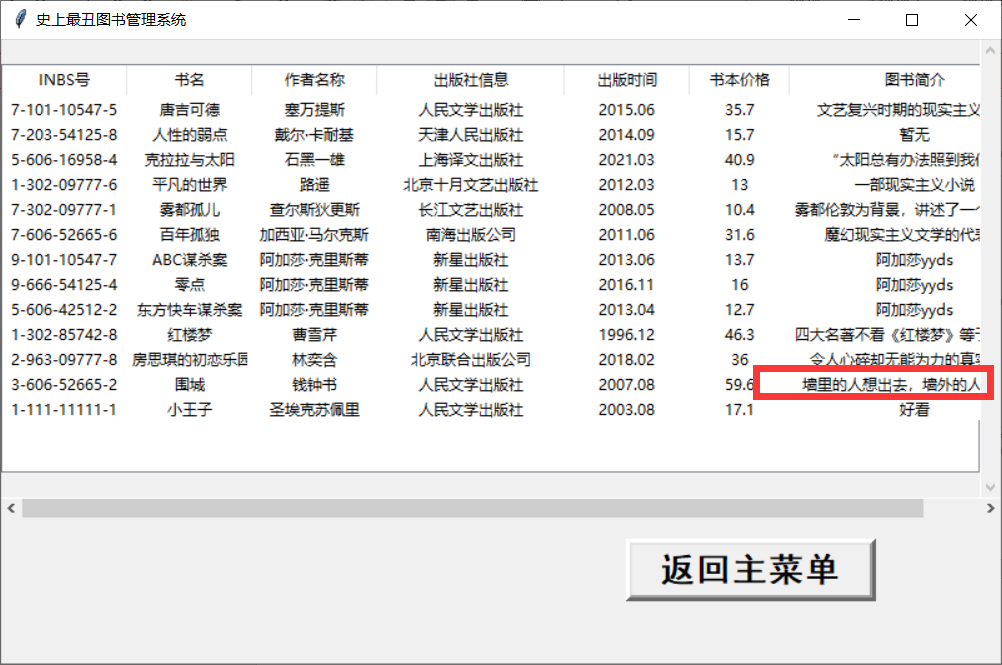
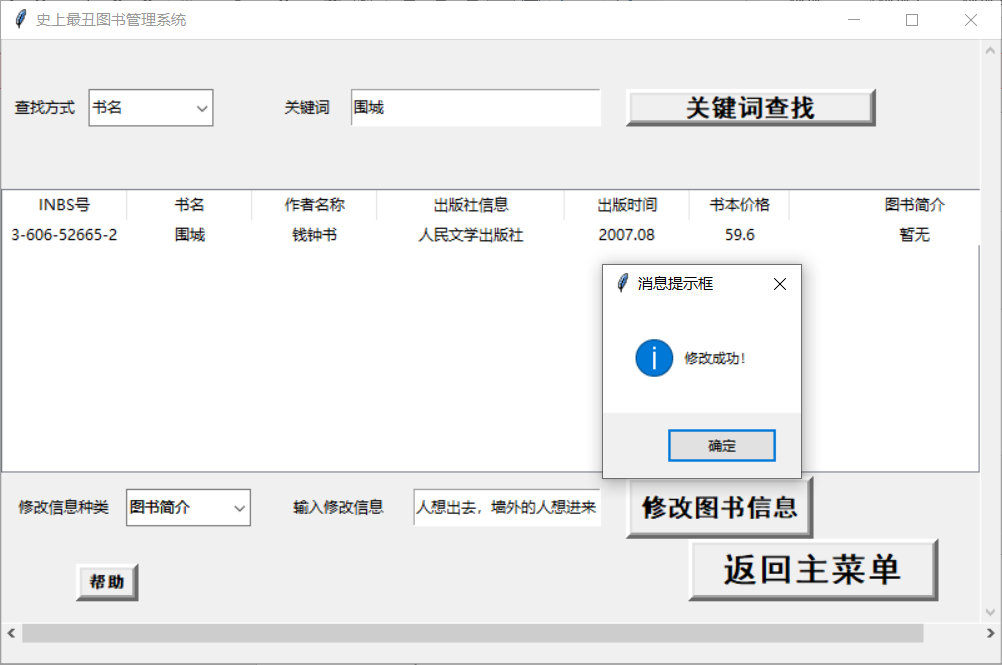


排序成功。

（4）删除图书信息测试



（5）修改图书信息测试



修改成功。

（6）检验ISBN合法性和唯一性测试



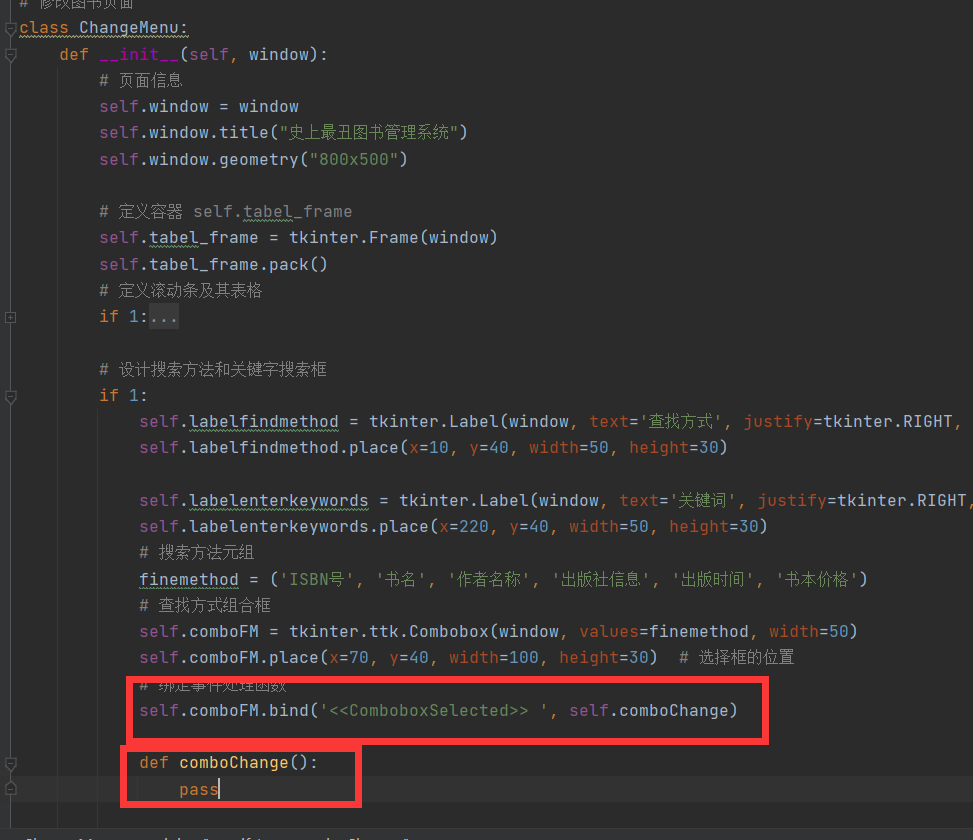
合法性检验成功！



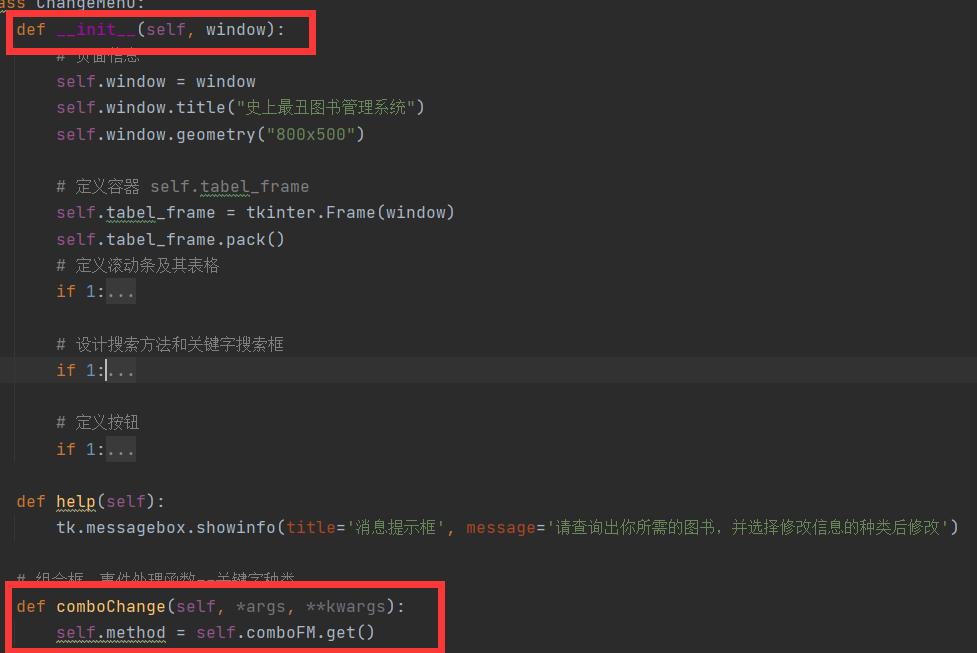
唯一性检验成功！

**六、调试过程中的问题**

（1）前端python代码编写：类的init函数内不能声明函数，如下图的下拉栏绑定的回调函数comboChange，实际上会报错且不能运行



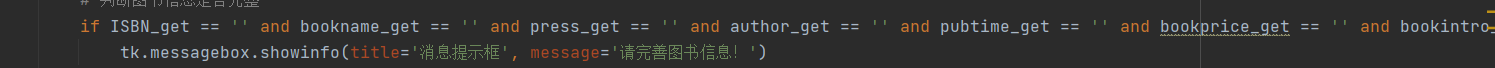
解决方法：将回调函数定义在init函数外，这样函数内部再调用self.comboChange（多加了self.）这样就解决了init内不能声明函数但回调函数一定要定义的问题



（2）在进行图书查找时，遇到如果查找结果有多本书则需要在这多本书中进行其他关键词的查找的情况，原来想着是在用一个逻辑解决这个问题，后来发现无法解决。于是想了在循环中用i == 0 与i==1的两种情况去表示重复和不重复两种情况，这样在一个循环内就可以使得遇到哪种情况都可以顺利的进行查找。

1. a = input("请输入关键词:")
2. index\_gather = [e for e, m in enumerate(number) if a in m]  *# 为了规避index只能查找一个*
3. copy = []  *# 专门针对重复情况而设置的， 正常情况下copy不会动为0， 重复情况可由i == 1 引出*
4. for index\_ in index\_gather:
5. if i == 0:  *# 表示未重复初始输入状态*
6. result.append(lines\_str[index\_])
7. print(lines\_str[index\_])
8. if lines\_str[index\_] in result and i == 1:  *# 主要用于重复后第二次判断用的，如果不在里面，则代表是其他重复情况*
9. copy.append(lines\_str[index\_])
10. print(lines\_str[index\_])
11. if i == 0:
12. length = len(result)
13. if i == 1:
14. result = [co for co in copy]
15. if len(result) == 0:
16. print('请输入其他有效信息！该信息在内部查不到任何内容！')
17. continue
18. elif len(result) >= 1:
19. length = len(result)
20. if length > 1:
21. k = input('查阅的图书不止一本！要想继续按照关键词查找图书吗？(y/n)')
22. if k == "Y" or k == 'y':
23. print('继续查询！')
24. print('请在下方选择追加输入的条件！')  *# 然后这里就会挑出循环到达while(1)开头*
25. i = 1
26. continue
27. if k == 'N' or k == 'n':
28. print('返回初始图书查找页面！')
29. lines\_str = line2  *# 为了防止图书查找有重复后再次查找后想继续查找别的；不然只会在result的范围内跑*
30. if length == 1:
31. k = input('以上可能是您要的图书！请问还要继续查询吗(y/n)')
32. if k == 'y' or k == "Y":
33. print('继续查找图书！')
34. result = []
35. i = 0
36. continue
37. if k == 'n' or k == "N":
38. print('查找结束，即将退出！')
39. break
40. if length == 0:
41. k = input('没有找到您输入的关键词！请更换关键词或按照其他方式查找！请问还要继续查询吗？(y/n)')
42. if k == 'y' or k == "Y":
43. print('继续查找图书！')
44. continue
45. if k == 'n' or k == "N":
46. print('查找结束，即将退出！')
47. break

（3）判断输入的图书信息是否完整时，我用的是以下代码



意思就是如果输入框的值为空则进入判断，提示完善图书信息并不写入图书信息。应该有的逻辑是只要有一个为空就进入判断，调试的时候发现有空的输入，但还是写入成功。后来审查发现判断的与或条件应该使用或，而不是上图那样用与。



改成上图，bug解除

**七、课程总结**

为期一周的程序设计告一段落了。到目前为止代码前后端的编写已经完成。在这三天的时间里，小谢和我算是体验了一把程序设计的事与愿违。

程序设计周开始前的周末，我们已经确认好了题目-----图书管理系统。我们初步决定在该项目C语言的基础上完善我们题目要求的功能，并采用Qt作为前端人机交互界面的实现方法，大致分工为小谢为后端，编写C语言算法程序实现排序、增改等功能；我为前端，用基于C++的Qt creator实现人机交互页面。一开始我们都信誓旦旦，一天就能写完！

第一天我们都兴冲冲的去机房干活，对未来充满希望和憧憬。

由于我蹩脚的C++能力以及学习能力，一个上午现学现卖，只实现了Qt中的按钮切换窗口界面功能

第一次接触Qt是在pixhawk飞控开发教程文档里看到的，pix的地面站就是用Qt开发的。Qt是一个图形化界面编程平台，拥有方便的图形化编程方法，拥有一番天地。Qt之所以能把各个控件连接起来，关键在于信号与槽的两个概念。两个控件，比如输入框和按钮之间，它们是毫不相关的两个Class，类里只定义了自己长什么样子和能干什么，而Qt中的信号和槽就是连接这两者的桥梁。我们可以设置按钮按下去发送信号给按钮外部，这时候输入框里的槽函数会接收到信号并且完成槽函数内部的代码工作（就像是回调函数一样），槽和信号之间通过connect函数绑定，即绑定什么信号响应什么槽，且一个信号可以绑定多个槽。

理解了信号与槽就等于理解了Qt的一半。可是作为小丑的我理解了这个远远不够，槽函数内部怎么编写，需要理解Qt的工程代码逻辑，我研究了整整一天，毫无头绪...........就简单的按键切换界面这一功能来说，网络上大神的方法百花齐放，有用自定义信号的，有自己写control类的...这些都不利于我理解代码结构。于是Ctrl CV后虽然能实现界面切换，但是我不能在此基础上改动，因为我看不懂它槽函数内的代码逻辑：是把现在这个窗体的class关掉再开另一个窗体的class吗？于是我遇到了瓶颈。

一天过去了，我的进展几乎为零。

长话短说，C语言的char型和int型转换把小谢整不明白了。作为使用python编程的他，根本不会考虑这么多数据类型的问题（懂得都懂），这个类型转换的问题也卡住了他许久，最终破防。

没错，白忙活了一天，我们决定用python从头开始写，基于原有C语言的代码工程将其功能迁移过来。而我，抛弃这个对新手不友好的Qt，采用了python自带的tkinter库来进行人机交互。

tkinter是python自带的库，只要下载了python就能import。之所以选择它是因为他代码结构简单，对新手友好。

tkinter中对窗体的定义与Qt大相径庭，也是采用类来定义。主要运行思路是将窗体放入主循环mainloop就可以了。窗口切换就简单的多了，我在按钮按下进入的事件函数（理解为回调函数和上面的槽函数）内将mainloop里这个窗体去掉，重新打开另一个窗体即可，代码只需1行......................

于是我便上手了，现学现卖，实现了按钮、文本、输入框、下拉选择栏等功能，并且在网上能搜到这些控件如何使用，有什么方法和函数。接下来就是和小谢后端链条了。

我认为，联调最重要的是和队友交流好，把代码需要的接口告诉后端队友，比如一个函数输入什么，输出0或1作为判断还是其它等等。但由于事先我并不会tkinter，无法与小谢沟通，为了节省时间，小谢就按照他的思路来写后端算法，接口等我把前端写好在改进。