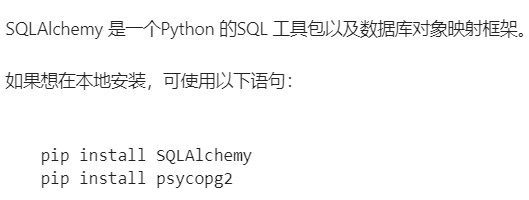
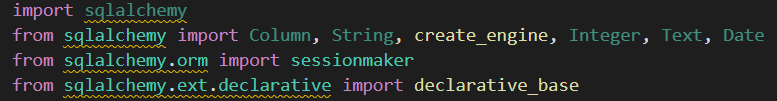
1. 实验前要在 VScode 上先运行下面两条指令：

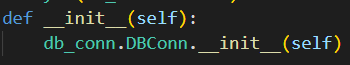


要换成作业要求的数据库，记得在文件前 import 需要的相关头文件。我自己是引用了下面这些东西，可以在我改好的文件里复制黏贴：



1. 要加什么属性或者表格直接按照格式在 be/model/store.py 插入即可。增加表格即插入一个类。此后每个文件要调用表格需要插入。

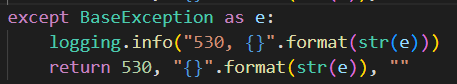
每个 be/model 功能文件里有下面东西就可以和数据库交互了：



交互语句示例：查找表格语句格式（需要注明表格类的来源为 store.表格名.属性名）



1. 通过看 doc 里面的 .md 文件知道要如何修改be/model/ 中的功能文件，明确传什么样的执行码和执行信息。不需要的码（比如530）可以去掉，比如下面这一坨选择去掉。

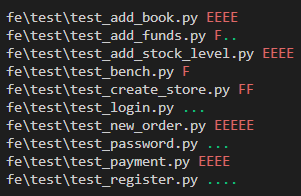


参考代码看 be/model/user.py。

1. 抓取错误原本代码：

改成即可，是同一意思。（如果我理解错误不止这一种的话，可以看看cn的代码）

1. 语句中的 self.conn 是一般查询语句中的 session。
2. 前60% 看样子确实只要改 be/model，但是不全是语句翻译，还要参照 fe/test 中的文件进行修改，test\_xx.py 文件中有多少个 test\_ 开头的函数就有几个测试用例（对应 ERROR / PASSED / FAILED）。而 fe/test 中的文件又链接着 fe/access 中的文件，注意看其中的 url 可以得知测试被链接到哪。参考注释看 fe/access/auth.py。
3. 我 db\_conn.py 文件也改了，但如果和后面 sell.py 和 buyer.py 功能不能匹配的话再自己改一下吧（注释部分的代码是它之前的原有代码，我没删掉）。
4. 进度（绿点是已经通过的，user.py 相关测试全部正确）：



如果pytest 整个太慢了也可以一个个test\_函数测试，执行语句示例“pytest test\_register.py”

1. 总体框架我应该算懂了个大概，其他有什么不懂的再问我吧