**学号： 2020330301112 姓名： 彭薪竹**

一：实验内容：

1.请大家登录acm.zstu.edu.cn上的Python程序设计高阶——课堂练习3，在1小时内完成；

2.构建一个电梯类，需要满足以下要求：

1)电梯都有品牌，额定载重和额定人数；

2)每一个电梯都有一个操作面板，上面有按钮，参考实际的电梯的按钮功能来设计你的按钮个数和对应的功能（**注意，按钮应该也是一个类**）；

3)电梯有上升下降功能；

4)新安装一个电梯时，需要设定这样一些参数：楼层数，额定载重和额定人数；

5)现在，一幢新的大楼即将落成，客户购买了一部上海三菱的电梯，进行安装并使用，请你用Python编写一个程序来模拟这个电梯的运行，人员的出现、数量等，请使用随机函数。

注意：电梯运行的模拟根据模拟的界面的美观性，酌情给分！

1. 程序代码：
2. **import** sys
3. **import** os
4. **import** random
5. **from** PyQt5 **import** QtCore, QtWidgets, QtGui
6. **from** PyQt5.QtWidgets **import** \*
7. **from** PyQt5.QtCore **import** \*
8. **from** PyQt5.QtGui **import** \*
10. **class** Elevator(QWidget):
11. aimFloor = 1
12. floor = 1  # 电梯当前楼层,初试电梯在1楼
13. prefloor = 1  # 表示电梯上下初始运行所处的楼层
14. status = 0  # 电梯当前状态：0表示暂停，1表示向上，-1表示向下
15. preStatus = 1  # 电梯前一状态，1表示向上，-1表示向下
16. door = 1  # 0表示打开，1表示关闭
17. upfloorList = []  # 电梯上行楼层
18. downfloorList = []  # 电梯下行楼层
19. buttonList = []  # 电梯楼层按钮
20. floorList = []
22. **def** \_\_init\_\_(self,buttonCount):
23. super().\_\_init\_\_()
24. self.initUI(buttonCount) #初始化UI界面
25. self.qtimer=QTimer(self) #初始化一个定时器
26. self.qtimer.timeout.connect(**lambda** :self.changeFloor())  #计时结束之后调用楼层显示方法
27. self.qtimer.start(1500)
29. **def** initUI(self,buttonCount):
30. lay =QGridLayout()
31. self.Lcd=QLCDNumber() #ui布局调用一个LCD显示
32. self.label=QLabel("状态")  #定义一个标签说明电梯状态
33. self.label.setStyleSheet("QLabel{color:blue}") #设置标签的背景颜色
34. self.Lcd.setSegmentStyle(QLCDNumber.Flat)
35. self.Lcd.setStyleSheet("border:0.5px solid black; color: red; background: silver;")
36. self.Lcd.display(1) #初始化楼层为1楼
38. #向控件lay中添加lcd和label
39. lay.addWidget(self.Lcd,0,0,1,1)
40. lay.addWidget(self.label,0,2,1,2)
42. #将楼层号按钮加入到lay中
43. **for** i **in** range(buttonCount):
44. button=QPushButton(str(i+1))
45. button.clicked.connect(self.Cli)
46. self.buttonList.append(button)
47. lay.addWidget(button)
49. #定义三个按钮，一个开门，一个关门，一个报警
50. self.close\_door=QPushButton("关门")
51. self.open\_door=QPushButton("开门")
52. self.warn\_but=QPushButton("报警")
53. self.close\_door.clicked.connect(self.Close)
54. self.open\_door.clicked.connect(self.Open)
55. lay.addWidget(self.close\_door)
56. lay.addWidget(self.open\_door)
57. lay.addWidget(self.warn\_but)
58. self.setLayout(lay)
59. self.show()
61. **def** Cli(self):
62. #获取按钮对象的内容并转换为数字
63. floor=int(self.sender().text())
64. #如果当前楼层等于按钮楼层，开门
65. **if** self.floor==floor:
66. self.Open()
67. **return**
68. **elif** floor > self.floor **and** floor **not** **in** self.upfloorList:
69. self.upfloorList.append(floor)
70. **if** self.status==0 **and** self.preStatus==1:
71. self.preStatus=1
72. self.status=1
73. **elif** floor<self.floor **and** floor **not** **in** self.downfloorList:
74. self.downfloorList.append(floor)
75. **if** self.status==0 **and** self.preStatus==-1:
76. self.preStatus=-1
77. self.status=-1
78. self.sender().setStyleSheet("QPushButton{background-color:red}")
79. self.Close()
81. #关门函数
82. **def** Close(self):
83. **if** self.door==0:
84. self.door = 1
85. self.status=self.preStatus
86. self.label.setText("电梯门关闭")
87. self.close\_door.setStyleSheet("QPushButton{background-color:red}")
88. self.open\_door.setStyleSheet("QPushButton{background-color:None}")
89. self.runStrategy()
90. self.qtimer.start(1500)
92. #开门函数
93. **def** Open(self):
94. **if** self.door==1:
95. self.door=0
96. self.preStatus=self.status
97. self.status=0
98. self.label.setText("电梯门打开")
99. self.open\_door.setStyleSheet("QPushButton{background-color:red}")
100. self.close\_door.setStyleSheet("QPushButton{background-color:None}")
101. self.qtimer.stop()
102. self.buttonList[self.floor-1].setStyleSheet("QPushButton{background-color:None}")
104. #运行策略函数
105. **def** runStrategy(self):
106. **if** self.status==1:
107. **if** len(self.upfloorList)==0 **and** len(self.downfloorList)!=0:
108. self.status=-1
109. self.label.setText("电梯下行")
110. **elif** len(self.upfloorList)!=0:
111. self.label.setText("电梯上行")
112. **elif** self.status==-1:
113. **if** len(self.downfloorList)==0 **and** len(self.upfloorList)!=0:
114. self.status = 1
115. self.label.setText("电梯上行")
116. **elif** len(self.downfloorList)!=0:
117. self.label.setText("电梯下行")
118. **if** len(self.downfloorList)==0 **and** len(self.upfloorList)==0:
119. self.status=0
120. self.label.setText("电梯停滞")
121. self.qtimer.stop()
123. #切换楼层函数
124. **def** changeFloor(self):
125. self.runStrategy()
126. **print**("还在运行中")
127. **if** self.status==1 **and** len(self.upfloorList)!=0 **and** self.aimFloor!=max(self.upfloorList):
128. self.aimFloor=max(self.upfloorList)
129. self.floorList=[]
130. **for** i **in** range(self.floor+1,self.aimFloor+1):
131. self.floorList.append(i)
132. **elif** self.status==-1 **and** len(self.downfloorList)!=0 **and** self.aimFloor!=min(self.downfloorList):
133. self.aimFloor=min(self.downfloorList)
134. self.floorList=[]
135. **for** i **in** range(self.aimFloor,self.floor):
136. self.floorList.append(i)
137. self.floorList.sort(reverse=True)
138. **if** self.status!=0:
139. self.displayfloor()
141. #楼层显示函数
142. **def** displayfloor(self):
143. **if** len(self.floorList)!=0 **and** self.door==1:
144. self.floor=self.floorList.pop(0)
145. self.Lcd.display(self.floor)
146. **if** self.status==1 **and** self.floor **in** self.upfloorList:
147. self.upfloorList.remove(self.floor)
148. self.Open()
149. **if** self.status==-1 **and** self.floor **in** self.downfloorList:
150. self.downfloorList.remove(self.floor)
151. self.Open()
153. **if** \_\_name\_\_=="\_\_main\_\_":
154. app=QApplication(sys.argv)
155. elevator=Elevator(random.randint(10,25))
156. app.exit(app.exec\_())
157. 运行结果与分析：



视频效果



四、实验心得与休会：

实验主要借助pyqt进行可视化编程，主要是构建电梯类，以及电梯运行的调度算法。

* 1. 使用pyqt进行可视化，设计布局，左上角为显示流程的lcd，以及状态，后面为根据楼层数目生成的楼层号，以及开门关门
  2. 主要思想是把上行和下行分为两个列表，在上行时，所有动作都在上行列表中，下行同理。