项目总结报告

题目：E

小组成员：范昌源（21373453）、柳浩东（21373450）、赵彦豪（21373454）、曹伯炜（20373516）

2023年7月

一、功能简介（简述项目拥有的功能）

1. 用户和管理员的注册、登录和个人信息管理，修改密码。
2. 用户和管理员可以选择加入组。管理员可以创建组和添加用户，用户可以自行搜索和加入组。
3. 用户和管理员可以上传问题，支持单个问题上传和通过读取文件批量上传。
4. 问题分组功能。设计自己的或利用现有的数据结构，根据章节或其他标准将问题组织成类别。在解决问题时，用户可以选择一组特定的问题来处理。
5. 问题共享功能。用户可以选择与特定的组共享一组问题，或者让所有人都可以使用这些问题。共享问题的接收者可以访问问题组。
6. 搜索问题组。根据主题或者问题组名称搜索问题。
7. 错误日志。根据用户的错误答案、错误频率以及用户指定的主题和问题数量，推荐题库中的错题。
8. 屏蔽敏感词汇。管理员输入敏感词汇，将包含敏感词汇的问题进行删除。

二、已完成任务

必做任务完成情况（已完成数量7/必做任务总数7）

1. 用户和管理员的注册、登录和个人信息管理。

2. 用户和管理员可以选择加入组。管理员可以创建组和添加用户，用户可以自行搜索和加入组。

3.用户和管理员可以上传问题，支持单个问题上传和通过读取文件批量上传。

4. 问题分组功能。设计自己的或利用现有的数据结构，根据章节或其他标准将问题组织成类别。在解决问题时，用户可以选择一组特定的问题来处理。解决问题的界面应该根据个人的喜好来设计，不要有过多的要求。

5. 问题共享功能。用户可以选择与特定的组共享一组问题，或者让所有人都可以使用这些问题。共享问题的接收者可以访问问题组。

6. 搜索问题组。搜索应该具有可定制的参数，但搜索范围应该包括共享的问题组和用户上传的问题。它不应该搜索尚未共享的问题组。

7. 错误日志。根据用户的错误答案、错误频率以及用户指定的主题和问题数量，参考相关推荐算法生成一组问题，用户应该优先使用您选择的科学有效的算法重新解决这些问题。

选做任务完成情况（已完成数量2/选做任务总数3）

选做任务自行设计的部分，请标名自定义。

1. 屏蔽敏感词汇。管理员输入敏感词汇，将包含敏感词汇的问题进行删除。
2. 修改用户密码。（自定义）

3. 小组成员可以选择退出组（自定义）

三、总体设计方案

（一）前端部分  
一.用户登录和注册：

1.1功能

用户可以在登录界面输入用户名和密码来登录平台。如果用户没有账号，可以选择用户注册或管理员注册，输入用户名和密码进行注册。登录成功后，打开应用程序主界面。

管理员可以创建小组，添加用户，并筛选敏感词并删除题库中的敏感词。

1.2代码分析

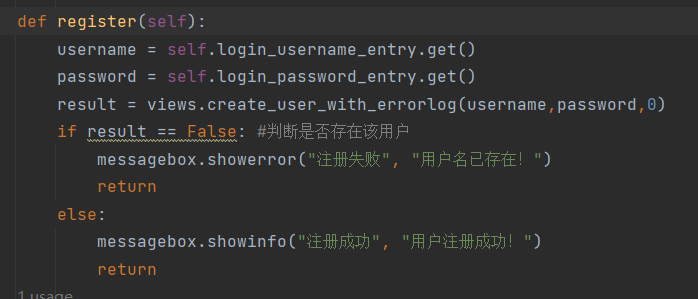


初始化方法\_\_init\_\_(self, root)接收一个tkinter的Tk()实例root，作为整个登录应用程序的根窗口。创建self.login\_frame，一个tk.Frame实例，用于放置登录界面的相关控件。在login\_frame中创建用户名输入框、密码输入框和登录、用户注册、管理员注册按钮。



用户登录功能：

login(self)方法用于处理用户登录逻辑。获取用户名和密码输入框中的值。调用views.login(username, password)，这是与后端代码对接的部分。根据返回结果，判断登录是否成功，如果成功则弹出登录成功提示框，并创建App的实例来打开主界面。



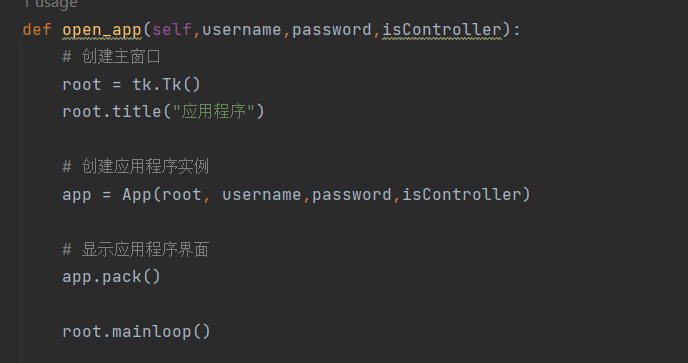
用户注册功能：

register(self)方法用于处理用户注册逻辑。获取用户名和密码输入框中的值。调用views.create\_user\_with\_errorlog(username, password, 0)，这是与后端代码对接的部分。根据返回结果，判断注册是否成功，并弹出相应的提示框。



管理员注册功能：

register2(self)方法用于处理管理员注册逻辑。获取用户名和密码输入框中的值。调用views.create\_user\_with\_errorlog(username, password, 1)，这是与后端代码对接的部分。根据返回结果，判断注册是否成功，并弹出相应的提示框。



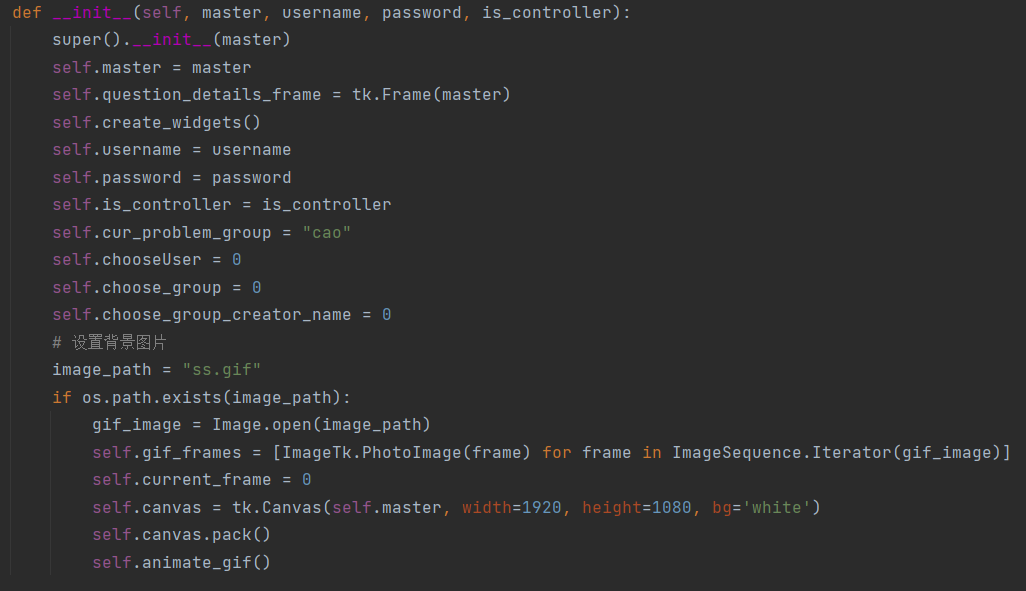
open\_app(self, username, password, isController)方法：用于打开应用程序的主界面。创建一个新的tk.Tk()实例，作为主界面的根窗口。创建一个App的实例，用于显示应用程序的主界面。开启主界面的消息循环。

二.应用程序主界面

2.1功能

创建应用程序的主界面，包含多个功能按钮，如上传问题、搜索问题、搜索小组、查看错题日志、解题功能等。这些按钮提供用户可见的操作入口。

2.2代码分析



\_\_init\_\_(self, master, username, password, is\_controller)：作为构造函数，接受四个参数：

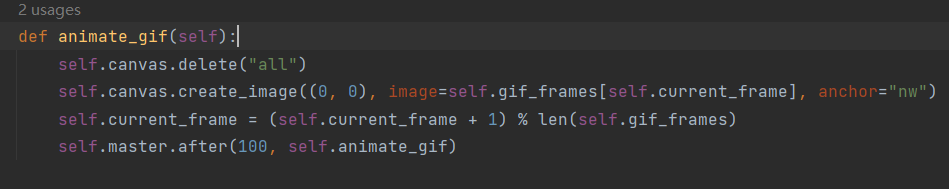
master：父部件或主应用程序窗口。

username：已登录用户的用户名。

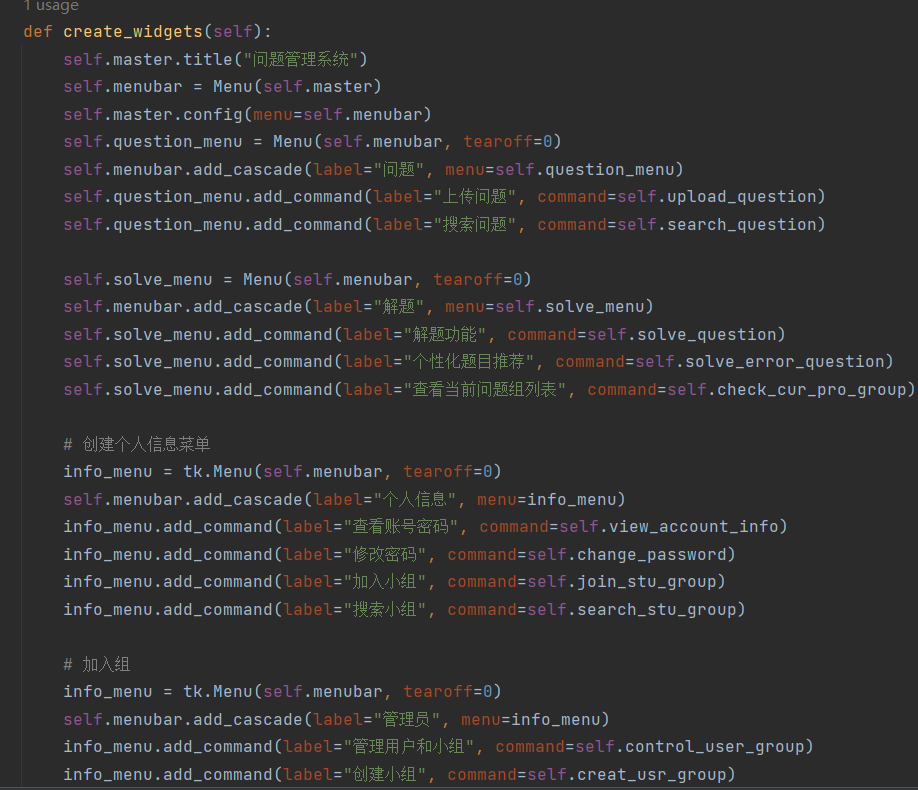
password：已登录用户的密码。

is\_controller：一个布尔标志，指示用户是管理员（True）还是普通用户（False）。

构造函数初始化了各种属性，例如username、password和is\_controller。如果文件"ss.gif"存在，它还会在画布上显示一个背景图像（一个动画GIF）。



animate\_gif(self)：这个方法负责在画布上播放动画GIF背景图像。它不断地从GIF中显示帧，创建了一个动画效果。



create\_widgets(self)：这个方法创建了主应用程序窗口的小部件。它设置了主菜单及其子菜单的不同选项：

"问题" 菜单包含选项："上传问题" 和 "搜索问题"。

"解题" 菜单包含选项："解题功能"、"个性化题目推荐" 和 "查看当前问题组列表"。

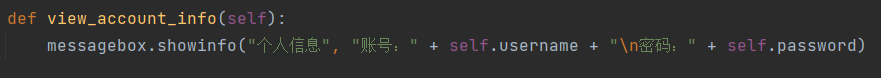
"个人信息" 菜单包含选项："查看账号密码"、"修改密码"、"加入小组" 和 "搜索小组"。

"管理员" 菜单包含选项："管理用户和小组" 和 "创建小组"。

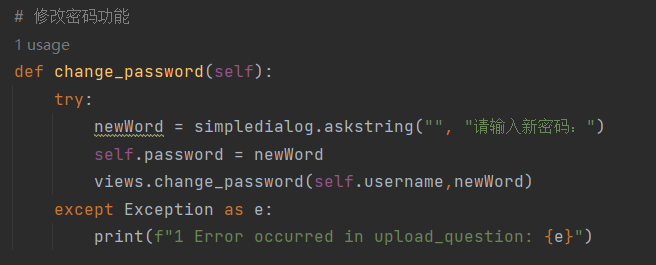
"问题日志" 菜单包含选项："查看当前错题"。



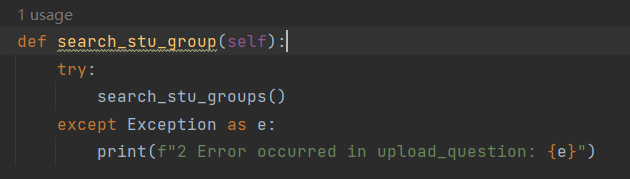
run(self)：这个方法启动主事件循环，允许应用程序处理用户交互和事件。



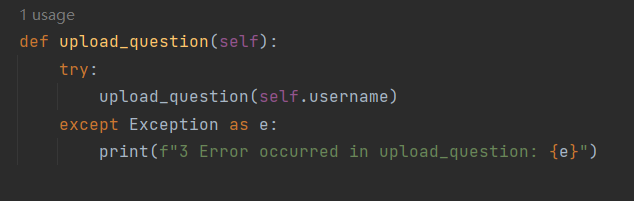
view\_account\_info(self): 这个函数用于显示个人账号信息。它会通过消息框显示用户名和密码的信息。



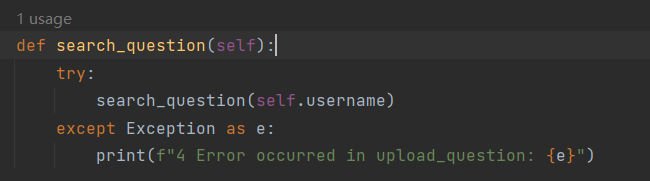
change\_password(self): 这个函数实现了修改密码的功能。它会通过简单的对话框（simpledialog）让用户输入新密码，并将新密码赋值给self.password，即更新当前用户对象的密码信息。然后，它会调用views.change\_password函数来更新数据库中用户的密码。



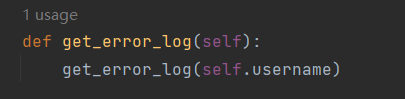
search\_stu\_group(self): 用于搜索小组，调用了名为search\_stu\_groups的函数，该函数用于从数据库中查询和显示小组信息。



upload\_question(self): 用于上传问题，调用了名为upload\_question的函数，该函数用于从用户端上传问题数据到数据库。



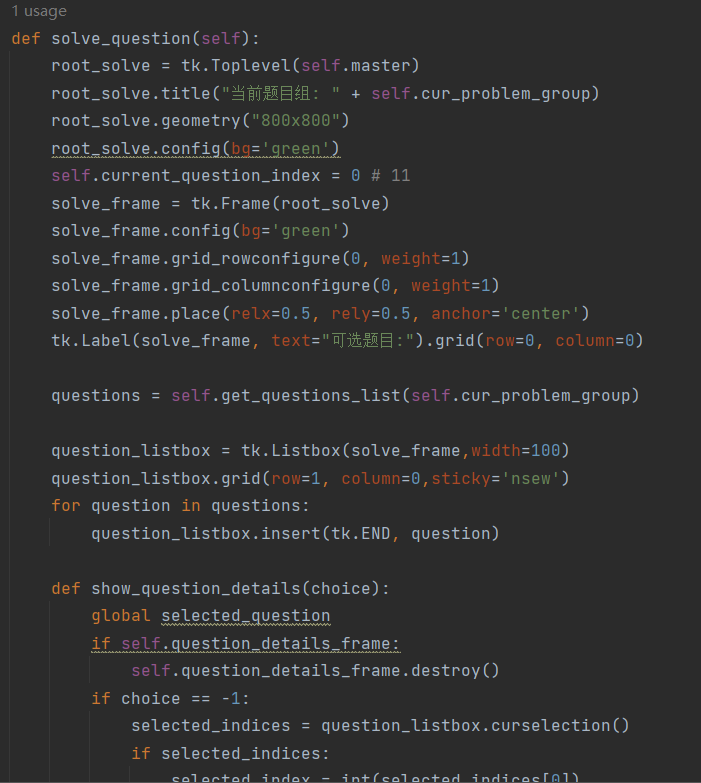
search\_question(self): 用于搜索问题。调用了名为search\_question的函数，用于从数据库中查询问题信息并展示给用户。



get\_error\_log(self): 用于查看当前错题（错误日志）。



check\_cur\_pro\_group(self): 当用户点击"查看当前问题组列表"菜单选项时，该函数被调用。它会创建一个新的顶层窗口（Toplevel）来展示当前可选的问题组列表。用户可以在列表中选择一个问题组，并点击"选择该问题组"按钮，完成选择。所选的问题组会保存在self.cur\_problem\_group中。创建新的顶层窗口root\_groups来展示问题组列表。在顶层窗口中添加标签和问题组列表框。获取当前可选的问题组列表并填充问题组列表框。定义"选择该问题组"按钮的回调函数sel()，该函数用于获取用户选择的问题组并保存在self.cur\_problem\_group中。显示顶层窗口。



solve\_question(self): 当用户点击"解题功能"菜单选项时，该函数被调用。它会创建一个新的顶层窗口（Toplevel），用于解题。在解题窗口中，用户可以选择并回答问题，并查看答案的正确与否。可以通过"下一题"按钮进入下一个问题。创建新的顶层窗口root\_solve来展示解题界面。

在解题界面中添加标签、题目列表框和按钮。

获取当前问题组的题目列表并填充题目列表框。

定义"解题"按钮的回调函数，用于显示选定题目的详情，这里调用show\_question\_details(-1)。

定义"下一题"按钮的回调函数，用于显示下一题，这里调用nextQ()。

显示解题界面。

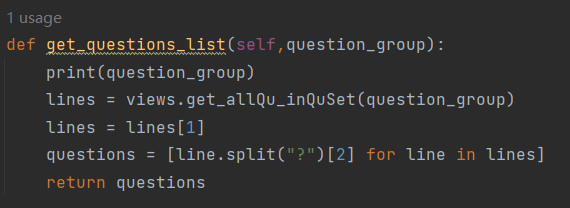
show\_question\_details(choice): 该函数用于显示选定题目的详情。choice参数为-1时表示用户点击"解题"按钮，为0时表示用户点击"下一题"按钮。

获取当前选定的题目。创建一个新的题目详情窗口。

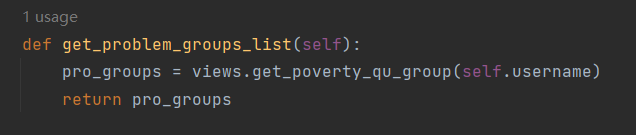
获取选定题目的详细信息，并根据题目类型（选择题或简答题）创建相应的展示。如果用户回答正确，从数据库中删除与该问题相关的错误日志，并显示"回答正确"。如果用户回答错误，将错误信息添加到数据库的错误日志中，并显示"回答错误，答案是：X"，其中X是正确答案。

定义"提交答案"按钮的回调函数submit\_answer()，用于提交用户的答案并显示结果。

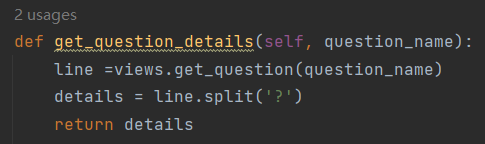
在题目详情窗口中添加展示内容和提交按钮。这个函数中，self.current\_question\_index用于跟踪当前题目的索引，用于显示下一题。



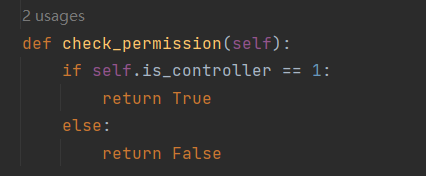
get\_questions\_list(self, question\_group): 该方法用于获取指定问题组的题目列表。它通过调用views.get\_allQu\_inQuSet(question\_group)来从数据库中获取问题组中的题目列表，然后将结果处理成一个问题列表并返回。



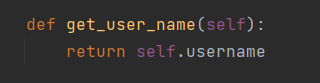
get\_problem\_groups\_list(self): 该方法用于获取当前用户所属的问题组列表。它通过调用views.get\_poverty\_qu\_group(self.username)来从数据库中获取当前用户所属的问题组列表，并返回该列表。



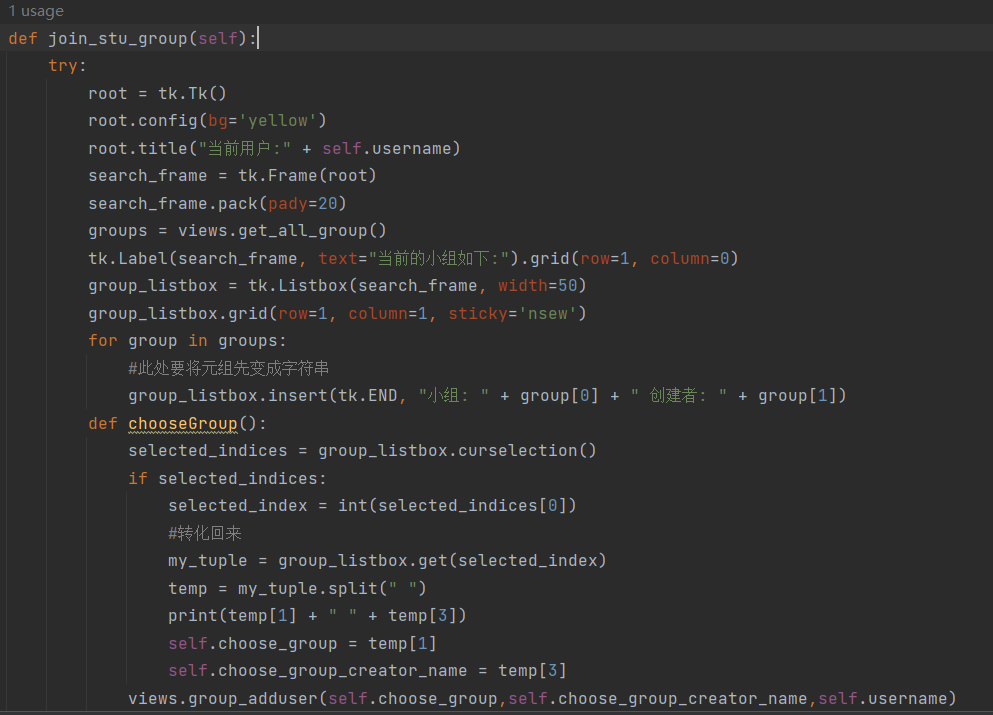
get\_question\_details(self, question\_name): 该方法用于获取指定题目的详细信息。它通过调用views.get\_question(question\_name)来从数据库中获取题目的详细信息，然后将获取的信息拆分为多个字段，最终以列表的形式返回。



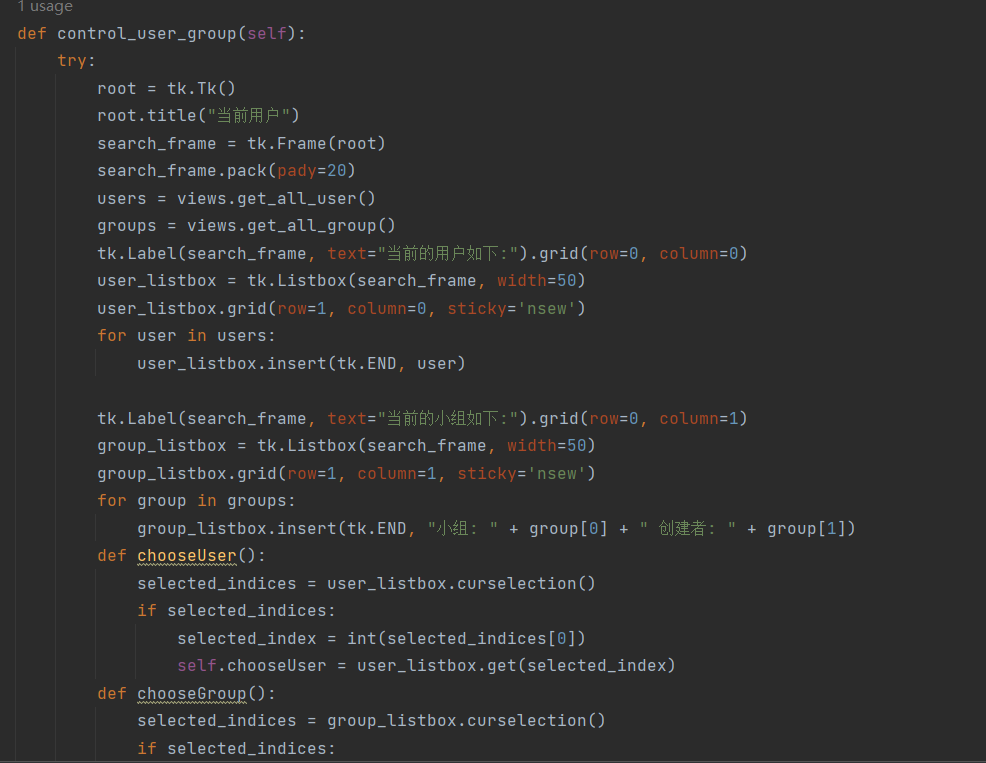
check\_permission(self): 该方法用于检查当前用户是否为管理员（controller）。如果self.is\_controller为1，表示当前用户是管理员，返回True；否则，返回False。



get\_user\_name(self): 该方法用于返回当前用户的用户名（self.username）。



join\_stu\_group(self): 该方法用于让用户加入一个小组。它首先创建一个新的顶层窗口（Tk）用于展示小组列表。然后通过调用views.get\_all\_group()来从数据库中获取所有小组的列表，并将这些小组信息添加到列表框中供用户选择。用户可以在列表中选择一个小组，并点击"选择该小组"按钮来加入选定的小组。所选的小组信息会保存在self.choose\_group和self.choose\_group\_creator\_name中，然后通过调用views.group\_adduser来将当前用户加入选定的小组，并弹出提示信息表示成功加入小组。



control\_user\_group(self): 该方法用于管理用户和小组。首先，它创建了一个新的顶层窗口（Tk）来展示当前的用户和小组列表。通过调用views.get\_all\_user()和views.get\_all\_group()分别从数据库中获取所有用户和小组的列表，并将它们添加到相应的列表框中供用户选择。用户可以选择一个用户和一个小组，并点击"选择该用户"和"选择该小组"按钮来保存所选的用户和小组信息。接着，它调用views.group\_adduser(self.choose\_group, self.choose\_group\_creator\_name, self.chooseUser)函数，将所选用户加入所选小组，并显示成功提示。

创建新的顶层窗口root来展示用户和小组列表。

获取所有用户列表和小组列表，并将它们添加到相应的列表框中。

定义两个回调函数chooseUser()和chooseGroup()，用于获取用户选择的用户和小组信息并保存在self.chooseUser、self.choose\_group和self.choose\_group\_creator\_name中。

定义回调函数add()，首先检查当前用户是否为管理员（通过调用self.check\_permission()），如果是管理员，则调用views.group\_adduser()将所选用户加入所选小组，并弹出成功提示。如果不是管理员，弹出提示信息表示无权限。

创建"选择该用户"按钮、"选择该小组"按钮和"把该用户加入该小组"按钮，并绑定相应的回调函数。



creat\_usr\_group(self): 该方法用于创建小组。首先，它检查当前用户是否为管理员（通过调用self.check\_permission()），如果是管理员，则通过对话框获取用户输入的小组名字，并调用views.create\_group(group\_name, self.username)函数来创建小组。然后根据返回结果弹出相应的提示信息，表示小组创建成功或失败。



用户可以根据指定的主题或者选择获取推荐的题目，然后可以一题一题地解答这些题目，并进行提交答案。

下面是代码的执行逻辑和功能：

用户输入主题：首先，用户会通过对话框输入一个主题。如果输入"all"，则获取随机推荐的20道题目；否则，根据用户之前错误回答的题目记录，获取包含该主题的所有错误题目。

创建题目展示窗口：然后，代码创建一个新的顶层窗口（Toplevel），用于展示个性化推送的题目列表。该窗口设置背景颜色为绿色，并命名为"个性化推送题目"。窗口大小为600x600。

获取题目列表：根据用户的主题选择，获取符合条件的题目列表，并将题目添加到列表框（question\_listbox）中供用户选择。如果用户输入了"all"，则获取随机推荐的题目；否则，根据用户输入的主题，从数据库中获取相应的错误题目，并将它们添加到列表框中。

展示题目详细信息：用户可以在列表框中选择一个题目，并点击"解题"按钮，展示所选题目的详细信息。详细信息包括题目类型、题面和选项（如果是选择题），以及一个文本框用于用户输入答案。

提交答案：用户可以在文本框中输入答案，并点击"提交答案"按钮，程序会根据用户输入的答案判断是否正确，并在题目的详细信息下方显示回答结果。如果答案正确，会在题目下方显示"回答正确"；如果答案错误，会在题目下方显示"回答错误，答案是："，然后显示正确的答案。同时，程序还会根据用户的答题情况更新错误日志，即如果回答正确，则将该题目从错误日志中删除；如果回答错误，则将该题目及用户的答案添加到错误日志中。

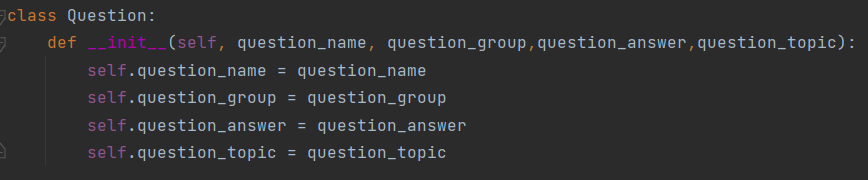
切换下一题：用户可以点击"下一题"按钮，切换到下一道题目，并继续上述步骤。

三.问题的创建：

3.1功能

关于问题，我们定义了三个类，Question类是一个基类，用于表示通用的问题信息。EssayQuestion类和MultipleChoiceQuestion类是继承自Question类的子类，分别用于表示简答题和多项选择题的问题信息。通过使用继承，可以实现代码的重用和扩展性，使得问题类型的定义更加灵活。

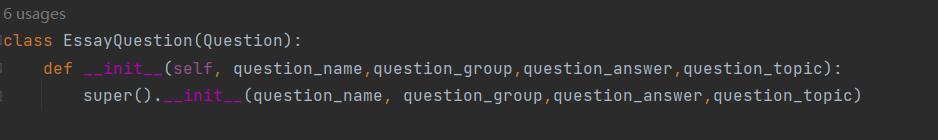
3.2代码分析



Question类：

构造方法\_\_init\_\_(self, question\_name, question\_group, question\_answer, question\_topic)接收问题的名称question\_name、问题组别question\_group、问题答案question\_answer和问题主题question\_topic作为参数。

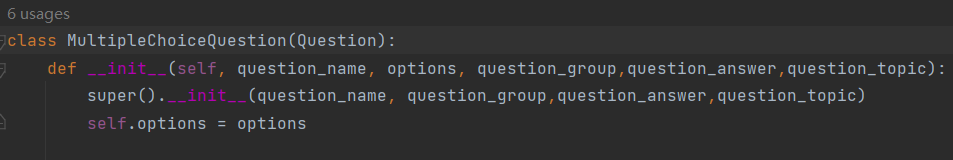
将参数的值分别赋给类的属性self.question\_name、self.question\_group、self.question\_answer和self.question\_topic。



EssayQuestion类：

EssayQuestion类继承自Question类。

构造方法\_\_init\_\_(self, question\_name, question\_group, question\_answer, question\_topic)与Question类的构造方法一样，通过调用super().\_\_init\_\_()来初始化继承自Question类的属性。



MultipleChoiceQuestion类：

MultipleChoiceQuestion类也继承自Question类。

构造方法\_\_init\_\_(self, question\_name, options, question\_group, question\_answer, question\_topic)接收问题的名称question\_name、选项列表options、问题组别question\_group、问题答案question\_answer和问题主题question\_topic作为参数。

通过调用super().\_\_init\_\_()来初始化继承自Question类的属性self.question\_name、self.question\_group、self.question\_answer和self.question\_topic。

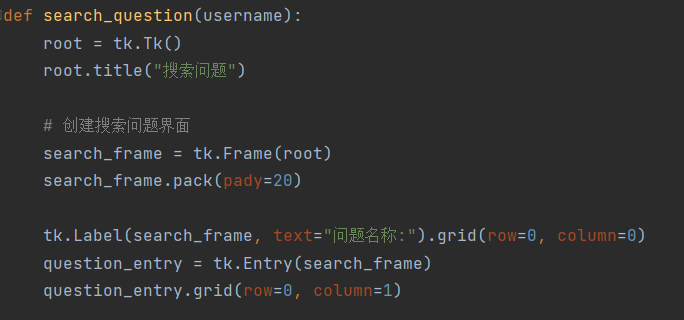
并将options作为新的属性self.options赋值给该类。

四.搜索问题：

4.1功能

我们在该类定义了一个搜索问题的界面，并通过后端对接实现了问题的搜索功能。用户可以输入问题名称，点击"搜索"按钮后，系统将返回与问题名称匹配的问题信息，并在新的弹出窗口中展示结果。搜索功能支持简答题和多项选择题的问题类型，并根据问题名称进行模糊搜索。

4.2代码分析



search\_question(username)函数：

函数接收一个参数username，表示当前用户的用户名。

创建一个新的Tkinter窗口，用于显示搜索问题的界面。

在窗口中创建一个框架search\_frame。

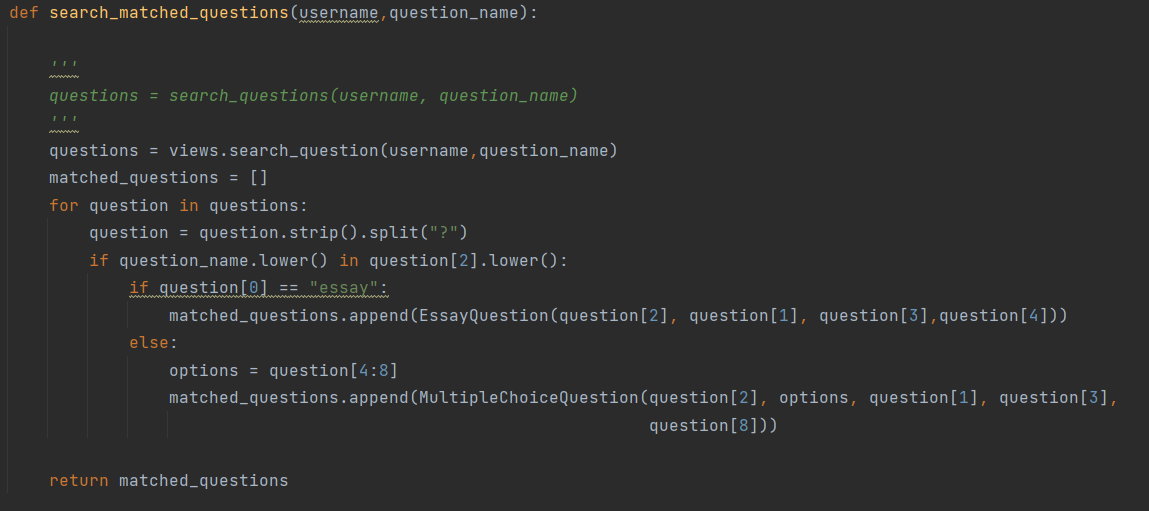
在search\_frame中放置一个标签tk.Label和一个文本输入框tk.Entry，用于输入要搜索的问题名称。

定义内部函数hint()，用于在点击"搜索"按钮时执行搜索操作。

在hint()函数中，获取用户在文本输入框中输入的问题名称，并判断是否为空，如果为空则弹出警告消息框，要求用户输入问题名称。

调用search\_matched\_questions(username, question\_name)函数进行问题搜索，返回匹配的问题列表matched\_questions。

调用display\_matched\_questions(matched\_questions)函数，将匹配的问题显示在新的弹出窗口中。



search\_matched\_questions(username, question\_name)函数：

函数接收两个参数，username表示当前用户的用户名，question\_name表示要搜索的问题名称。

调用views.search\_question(username, question\_name)函数，该函数与后端对接，返回一个包含问题信息的列表questions，其中每个元素为一个问题字符串。

对于每个问题字符串，切割并提取问题的类型、组别、题面、答案和选项等信息。

如果问题的题面包含了输入的问题名称（不区分大小写），则根据问题类型创建相应的EssayQuestion或MultipleChoiceQuestion对象，并将其添加到matched\_questions列表中。

返回matched\_questions列表，其中包含匹配的问题对象。



display\_matched\_questions(matched\_questions)函数：

函数接收一个参数matched\_questions，表示匹配的问题对象列表。

创建一个新的弹出窗口display\_window，用于显示搜索结果。

在display\_window中创建一个文本框text\_box，用于显示问题的信息。

如果matched\_questions不为空，则遍历每个问题对象，根据问题类型显示不同的信息，包括题面、答案或选项、问题组名称等，将问题信息插入到文本框中。

如果matched\_questions为空，则插入"未找到匹配的问题"的提示信息。

将文本框设置为只读状态，禁止用户编辑。

最后，在搜索界面中添加一个"搜索"按钮，并设置其回调函数为hint()，即当用户点击该按钮时，执行搜索操作。

五.搜索小组：

5.1功能

我们定义了一个搜索小组的界面，并通过后端对接实现了小组的搜索功能。用户可以输入小组名称的一部分，点击"搜索"按钮后，系统将返回以该名称为前缀的所有匹配的小组信息，并在新的弹出窗口中展示结果。搜索功能通过调用views.get\_group\_some\_name(stu\_group\_name)函数与后端对接，返回匹配的小组信息列表。

5.2代码分析



search\_stu\_groups()函数：

创建一个新的Tkinter窗口，用于显示搜索小组的界面。

在窗口中创建一个框架search\_frame。

在search\_frame中放置一个标签tk.Label和一个文本输入框tk.Entry，用于输入要搜索的小组名称。

定义内部函数hint()，用于在点击"搜索"按钮时执行搜索操作。

在hint()函数中，获取用户在文本输入框中输入的小组名称，如果未输入则弹出警告消息框，要求用户输入小组名称。

调用views.get\_group\_some\_name(stu\_group\_name)函数进行小组搜索，返回一个包含小组信息的列表matched\_stu\_groups。

调用display\_matched\_groups(matched\_stu\_groups)函数，将匹配的小组信息显示在新的弹出窗口中。



display\_matched\_groups(matched\_stu\_groups)函数：

函数接收一个参数matched\_stu\_groups，表示匹配的小组信息列表，每个元素为一个包含小组名和创建者的元组。

创建一个新的弹出窗口display\_window，用于显示搜索结果。

在display\_window中创建一个文本框text\_box，用于显示匹配的小组信息。

如果matched\_stu\_groups不为空，则遍历每个小组信息元组，将小组名和创建者信息插入到文本框中。

如果matched\_stu\_groups为空，则插入"未找到匹配的小组"的提示信息。

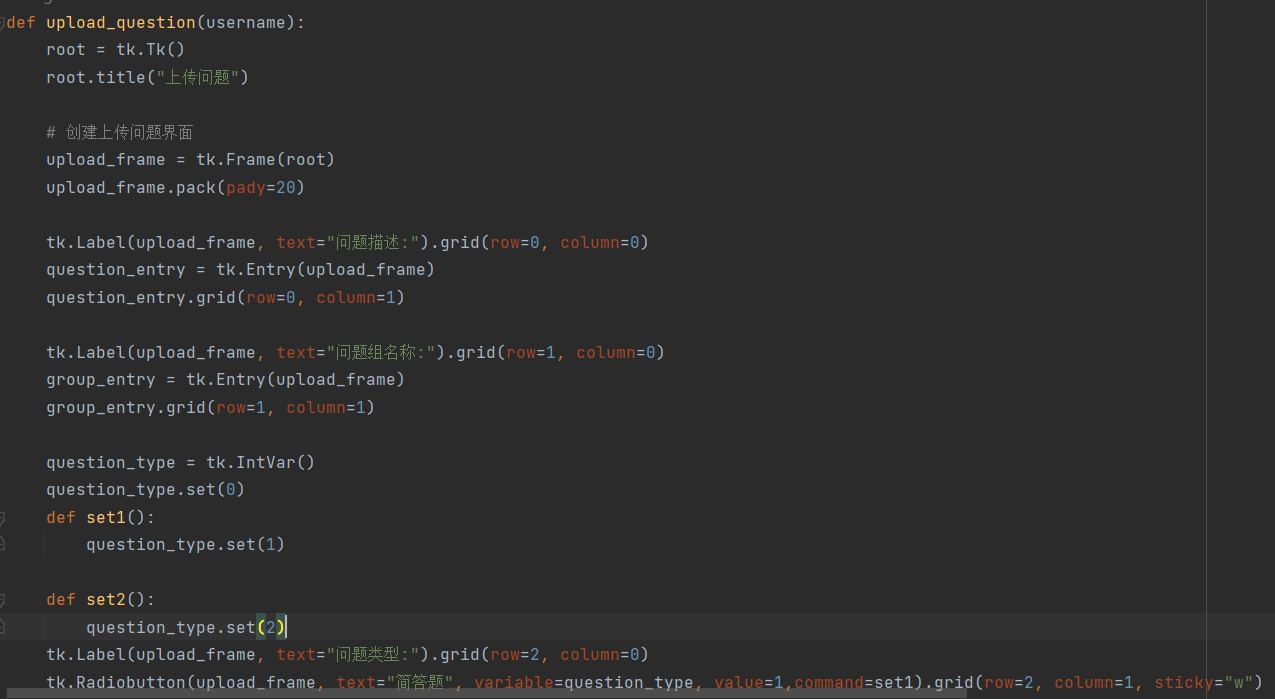
将文本框设置为只读状态，禁止用户编辑。

最后，在搜索界面中添加一个"搜索"按钮，并设置其回调函数为hint()，即当用户点击该按钮时，执行搜索小组的操作。

六.上传问题：

6.1功能

我们定义了一个上传问题的界面，用户可以通过文本输入框上传单个问题，或通过浏览文件选择问题文件进行批量上传。在上传问题时，用户需要输入问题描述、问题组名称和选择问题类型。如果问题组不存在，则需要输入问题组可见的小组和组长名，并调用creat\_pro\_group()函数创建新的问题组。上传成功后，系统会弹出消息框显示问题上传成功，并关闭窗口。

6.2代码分析  


upload\_question(username)函数：

创建一个新的Tkinter窗口，用于显示上传问题的界面。

在窗口中创建一个框架upload\_frame。

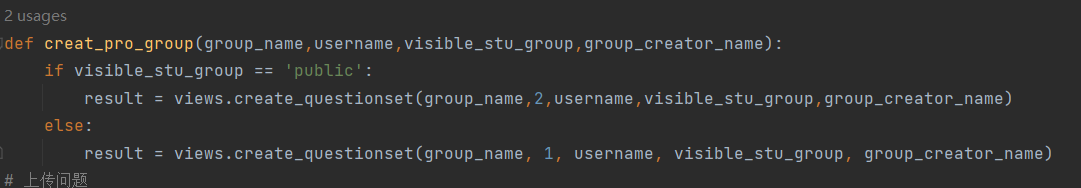
在upload\_frame中放置两个标签tk.Label和两个文本输入框tk.Entry，用于输入问题描述和问题组名称。

在upload\_frame中放置两个单选按钮tk.Radiobutton，用于选择问题类型（简答题或选择题）。

定义内部函数single\_upload()和browse\_file()，分别用于上传单个问题和通过文件上传多个问题。

在single\_upload()函数中，获取用户输入的问题描述和问题组名称，并获取用户选择的问题类型。

如果问题组不存在，则弹出输入框要求用户输入该问题组可见的小组和组长名，并调用creat\_pro\_group()函数创建新的问题组。



根据问题类型和用户输入的信息调用views.create\_question()函数上传问题到后端。

弹出消息框显示问题上传成功，并关闭窗口。

在browse\_file()函数中，弹出文件选择对话框，让用户选择问题文件。

调用process\_file(file\_path)函数处理选择的问题文件。

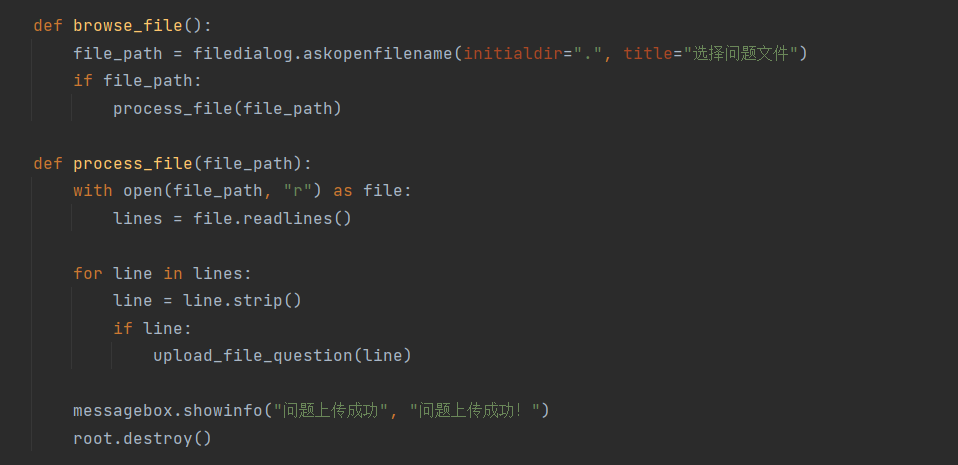
在process\_file(file\_path)函数中，打开问题文件，逐行读取问题信息，并调用upload\_file\_question(line)函数上传每个问题。

在upload\_file\_question(line)函数中，解析每行问题信息并调用views.create\_question()函数上传问题到后端。

如果问题类型无效，则弹出警告消息框。

上传成功后，弹出消息框显示问题上传成功，并关闭窗口。

最后，在上传问题界面中添加一个"上传"按钮和一个"浏览文件"按钮，并分别设置它们的回调函数为single\_upload()和browse\_file()，即当用户点击这两个按钮时，执行上传问题或浏览文件的操作。

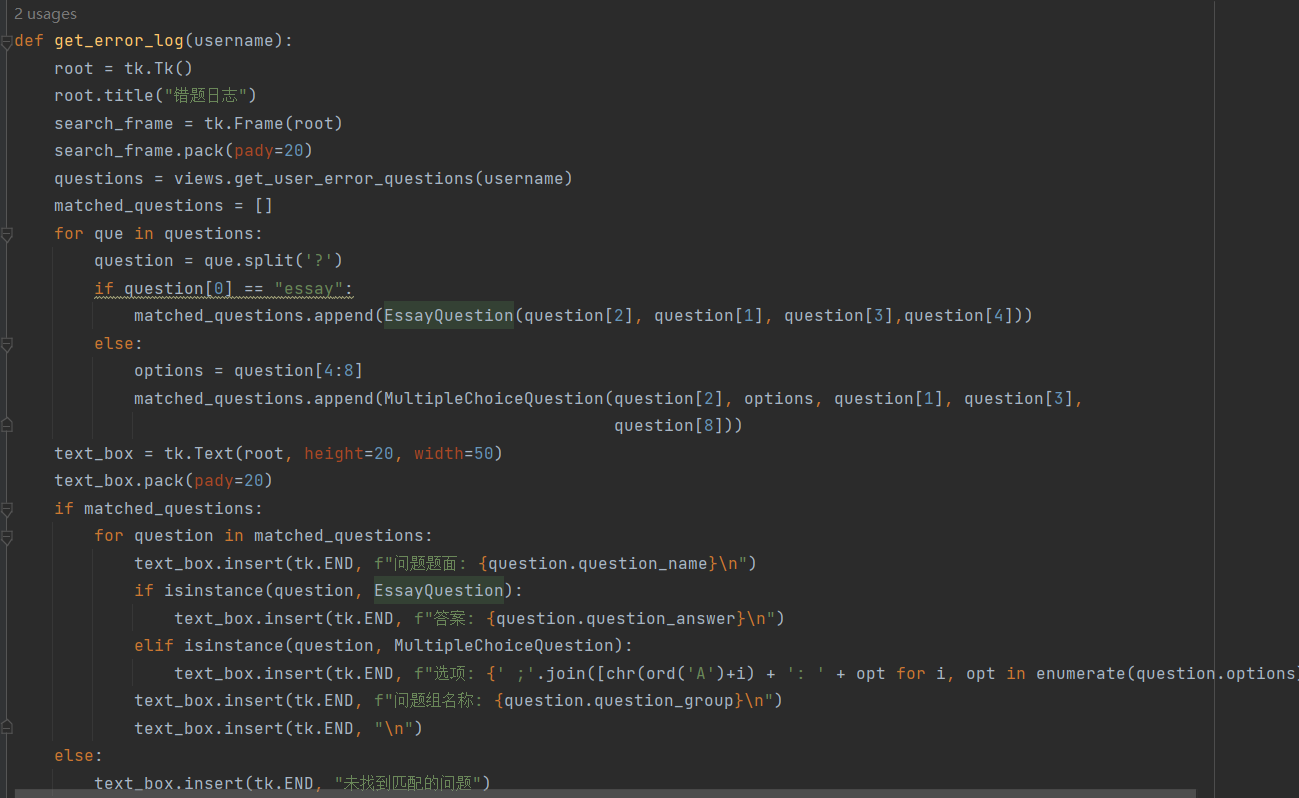


七.错误日志：

7.1功能

我们定义了一个错题日志界面，根据指定用户的错题日志显示题面、答案或选项以及问题组名称等信息。用户可以通过界面查看自己的错题记录。目前代码中的pass占位符尚未实现具体的功能，因此用户无法进行更多的操作。如果需要进一步扩展功能，可以在这个函数中添加相关的逻辑和操作。

7.2代码分析



get\_error\_log(username)函数：

创建一个新的Tkinter窗口，用于显示错题日志界面。

在窗口中创建一个框架search\_frame。

调用views.get\_user\_error\_questions(username)函数获取指定用户的错题日志。

将错题日志解析为EssayQuestion和MultipleChoiceQuestion对象，并将它们添加到matched\_questions列表中。

在窗口中创建一个文本框text\_box，用于显示错题日志的内容。

如果matched\_questions列表不为空，则逐个遍历其中的问题对象，并将题面、答案或选项和问题组名称等信息插入到text\_box中。

如果matched\_questions列表为空，则在text\_box中插入未找到匹配问题的提示信息。

将text\_box设置为只读状态，以禁止用户编辑内容。

最后，函数体内有一个pass占位符，暂时没有实现任何功能。

**（二）后端部分**

后端我们选用django框架进行搭建。在生成django框架，配置完settings.py并生成app文件后，我们在models.py中定义User，Question等多个模型，在views.py中定义多个功能函数，为前端的函数调用提供接口。

**模型**

以下是models.py中定义的模型内容，每一个模型对应着一个数据库表。

1. **Question**（问题）模型：
   * **content**（题目内容）：使用**TextField** 类型，用于存储题目的内容。
   * **topic**（题目主题）：使用**CharField** 类型，最大长度为100，用于存储题目的主题。
   * **correct\_answer**（正确答案）：使用**TextField** 类型，用于存储问题的正确答案。
   * **question\_type**（题目类型）：使用**CharField** 类型，最大长度为100，表示题目类型，一共有'multiple\_choice'(选择题)和'essay'(填空题)两种类型。
2. **ErrorLog**（错误日志）模型，表示**每道题**的错误日志：
   * **question**（问题）：使用**ForeignKey** 类型，指向 **Question** 模型，表示与该错误日志相关联的问题。**on\_delete=models.SET\_NULL** 表示当关联的问题被删除时，该错误日志的问题字段将被设置为 NULL。
   * **wrong\_answer**（错误答案）：使用**TextField** 类型，用于存储用户的错误答案。
   * **error\_frequency**（错频）：使用**IntegerField** 类型，表示该错误日志被记录的次数，默认值为0。
3. **ErrorLogs\_Topic**（以主题为索引的错误日志）模型，表示一个人的**同一个主题**的所有题目的错误日志：
   * **topic**（主题）：使用**CharField** 类型，最大长度为100，用于存储错误日志的主题。
   * **error\_Logs**（错误日志）：使用**ManyToManyField** 类型，指向 **ErrorLog** 模型，表示与该主题相关联的错误日志。
   * **number**（总共有多少个错题）：使用**IntegerField** 类型，表示与该主题相关的错误日志数量，默认值为0。
4. **User**（用户）模型：
   * **username**（用户名）：使用**CharField** 类型，最大长度为100，用于存储用户的用户名。
   * **password**（密码）：使用**CharField** 类型，最大长度为100，用于存储用户的密码。
   * **user\_type**（用户类型）：使用**IntegerField** 类型，表示用户的类型。
   * **error\_logs**（错误日志）：使用**ManyToManyField** 类型，指向 **ErrorLogs\_Topic** 模型，表示与该用户相关联的所有以主题为索引的错误日志。
5. **Group**（用户组）模型：
   * **name**（组名）：使用**CharField** 类型，最大长度为100，用于存储用户组的名称。
   * **creator**（创建者）：使用**ForeignKey** 类型，指向 **User** 模型，表示该用户组的创建者。**on\_delete=models.CASCADE** 表示当关联的用户被删除时，该用户组也将被删除。
   * **users**（用户成员）：使用**ManyToManyField** 类型，指向 **User** 模型，表示该用户组的成员。
6. **QuestionSet**（以创建者为索引的问题集合）模型：
   * **name**（问题集合名称）：使用**CharField** 类型，最大长度为100，用于存储问题集合的名称。
   * **creator**（创建者）：使用**CharField** 类型，最大长度为100，用于存储问题集合的创建者。
   * **permission**（访问权限）：使用**IntegerField** 类型，表示问题集合的访问权限，共有0（Private），1（Group），2（Public）三种情况。
   * **group**（组名称）：使用**CharField** 类型，最大长度为100，如果访问权限是GROUP，则用于存储组名称。
   * **group\_creator\_name**（用户组的创建者的名字）：使用**CharField** 类型，最大长度为100，用于存储用户组的创建者的名字。
   * **questions**（所有问题）：使用**ManyToManyField** 类型，指向 **Question** 模型，表示该问题集合包含的所有问题。
7. **TopicSet**（以主题为索引的问题集合）模型，用于推荐错题：
   * **topic**（主题）：使用**CharField** 类型，最大长度为100，用于存储主题集合的主题。
   * **questions**（所有问题）：使用**ManyToManyField** 类型，指向 **Question** 模型，表示与该主题相关联的所有问题。

**方法函数**

以下是views.py中的方法函数，用于对模型中的数据进行处理。

1. **add\_error\_log(username, content, wrong\_answer)**：

* 函数功能：该函数用于向用户的错误日志中添加错误记录。
* 函数参数：
  + **username**：用户名，表示要添加错误记录的用户。
  + **content**：题目内容，表示用户做错的题目内容。
  + **wrong\_answer**：错误答案，表示用户错误的答案。
* 函数实现：该函数首先通过题目内容找到对应的问题（**Question**），再根据用户名找到对应的用户（**User**）。然后它尝试查找与问题主题相关的错误日志主题（**ErrorLogs\_Topic**）。如果不存在相关的错误日志主题，则新建一个。接下来，它查找与指定问题相关的错误日志（**ErrorLog**）。如果已存在相关的错误日志，则更新错误答案和错误频率。如果不存在相关的错误日志，则新建一个，并将其添加到错误日志主题中。最后，将错误日志主题添加到用户的错误日志中。

1. **add\_question\_to\_topicset(question, topic)**：

* 函数功能：该函数用于将题目添加到指定主题的题目集合（**TopicSet**）中。
* 函数参数：
  + **question**：题目实例，表示要添加的题目。
  + **topic**：主题名称，表示题目所属的主题。实现：该函数尝试查找与指定主题相关的题目集合（**TopicSet**）。如果存在相关的题目集合，则将题目添加到集合中。如果不存在相关的题目集合，则新建一个题目集合，并将题目添加到集合中。
* 函数实现：该函数尝试查找与指定主题相关的题目集合（**TopicSet**）。如果存在相关的题目集合，则将题目添加到集合中。如果不存在相关的题目集合，则新建一个题目集合，并将题目添加到集合中。

1. **remove\_error\_log(username, topic, content)**：

* 函数功能：该函数用于从用户的错误日志中删除指定主题的指定题目的错误记录。
* 参数：
  + **username**：用户名，表示要删除错误记录的用户。
  + **topic**：主题名称，表示要删除的错误日志主题。
  + **content**：题目内容，表示要删除的题目内容。
* 实现：该函数首先根据用户名找到对应的用户（**User**）。然后在用户的错误日志中查找与指定主题相关的错误日志主题（**ErrorLogs\_Topic**）。接着，在错误日志主题中找到与指定内容相关的错误日志（**ErrorLog**），然后将其删除。最后，更新错误日志主题中的错题数量。

1. **get\_user\_error\_questions\_by\_topic(username, topic)**：

* 函数功能：该函数用于获取用户在指定主题下的错误题目列表。
* 函数参数：
  + **username**：用户名，表示要获取错误题目的用户。
  + **topic**：主题名称，表示要获取的错误日志主题。
* 函数实现：该函数首先通过用户名找到对应的用户（**User**）。然后查找与指定主题相关的错误日志主题（**ErrorLogs\_Topic**）。接下来，获取每个错误日志主题中与问题相关的错误日志列表，然后将错误日志列表中的问题（**Question**）放在一个列表中，并返回该列表。

1. **recommend\_questions\_by\_topic\_ratio(username, num)**：

* 函数功能：该函数用于根据用户在不同主题下的错误题目比例，推荐一定数量的题目给用户。
* 函数参数：
  + **username**：用户名，表示要获取推荐题目的用户。
  + **num**：推荐题目的数量。
* 函数实现：该函数首先通过用户名找到对应的用户（**User**）。然后获取用户错题日志中每个主题的错题数，并计算每个主题的错题比例。接下来，根据主题的错题比例从题目集合中随机抽取一定数量的题目，并返回这些题目。

1. **get\_user\_error\_questions(username)**：

* 函数功能：该函数用于获取用户的所有错误题目列表。
* 函数参数：
  + **username**：用户名，表示要获取错误题目的用户。
* 函数实现：该函数首先通过用户名找到对应的用户（**User**）。然后查找用户的所有错题日志主题，并获取每个错误日志主题中的所有错题。接着，将所有的错误题目放在一个列表中，并返回该列表。

1. **search\_question\_by\_topic(username, topic)**：

* 函数功能：该函数用于在指定主题下搜索用户能看到的题目列表。
* 函数参数：
  + **username**：用户名，表示要进行题目搜索的用户。
  + **topic**：主题名称，表示要搜索的题目所属的主题。
* 函数实现：该函数根据指定主题从题目集合中搜索题目，并通过**have\_poverty**函数检查用户是否有权限查看该题目。如果用户有权限，则将题目添加到列表中，并返回该列表。

1. **delete\_questions\_by\_sensitive\_word(sensitive\_word)**：

* 函数功能：该函数用于删除所有题目内容和正确答案中含有敏感词的问题。
* 函数参数：
  + **sensitive\_word**：敏感词，表示要删除含有该敏感词的题目。
* 函数实现：该函数首先获取所有的问题，然后遍历所有问题，检查题目内容和正确答案中是否含有敏感词。如果含有敏感词，则将该问题删除。

1. **get\_question**:

* 函数功能：获得指定题面的题目，使用自定义格式返回题目
* 函数参数：
  + **question\_name**表示被搜索的题目的题面
* 函数实现：通过Django的get方法，以question\_name为参数，获得该题目的实例；通过show\_question方法，获得该题目的自定义的标准形式

1. **get\_user\_username**：

* 函数功能：获得某用户的模型实例
* 函数参数：
  + **username**表示该模型实例的用户名
* 函数实现：通过Django的get方法，以username为参数获得相应的模型实例

1. **creat\_user\_with\_errorlog**:

* 函数功能：创建一个用户
* 函数参数：
  + **username**表示待创建用户的用户名
  + **password**表示待创建用户的密码、
* 函数实现：通过Django的get方法，判断是否已经有用户使用了该用户名；通过User模型的create方法创建一个用户实例，并保存到数据库中。

1. **Get\_all\_user**:

* 函数功能：获取所有非管理员的用户的名字
* 函数实现：通过User的all方法遍历所有User实例

1. **Login**：

* 函数功能：判断数据库中是否存在该用户，并进行登陆操作
* 函数参数：
  + **username**表示用户名
  + **password**表示密码
* 函数实现：通过User的get方法，以用户名为参数找到此用户名对应的User实例；判断传入的password与得到的User实例的password是否相同。

1. **change\_password**：

* 函数功能：修改对应用户的密码
* 函数参数：
  + **username**表示用户名
  + **password**表示密码
* 函数实现：通过User的get方法，以用户名为参数找到此用户名对应的User实例；将得到的User实例的password修改为传入的password

1. **create\_group**：

* 函数功能：创建一个用户组
* 函数参数：
  + **name**表示小组名
  + **creator\_username**表示该组的创建者的用户名
* 函数实现：使用Group的get方法判断是否已存在该小组；使用Group的creat方法创建该小组

1. **get\_all\_group**：

* 函数功能：获取所有用户组的信息
* 函数实现：通过Group的all方法遍历所有Group实例

1. **group\_adduser**：

* 函数功能：将用户加入到某用户组中
* 函数参数：
  + **name**表示用户组名
  + **creator\_user\_name**表示该用户组的创建者的名字
  + **username**表示待添加的用户名
* 函数实现：使用Group的get方法通过name，creator\_user\_name获得该Group的实例；使用User的get方法通过username获得用户实例；使用Group.users的add方法将用户实例添加进去

1. **create\_question**：

* 函数功能：创建一个问题
* 函数参数：
  + **username**表示创建该题目的用户名
  + **question\_type**表示该题目的类型
  + **question\_groupname**表示该题目所在的问题组
  + **content**表示题目的题面
  + **correct\_answer**表示题目的正确答案
  + **sel1,sel2,sel3,sel4**分别表示选择题的四个选项的内容
  + **topic**表示题目的主题
* 函数实现：通过Question的get方法判断是否已存在题面相同的题目；通过自定义的get\_questionset函数和have\_poverty\_qu\_set函数，分别判断是否存在该问题组和用户是否有权限将问题添加到该问题组；通过Question的creat方法创建该问题实例；通过Questionset方法获得该问题组的实例；通过Questionset的questions的add方法将创建的问题实例添加进去

1. **create\_questionset**：

* 函数功能：创建一个问题组
* 函数参数：
  + **name**表示该问题组的名字
  + **permission**表示该问题组的共享范围
  + **creator\_name**表示创建者的名字
  + **group**表示与该问题组关联的用户组的名字
  + **group\_creator\_name**表示与该问题组关联的用户组的创建者的名字
* 函数实现：通过Questionset的get方法判断是否已经存在该问题组；通过Group和User的get方法分别判断是否存在用户组和用户；通过Questionset的create方法创建一个对应的questionset实例

1. **get\_questionset**：

* 函数功能：获得问题组实例
* 函数参数：
  + **name**表示问题组的名字
* 函数实现： 通过Questionset的get方法，以name为参数获得问题组实例

1. **get\_group**：

* 函数功能：获得用户组实例
* 函数参数：
  + **name**表示用户组的名字
  + **creator\_name**表示用户组的创建者的名字
* 函数实现：通过Group的get方法，以name和creator\_name获得对应的用户组实例

1. **have\_poverty**：

* 函数功能：判断某用户是否有权限回答某个问题
* 函数参数：
  + **username**表示答题的用户名
  + **question1\_instance**表示该问题的模型实例
* 函数实现：通过自定的get\_user\_username函数获得该用户的User实例；通过Question模型中Questions\_set的all方法获得该问题所在的问题组；通过查看问题组的公开范围分类讨论；通过if user in relate\_group.users.all()判断用户是否在问题组关联的用户组中

1. **have\_poverty\_qu\_set**：

* 函数功能：查看一个用户是否有资格向一个问题组上传问题
* 函数参数：
  + **username**表示用户名
  + **questionset1**表示该问题组的模型实例
* 函数实现：通过查看问题组的公开范围分类讨论；通过questionset中保存的关联的用户组的信息和Group的get方法获得与之关联的用户组实例；通过if user in relate\_group.users.all()判断用户是否在问题组关联的用户组中

1. **show\_question**：

* 函数功能：返回一个问题的规范形式
* 函数参数：
  + **question**表示该问题的模型实例
* 函数实现：将question中的每个字段和相应的衔接符拼接起来

1. **get\_group\_some\_name**：

* 函数功能：搜索与参数有关联的小组
* 函数参数：
  + **someName**表示与用户组名有关的字符串
* 函数实现：通过Group的all方法获得Group的所有模型实例；通过in字段判断someName是否为Group中name的子串

1. **search\_question**：

* 函数功能：获取所有用户有权限且与参数关联的问题
* 函数参数：
  + **username**表示用户名
  + **someStr**表示与题面关联的部分
* 函数实现：通过Question的all方法获得所有问题实例；通过User的get方法获得用户实例；遍历所有的Question实例通过自定义的have\_poverty判断用户是否有权限回答该问题，通过in字段判断someStr是否是题面的子串

1. **get\_poverty\_qu\_group**：

* 函数功能：得到用户能回答的所有问题组
* 函数参数：
  + **username**表示用户名
* 函数实现：通过Questionset的all方法获得所有问题组实例；通过User的get方法获得用户实例；遍历所有问题组实例调用自定义的have\_poverty\_qu\_set函数。

1. **get\_allQu\_inQuSet**：

* 函数功能：获得一个问题组的所有问题
* 函数参数：
  + **quSet\_name**表示该问题组的名字
* 函数实现：通过Questionset的get方法获得该问题组实例；通过questionset.questions.all获得问题组里的所有问题实例

1. **remove\_user\_from\_group**：

* 函数功能：将一个用户从用户组中删除
* 函数参数：
  + **group\_name**表示用户组名
  + **group\_creator\_name**表示用户组创建者的名字
  + **username**表示用户名
* 函数实现：通过Group的get方法获得用户组实例；通过User的get方法获得用户实例；通过group.users的remove方法移除该用户

1. **get\_user\_in\_group**：

* 函数功能：获得一个用户所在的所有小组
* 函数参数：
  + **username**表示用户名
* 函数实现：通过Group的all方法获得所有用户组实例；通过User的get方法获得用户实例；遍历用户组实例通过in字段判断用户是否在group.users中

四、项目运行过程

需要安装django、tkinter、PIL库。

在运行前需要将untitled4文件夹中的每一个文件头中的sys.path.append("路径")中的路径修改为自己电脑下载项目后UserAquestion的路径名。

在main.py中进行运行即可正常运行项目。

五、项目总结

项目总体完成情况较为理想，完成了小组成员的预期实现效果。

六、主要参考资料

ChatGPT，B站中的Django网课：“2022 B站最详细django3教程(django从入门到实践)”——银角大王-武沛齐-真身