

[toc]

选做

ch00

作业题目：阅读习惯

任务详情

- 1. 推荐参考 [批判性思维书单](#)，[机关公文写作书单](#)，[\[公务员素质书单\]](#) (https://weread.qq.com/misc/booklist/3107758_7usfrsrTZ) 从中选择阅读，养成阅读习惯
 - 2. 提交学期初微信读书（或其他平台）的读书数据（总时长，册数，笔记数等）的截图（看到作业就截图）
 - 3. 提交学期末微信读书（或其他平台）的读书数据（总时长，册数，笔记数等）的截图
 - 4. 谈谈本学期收获，对阅读的理解，养成良好的阅读习惯了吗？会一直坚持阅读吗？
 - 5. 这学期所读的书中，如果只推荐一本，你愿意推荐哪本？这本书主要内容是什么？让你产生了什么改变？让你有了什么新行动？
 - 6. 使用 Markdown 格式，并转化为 PDF一起提交。
- [文档的github链接](#)

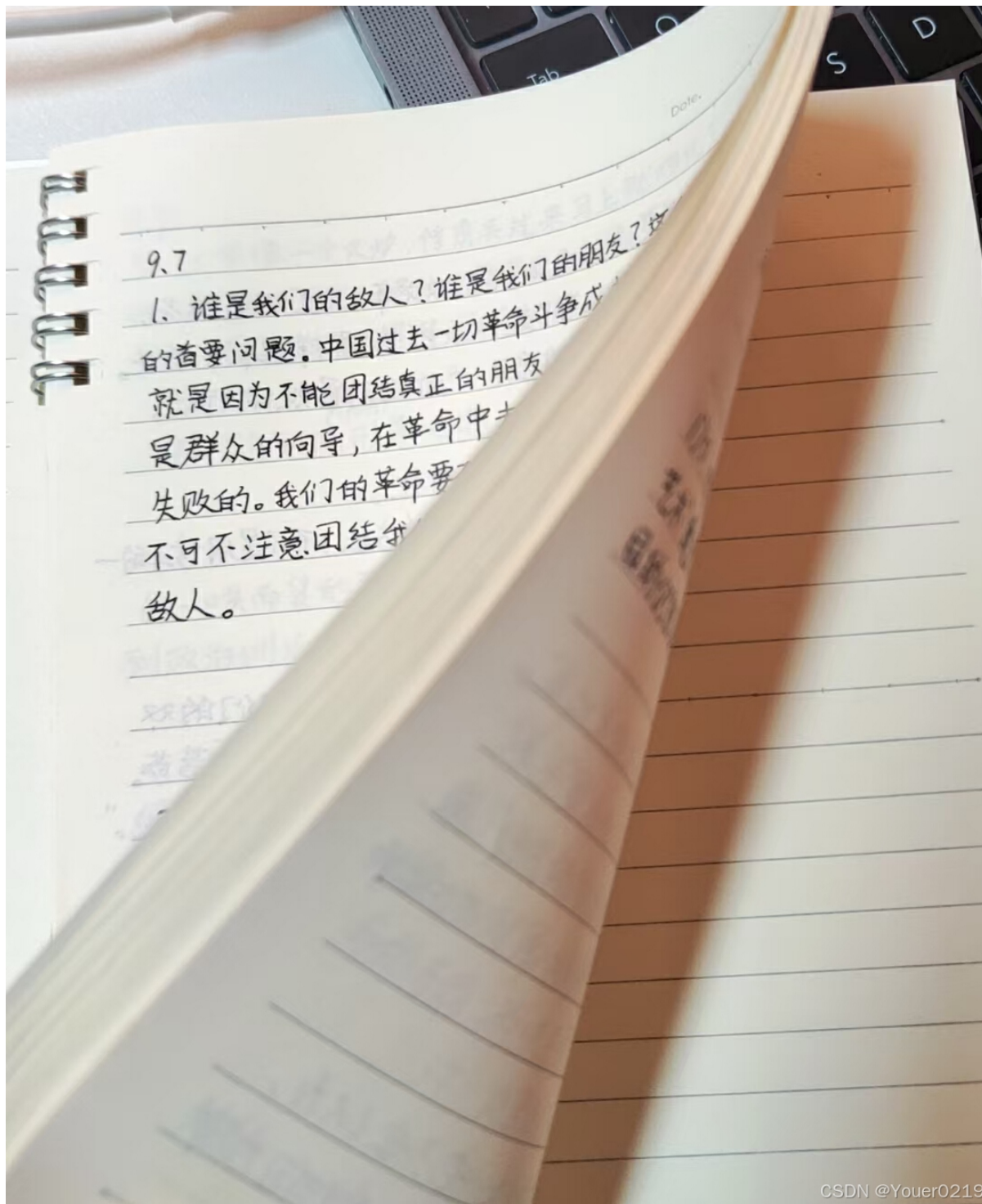
具体内容

总述

- 本学期没有在线上平台读书，而是购买了纸质书，包括：《代码整洁之道》、《代码整洁之道——程序员的职业素养》、《毛泽东选集》。
- 对于《毛泽东选集》，我坚持每日阅读并摘抄（这也是思政课程的要求）。对于《代码整洁之道》等书，我更加注重在实际的应用，特别是在这一学期的代码编程中。
- 书籍展示



- 摘抄展示



谈谈本学期收获，对阅读的理解，养成良好的阅读习惯了吗？会一直坚持阅读吗？

- 我个人对阅读的理解可以偏实用一些，读了书要“用”
 - 毛选
 - 阅读毛选让我学习毛主席分析问题的方法与立场，学习什么是实事求是，什么是人民立场，这可以用于理解很多政治概念的由来与演变，理解中国的发展。
 - 同样，这也是学习理论，也可以用于公文写作之中。
 - 阅读《代码整洁之道》则是了解更多的编程概念与(别人的)“最佳”实践，这可以直接用在实验四以及其他的项目中，来改正不良习惯，增强编程能力。
- 养成良好的阅读习惯了吗？会一直坚持阅读吗？

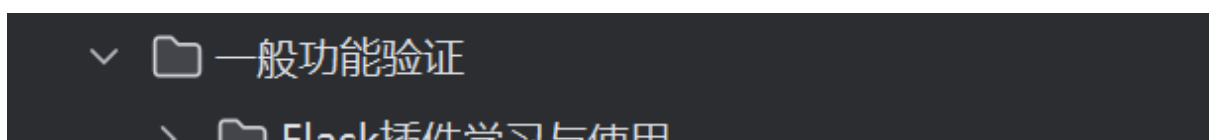
- 阅读肯定会一直坚持的，书就在那里，想要做事情就要阅读！
- 但很难说自己的阅读习惯是否良好，这需要更多的实践。




书籍推荐：《代码整洁之道》

- 这里推荐《代码整洁之道》。
- 书籍内容：
 - 本书主要讲述如何编写整洁的代码，从命名、函数、注释、格式等角度来提高代码的整洁性(也可以理解为可读性、可维护性、可扩展性的结合)。
 - 命名规范：表意清晰、名副其实，让人一眼能明白其用途
 - 函数规范：短小精悍、参数控制（无参最佳，尽量少）、单一职责
 - 类的设计：职责明确、尽量小
 - 注释规范：用在那些代码逻辑难以直观体现或者需要特别提醒等必要的地方，避免过多无意义、冗余的注释
 - 格式规范：统一、合理的代码格式，包括缩进、空格。让代码在视觉上就呈现出清晰的逻辑层次。
 - 测试规范：及时测试。测试代码的质量要求与其他代码一致。
 - 系统设计：构造和使用相互分离(依赖注入、工厂模式)，[测试驱动](#)
 - 迭进：运行所有测试、重构、消除重复代码以及提高代码表达力
 - 从第13章开始，其所涉及的代码量与业务逻辑都超过了我的知识范畴，这方面无法进行总结
- 书籍中大量加入了Java代码作为示范，演示了错误代码的写法与改进之道。
- 但是，我本身对于Java语言不是很熟悉，对其涉及的业务逻辑也不熟悉，这又是纸质书籍，无法很好的借助AI来理解代码。所以，越读到后面就越难理解。（这或许是最应该读电子版的书了）
- 尽管作者声称不细细阅读后面几章(第二部分)的对大段代码的重构，那么这本书的价值就会大打折扣。但对于我目前的水平而言，能消化一些就行。
- 同时，[这也不影响我实践这些原则，找到适合当前自己的最佳实践。](#)
- 那么，下面我就用我在实验四中的实践来展示我从书中学到了什么吧。

测试功能与系统搭建

- 代码整洁一书十分百分得重视测试的作用。测试代码的要求与功能测试是一致的！同时系统一章中的测试驱动概念我也很喜欢。
- 我从中学到的是，要先测试好功能能否可以实现与如何实现，再将其一步步集成到项目系统中
- 在实验四的项目中，有一个额外的文件夹——“[一般功能验证](#)”。里面都是我事先测试功能留下的代码。
- 就目前而言，我做不到规范的写测试，也不会什么单元测试。但先测试验证功能，再集成到核心项目中，这个还是很适合我的。



- ✓  SDF设备连接测试
 - ✓  龙脉
 - >  客户端WEB访问KEY
 - >  服务端python访问KEY
- ✓  国密算法测试
 -  testCreateSM2keys.py
 -  testHMAC.py
 -  testSM2.py
 -  testSM3.py
 -  testSM4.py
- ✓  强制HTTPS访问
 -  cert.pem
 -  key.pem
 -  testForceHTTPS.py
- ✓  数据库备份功能验证
 - >  CSV备份方法(失败)
 - >  mysqldump方法
- ✓  简单功能测试
 -  first_python.py
 -  testDecorator.py
 -  testEmail.py
 -  testMysql.py
 -  testTuple.py

- 就实际应用而言，与测试驱动有些不同。我是先明确一个需求，大概需要什么功能，但没有直接写测试数据，而是让AI把相关功能的代码与数据都生成了再验证功能。毕竟我对这些业务也不熟。
- 同时，在这过程中也不是直接生成就完事的。一般有这么几步：
 - 一：直接的代码，测试直到成功
 - 二：将代码封装成为函数，明确合适的输入输出，再测试与重构
 - 三：一般没有封装为类的需求，直接将其导入到项目中，再测试与重构
 - 中途的每一步都可能根据功能的复杂程度划分为一个个小步，注意每一步都测试与重构即可
- 这种先验证功能的做法为我带来了不少好处。
 - 一是开发决策时避免不可能实现的方向：比如，在使用龙脉KEY时，我最初的打算是应用在web端，但功能验证时根本通不过测试，所以就放弃了，改为正常的口令登录验证了。
 - 二是可以多线程开发——主项目做累了就可以根据自己预计的方向去做功能验证，验证成功的功能可以等主项目进行到这一步后再集成到系统中，同样可以举出龙脉KEY的测试例子。
 - 龙脉KEY虽然无法在web端使用，但他在Windows上的使用时是可行的，其c语言编写的demo验证了这一点。所以之后就是用python调c，转换代码的事情了。
 - 但这一步并不是我做到密钥管理具体功能时才去测试的，我提前就做好的相关的准备。在做这个测试的同时，主项目那边再干公文界面相关功能。
 - 结合一起来看，提前验证可以给自己开发增加信心，知道这些功能是自己可以实现的。



如何重构——以执行SQL语句的函数(con-my-sql)为例

- 我在书中学到的重要一点，就是既要有时时重构的意识，也要知道，重构不是破坏性的。这里就以执行SQL语句的函数为例。
- 现在我们有一个con-my-sql函数，函数的作用是执行SQL语句，返回结果。这是他最开始的样子(伪代码):

```
conn = pymysql.connect(  
    # 连接参数省略  
)  
def con_my_sql(sql_code, params=None):  
    """
```

执行SQL语句的函数，支持参数化查询，增强安全性。

```

:param sql_code: SQL语句模板，对于参数化查询部分用占位符表示（如 %s）
:param params: 对应SQL语句模板中占位符的参数值，以元组或列表形式传入（可选参数，如果SQL语句不需要传入参数则可以不传）
:return: 如果执行成功，返回游标对象；如果出现异常，返回包含异常类型和异常信息的元组
"""
# 使用conn对象执行SQL语句，具体业务逻辑省略

```

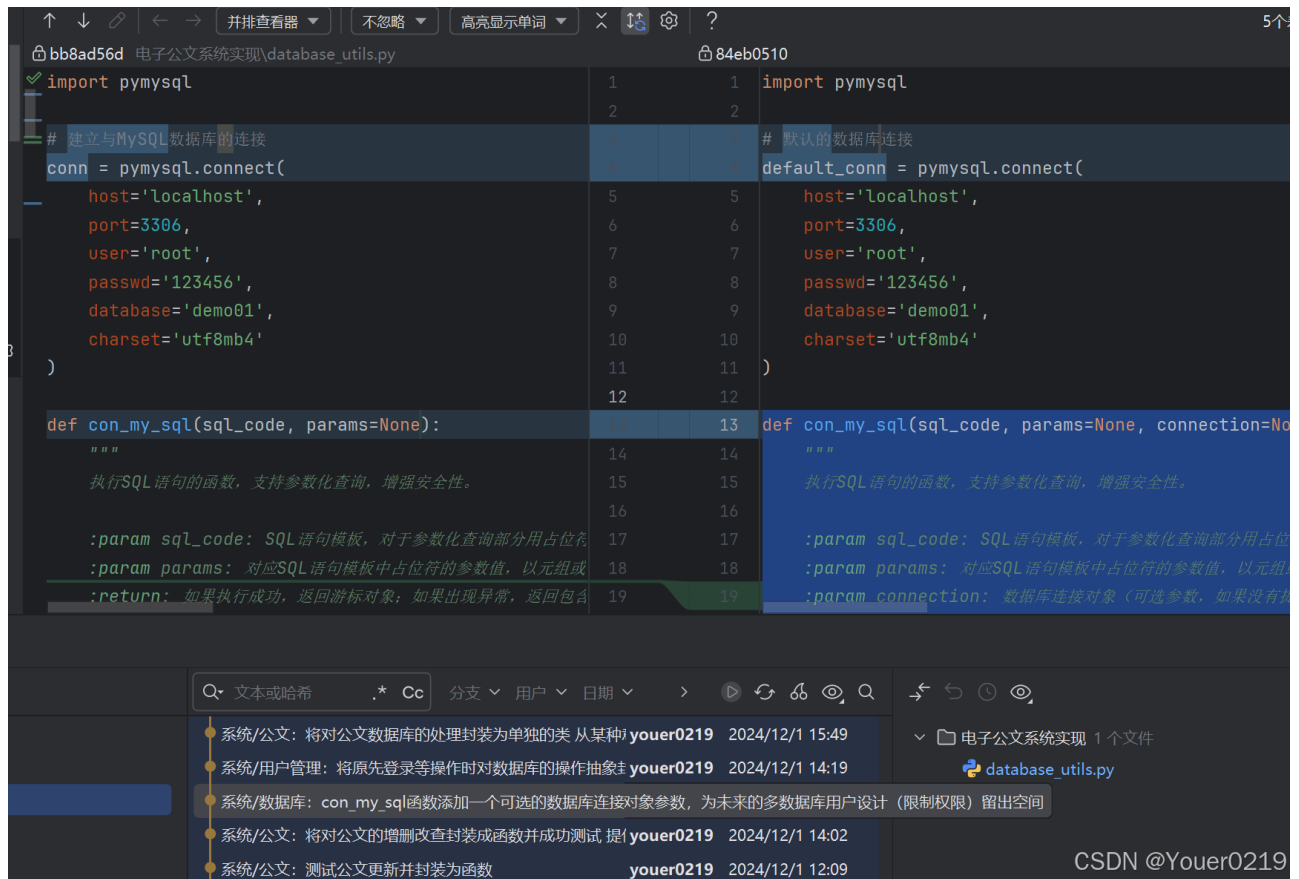
- 重构总是有功能需求的，当时我想到：未来我希望能用不同的数据库用户连接数据库，以合理划分权限。这需要构建不同的连接，也就是说，不能使用默认连接而是需要传入连接参数。这就有一个问题：如何既为实现这个功能保留框架又能兼容目前的系统
 - 这有提前设计的嫌疑，因此，如果想不到，可以放弃
- 我的方法是，添加可选参数，并在代码中判断是否有合法的连接传入，如果没有，还是使用默认连接。

```

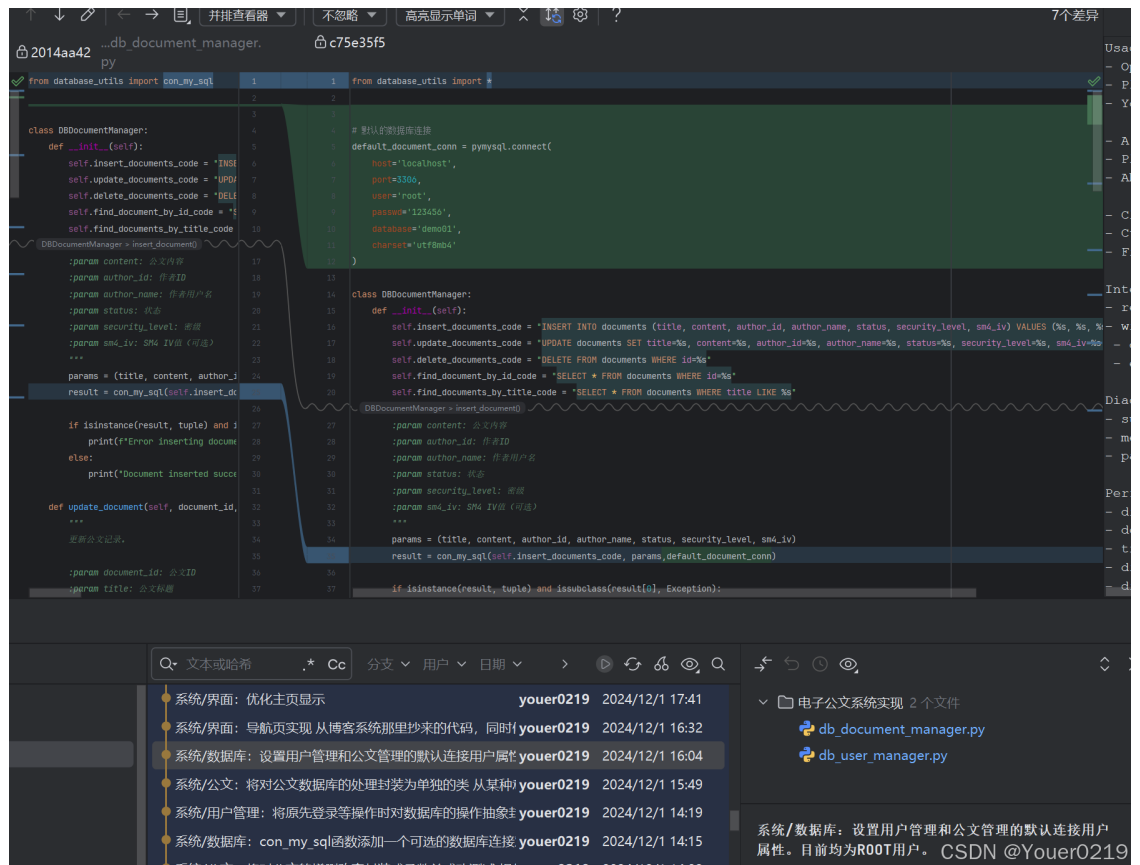
default_conn = pymysql.connect(
# 连接参数省略
)
def con_my_sql(sql_code, params=None, connection=None):
    """
    执行SQL语句的函数，支持参数化查询，增强安全性。

    :param sql_code: SQL语句模板，对于参数化查询部分用占位符表示（如 %s）
    :param params: 对应SQL语句模板中占位符的参数值，以元组或列表形式传入（可选参数，如果SQL语句不需要传入参数则可以不传）
    :param connection: 数据库连接对象（可选参数，如果没有提供则使用默认连接）
    :return: 如果执行成功，返回游标对象；如果出现异常，返回包含异常类型和异常信息的元组
    """
    # 使用提供的连接或者默认连接
    conn = connection if connection else default_conn
    # 使用conn对象执行SQL语句，具体业务逻辑省略

```

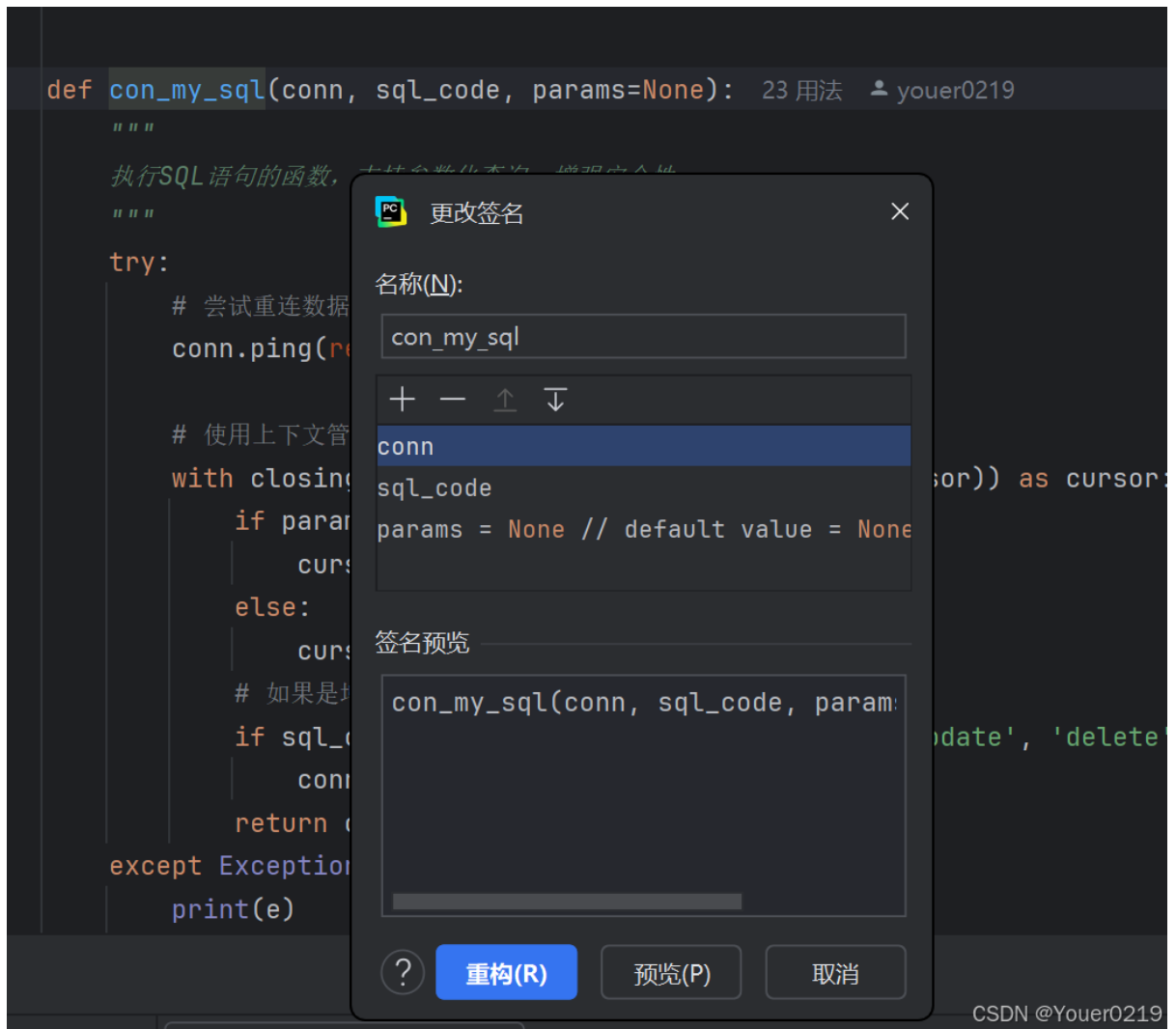


- 这样就不需要改动其他地方的代码也可以让程序正常运行。之后再一个个为其他地方添加上自己专属的连接并应用+及时测试。
- 这个“专属的连接”先用和默认用户一样的root用户即可。因为这不是当时的重点

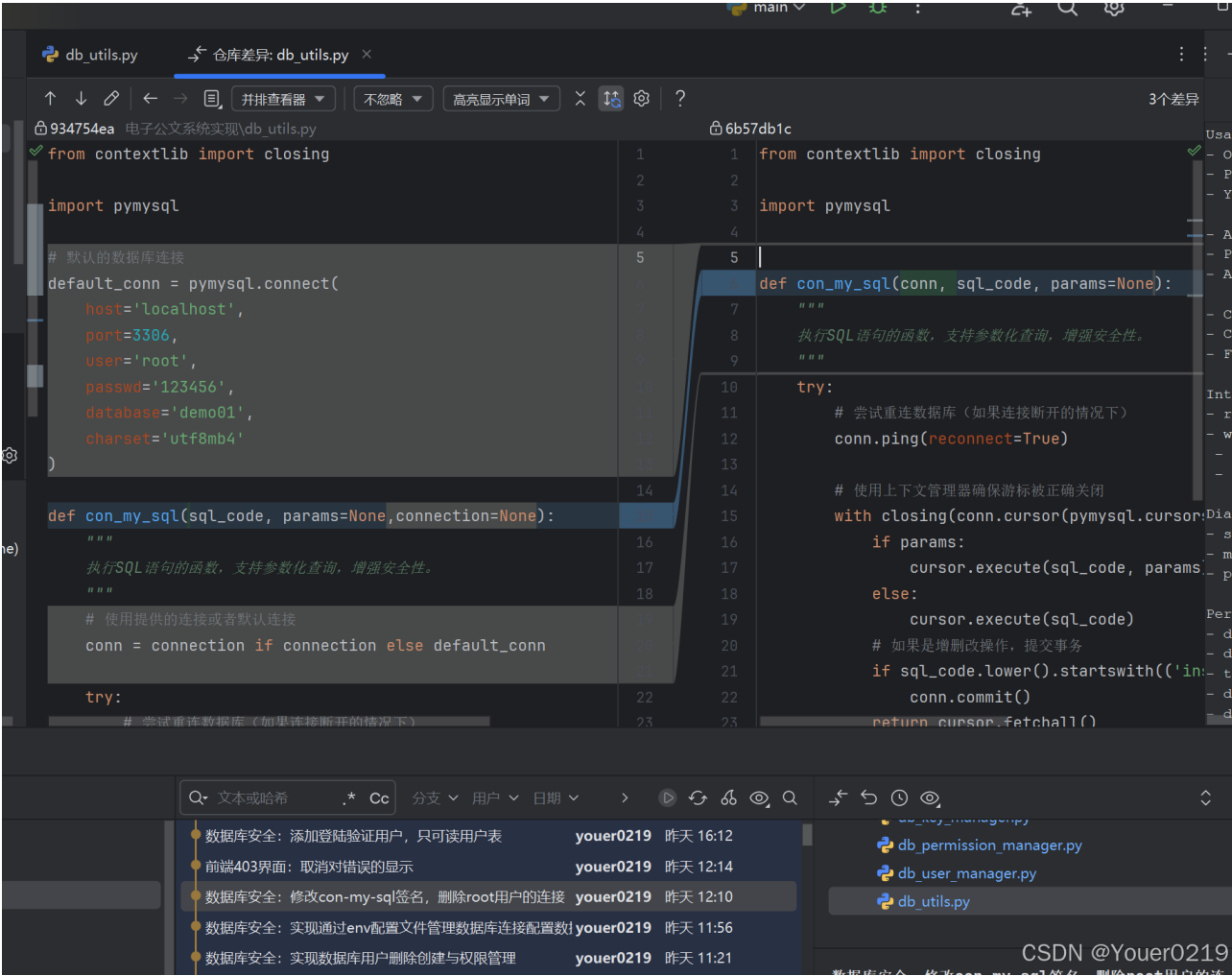


- 现在，我们已经有了这个框架，各个调用con-my-sql函数的地方都有自己的专属用户的连接，我们可以将为了兼容留下来的root用户的连接删除，并要求所有调用该函数的实体传入自己的连接参数。

- 删除'conn = connection if connection else default_conn'并修改'connection'的命名为'conn'即可解决绝大部分问题。
- 但是，由于之前是可选参数且放置在'params'后面，不允许将连接参数变为必选参数。而如果保留其可选性质的话，无疑是给后续开发埋坑。
- 那么，这就是现代IDE的作用了，直接重构代码，全局修改函数签名，将连接参数置为第一个。
- 下图是已经变更后的结果，当时没有截图：



- 由此，再配合其他使用con-my-sql函数的地方对其连接参数的修改，就可以直接实现用不同的数据库用户连接数据库的目标了。



结语

读书总是要用的，不用的话读不活。这是我这一学期对于阅读的感悟。

很多书我确实读不懂，但用起来后就能多少明白作者的意思与自己的方向。

接下来的2025年，继续阅读，继续应用，继续成长。