软件工程作业纸(1)

学号：221220144

姓名：张路远

邮箱：craly199@foxmail.com

1.

对于$Python$语言，最危险的特性或许是`动态类型`：变量可以在程序执行过程中被赋予不同类型的值，而不需要在编写代码时明确指定类型，可能导致运行时错误

然而，如果我编写编程团队的编码指南，我不会完全禁用该特性，因为这一特性的正面效果过于显著，而危险是可控的。我会允许编程团队在简单函数和封装模块中自由使用动态类型，如果该变量不存在风险；对于有风险的变量，需要进行类型注释并维护文档。此外，`mypy` 等静态类型检查工具也可以帮助规避风险

2.

对于我在“C语言程序设计”中完成的课程设计：简易通讯录系统

缩进：

我采用了`clang-format` 进行自动格式化，因此缩进正确

命名：

大部分函数命名能直接反应函数功能，并具有良好的大小写习惯；但也有部分命名，由于在编写代码时很清楚其具体功能，因此在命名中简化了一部分，例如“最大邮箱数量”简写为了`maxEmail`，这导致理解难度增加

重构：

代码可以从重构中收益。该项目是在大一时完成，是我的第一个编程项目，因此代码比较稚嫩

代码中有大量重复片段。例如我采用SDL来完成图形化界面，由于SDL库较为原始，我封装了一些函数来使用，但封装程度不够，在实际使用中往往有大量重复代码并没有封装到函数中：

```
*prepare something*
*prepare something*
*prepare something*
```

```
...
*My function*
```

如果将这些`*prepare something*`封装到函数中，代码会更简洁

此外，代码中的一些逻辑判断，如检查输入长度、处理用户输入等，也可以进行抽象和封装

在注释方面，虽然代码中有一些注释，但整体上注释不够充分，仅仅足够我本人理解代码。如果在关键逻辑和复杂函数处添加更多的注释，就能帮助其他阅读者理解代码的功能和目的

3.

对于**2.**中提到的程序，圈复杂度最大的函数是`home`函数，这是程序的主循环函数，它的圈复杂度为11：主循环while有1条边；按钮循环检测两种鼠标事件，有2条边；8个按钮对应8个分支，有8条边

高圈复杂度的缺点：

可读性差，很难直接理解代码的执行逻辑或追踪其行为

难以维护或测试，可能的错误点更多

控制流结构可能冗余，灵活性不高

降低代码圈复杂度的方法：

提取函数：将具有高圈复杂度的代码块提取到独立的函数中，每个函数负责单一的功能，减少逻辑复杂性

使用状态模式：每个状态作为一个独立的类或函数，避免使用大量条件语句来判断当前状态

分支合并：如果多个`if`或`case`分支中的代码非常相似，可以合并以减少冗余的代码

抽象与封装：对功能模块进行抽象和封装，避免将所有逻辑都集中在一个函数中