**Python实验一**

E51814014 吴振龙

1、

使用列表存储Fibonacci数列，首先初始化数列的前两项，根据规律生成数列的第2到100项：

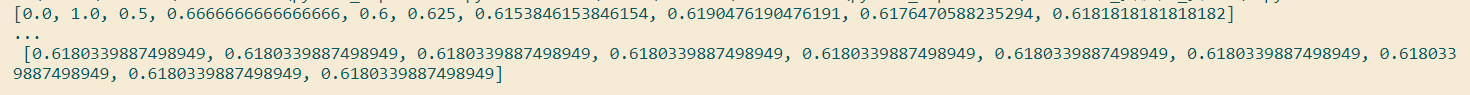
|  |
| --- |
| *f = [0, 1]              # Fibonacci数列*  *for i in range(2, 100):*  *f.append(f[i - 1] + f[i - 2])* |

使用列表推导式生成数列的前项比后项的值：

|  |
| --- |
| *ratio = [f[i] / f[i + 1] for i in range(len(f) - 1)]              # 数列的前项比后项* |

输出前十项和后十项结果：

|  |
| --- |
| *print(ratio[:10], '\n...\n', ratio[-10:])* |



可以看出，结果Fibonacci数列的前项比后项的值趋近于0.618。

2、

使用pyqt5设计gui界面。

首先设计Manager类:

管理所有记录数据，类中有两个属性，分别是保存文件的路径file\_path，信息列表infos\_list。信息列表与本地文件保持同步。

类中有四个方法，分别是

|  |  |  |
| --- | --- | --- |
| 定义 | 参数 | 作用 |
| read\_file(self) | 无 | 从文件中读取信息列表保存到infos\_list中 |
| save\_file(self) | 无 | 将infos\_list中的内容保存到文件中，每次对infos\_list进行修改时都执行此方法用来保持infos\_list的同步 |
| add\_item(self, item) | item: <list>长度为8的列表，列表中的每个元素是一个字符串，表示一条记录的某个信息 | 将item中的信息添加到infos\_list中 |
| del\_item(self, num) | num: <int>要删除的记录所在infos\_list的下标 | 将infos\_list中下标为num的项目删除 |

注：存储数据选择使用写入文件方式，文件选择csv格式。

设计gui主界面：

使用QtDesigner设计如下界面：



主界面中主要包括以下控件：

|  |  |
| --- | --- |
| 控件名 | 作用 |
| tw\_display | 显示目前所有的记录信息 |
| btn\_add | 添加信息按钮，点击后弹出一个页面，供用户输入信息 |
| btn\_delete | 删除一条记录，使用方法为在表格中点击一个想要删除的记录，再点击删除按钮即可。 |
| btn\_close | 关闭窗口按钮，点击可关闭窗口 |
| line\_search | 搜索输入框，当输入的字符串在表格中有匹配内容时，会高亮显示匹配到的内容 |

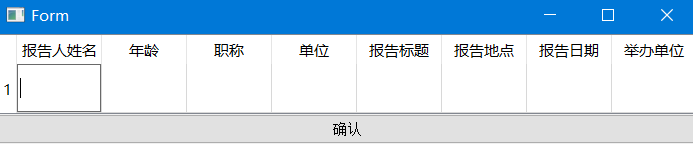
除了使用Designer可视化设计的界面以外，在代码中还使用了以下语句修改主界面：

|  |
| --- |
| *# 水平表头居中*  *self*.tw\_display.horizontalHeader().setSectionResizeMode(QHeaderView.Stretch)  *# 垂直表头居中*  *self*.tw\_display.verticalHeader().setSectionResizeMode(QHeaderView.Stretch) |

设计输入信息页面：

使用QtDesigner设计如下界面：

（界面对象名为ui\_input）



除了使用**Designer**可视化设计的界面以外，在代码中还使用了以下语句修改主界面：

|  |
| --- |
| *# 设置输入窗口的水平表头填充*  *self*.ui\_input.tw\_input.horizontalHeader().setSectionResizeMode(QHeaderView.Stretch)  *# 设置输入窗口的垂直表头填充*  *self*.ui\_input.tw\_input.verticalHeader().setSectionResizeMode(QHeaderView.Stretch) |

显而易见，改界面可以支持用户输入记录的各种信息，点击确认按钮后将记录添加到本地文件中。

功能实现：

PyQt主窗口使用类**Interface**进行组织，类初始化时会新建一个**Manager**对象，用于管理所有的记录信息。初始化时还会调用一个**display\_items**方法，将**Manager**对象中**infos\_list**中保存的所有信息显示在主界面的表格中。

1、添加信息：

在用户点击添加信息按钮后，弹出添加信息窗口，输入记录的各种信息后，点击确认按钮，确认按钮的点击信号使用**connect**方法连接**Interface**类的**add\_item**方法，程序的后端会获取输入的所有信息，输入为空的位置使用空字符串替代，之后将信息列表传给**Manager**对象的**add\_item**方法，可以实现添加信息的功能。如果用户没有输入数据却点击了确认按钮，程序会提示信息而且不进行添加操作。

|  |
| --- |
| **def** add\_item(**self**):          ''' 添加一条项目 '''          item **=** []  **for** col **in** range(*self*.ui\_input.tw\_input.columnCount()):  **try**:                  item.append(*self*.ui\_input.tw\_input.item(0, col).text())  **except** AttributeError:  *# 发生此错误说明输入的第col列没有值，则将该分量置为空*                  item.append("")  **if** item **==** ["","","","","","","",""]:  *# 如果输入的信息全为空*              QMessageBox.information(None, "提示", "输入信息为空")  **return** ""  *self*.manager.add\_item(item)  *self*.ui\_input.tw\_input.clearContents()          *# 将输入窗口清空*  *self*.display\_items()          QMessageBox.information(None, "提示", "添加信息成功") |

2、删除

使用删除功能需要用户先使用鼠标点击表格中的一行，程序的后端通过连接表格的itemClicked信号与自定义的slot\_get\_widget\_selected\_row槽函数可以获取点击表格的行号，并保存到类属性selectedRow中，当用户点击删除按钮时，PyQt发送btn\_delete.clicked信号，后端通过connect连接该信号与自定义的delete\_item方法，此时会调用delete\_item方法，delete\_item方法会调用Manager类的del\_item方法，并将selectedRow值传递进去，del\_item方法会根据行号selectedRow从infos\_list中删除相应的项目，并更新本地文件，最后delete\_item方法调用display\_items方法更新表格的显示的信息。如果用户点击删除按钮之前没有先选中表格中的某一行，slot\_get\_widget\_selected\_row函数会将-1赋值给类属性selectedRow，在调用display\_items方法时，如果selectedRow的值为-1，会弹出提示信息窗口，并不会做任何事。

|  |
| --- |
| **def** delete\_item(**self**):          ''' 根据display表格中选中的行，删除一条记录 '''  **if** *self*.selectedRow **==** **-**1:              QMessageBox.information(MainWindow, "提示", "请在表格中选择你要删除的记录")  **return**  *self*.manager.del\_item(*self*.selectedRow)  *self*.display\_items()          QMessageBox.information(MainWindow, "提示", "记录已删除") |

3、搜索

使用搜索功能时需要用户在搜索框中直接输入想要搜索的内容，搜索框是一个lineEdit类控件，当搜索框中的内容发生改变时，前端会发送textChanged(**string**)信号，该信号的参数是此时输入框中的字符串，在后端使用connect函数将该信号连接到自定义的槽函数slot\_search(**self, search\_string**)，该槽函数接收一个字符串参数，而connect函数会自动将textChanged(**string**)信号中的参数string传递给slot\_search(**self**, **search\_string**)方法。在slot\_search方法中，使用一个列表推导式，调用Qt中QTableWidget类的findItems方法查找表格中匹配字符串string的item，列表推导式的作用是将所有匹配到的item设为选中状态，即实现高亮显示（注意，在slot\_search方法中首先要将表格中的所有选中项目取消），选中状态的样式可以在Designer中进行可视化设置。

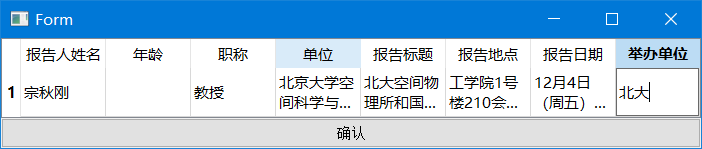
|  |
| --- |
| **def** slot\_search(**self**, **search\_string**):          ''' 搜索功能 '''  *self*.tw\_display.setCurrentItem(None)  **if** search\_string **==** "":  **return**          [item.setSelected(True) **for** item **in** *self*.tw\_display.findItems(search\_string, QtCore.Qt.MatchContains)] |

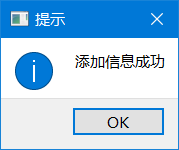
程序演示：

1、运行程序



2、添加信息





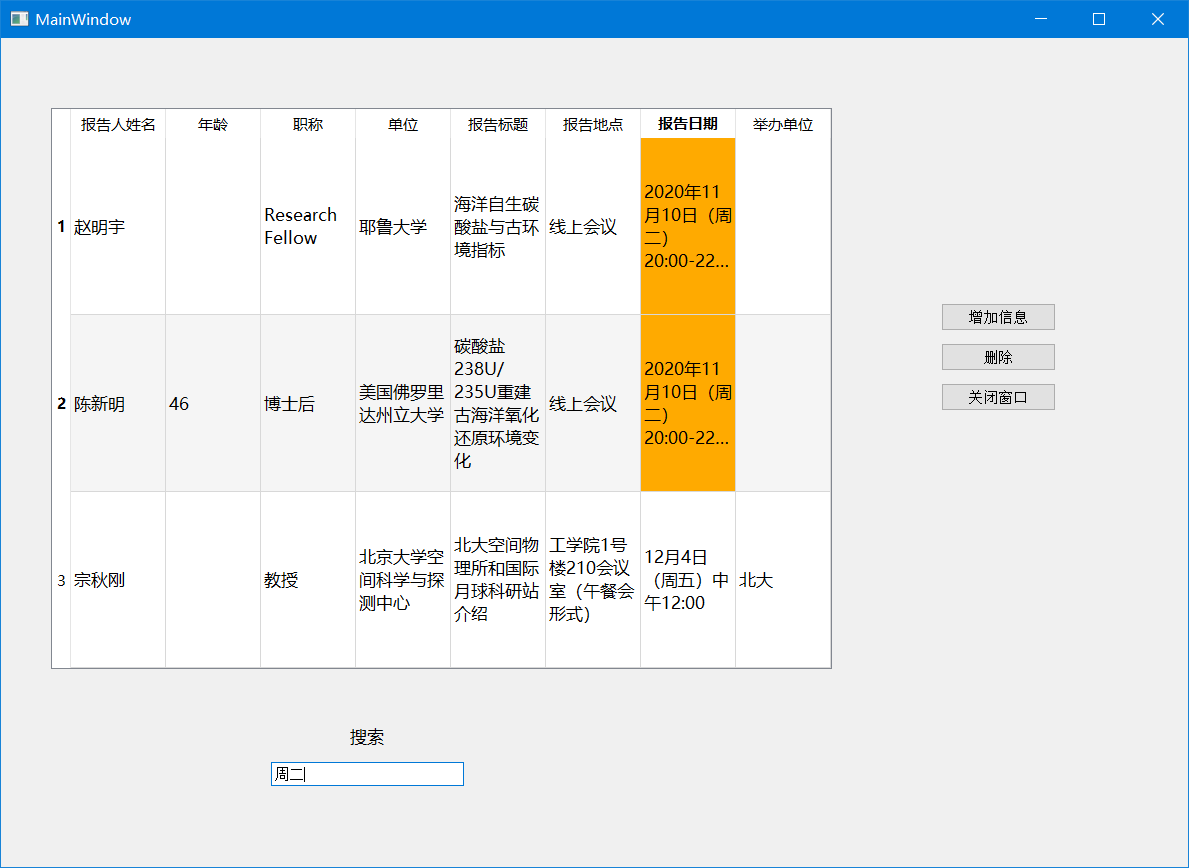


3、删除





4、搜索







【总结】

第二个实验中还有很多可以优化的地方，比如对界面的美化，还有增强程序的健壮性，当用户没有按照意愿操作时，提示更多的信息。虽然很想做，但介于课业紧张，只好停留在现在的程度。