**《Python自动化运维》课程**

**期末学习报告**

班级：\*\*\*\*\*

姓名：\_\_\_\_\_\_\*\*\*\_\_\_\_\_\_\_\_\_\_\_

学号：\_\_\_\_\_\_\*\*\*\*\*\*\*\*\_\_\_\_\_\_\_\_\_\_\_

1. **任务背景与需求分析（宋体，加粗，小三号字）**

1．任务背景（宋体，四号字）

本任务的背景是公司最近在进行新产品的市场测试，需要收集关于产品的用户反馈数据。每天需要从公司的邮箱中提取并整理用户反馈邮件，并将数据自动添加到公司的分析报告中。报告需要包括反馈内容，用户信息和产品版本号。将他们的联系信息存档以备后续联系使用。

2. 需求分析

**本任务需要实现一个自动化邮件处理任务，需求如下：**

**1,自动提取反馈数据：程序需要能够实现自动从公司邮箱中提取反馈邮件，并将邮件中的反馈内容、用户信息等数据提取出来。**

**2,数据整理和分析：程序需要将提取出来的数据进行整理和分析，生成分析报告。并且存储到数据库中，报告需要包括反馈内容、用户信息。**

**3,程序可定制：程序需要能够根据公司的具体需求进行化定制，比如关键词匹配、过滤垃圾邮件等功能。**

**4,程序稳定性和安全性：程序需要具有稳定性和安全性，能够保证数据的准确性和安全性。**

1. **设计方案**
2. 程序结构

程序可以分为三部分：邮箱模块、反馈数据处理模块和邮件发送模块。其中，邮箱模块负责连接到公司邮箱，并自动获取最新邮件；反馈数据处理模块负责提取邮件中的数据，并进行分析和整理；邮件发送模块负责对用户进行回复，并将数据存档。这些模块都可以单独实现，然后通过API或其他方式进行集成。

1. 业务处理流程

程序从邮箱中获取待处理邮件，并将邮件内容传递给反馈数据处理模块。反馈数据处理模块对邮件进行解析和分析，提取出反馈内容、用户信息和产品版本号等数据。然后，将这些数据整理成表格或其他形式的报告，并将报告保存到数据库中。用户的联系信息存档到数据库中。

1. 数据存储

程序的反馈数据处理模块可以将提取到的数据存储到数据库中。数据库可以使用关系型（如MySQL或PostgreSQL）或非关系型（如MongoDB或Redis）数据库。以及输出到excel文件当中

1. 数据处理分析

对于数据的分析和整理，可以使用Python编写过滤和分析代码。另外，程序还可以使用线性判别分析、主成分分析或其他的机器学习算法来提取数据特征，为公司的决策提供支持。

1. **实现细节**

1,关键代码

1,从公司邮箱中获取最新邮件

# 2，设置IMAP服务器和认证信息

# IMAP服务器

imap\_server = "imap.gmail.com"

# 用户名

imap\_username = "wanlonglongiii@gmail.com"

# 密码

imap\_password = "ovbiornryoqckujk"

# 3，连接IMAP服务器并登录

# SSL方式连接IMAP服务器

mail = imaplib.IMAP4\_SSL(imap\_server)

mail.login(imap\_username, imap\_password)

1. 提供反馈数据

# 6，遍历所有邮件的UID并获取RFC822内容

for uid in data[0].split():

# 获取RFC822格式的邮件内容

result, data = mail.uid('fetch', uid, '(RFC822)')

# 获取邮件内容的第二部分

raw\_email = data[0][1]

# 解析邮件内容

email\_message = email.message\_from\_bytes(raw\_email)

# 邮件的主题和发件人。

# 解码邮件主题

email\_subject = decode\_header(email\_message["Subject"])[0][0]

# 解码邮件发件人

email\_from = decode\_header(email\_message["From"])[0][0]

print(f"Email: {email\_subject} from {email\_from}")

# 获取邮件正文

for part in email\_message.walk():

if part.get\_content\_type() == "text/plain":

# 解码邮件正文

email\_body = part.get\_payload(decode=True).decode()

3,数据存储

# 连接MySQL数据库，需要提供主机名（host）、用户名（user）、密码（password）、数据库名称（database）和字符集（charset

mydb = pymysql.connect(

host="localhost",

user="root",

password="root123",

database="mydatabase",

charset='utf8'

)

# 获取数据库操作游标

mycursor = mydb.cursor()

# 插入邮件正文到MySQL中 定义要执行的SQL语句，将邮件正文和发件人地址插入到名为emails的表中的body和address字段中

sql = "INSERT INTO emails (body, address) VALUES (%s, %s)"

# 定义要插入的数据，即邮件正文和发件人地址，使用元组（tuple）来存储

val = (email\_body,email\_from)

# 执行SQL语句，将数据插入到数据库中

mycursor.execute(sql, val)

# 提交事务，将数据插入到数据库中

mydb.commit()

**4**,数据分析

# 获取查询结果

result = cursor.fetchall()

for row in result:

print(row)

# 创建DataFrame

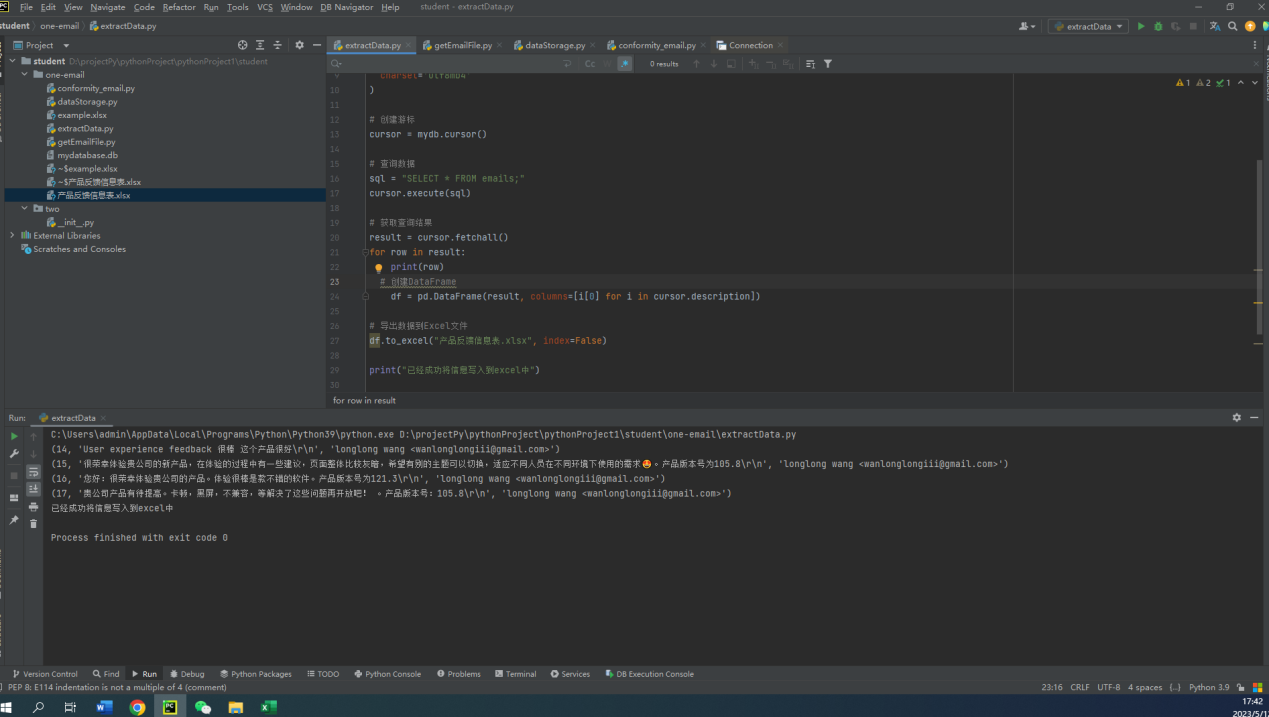
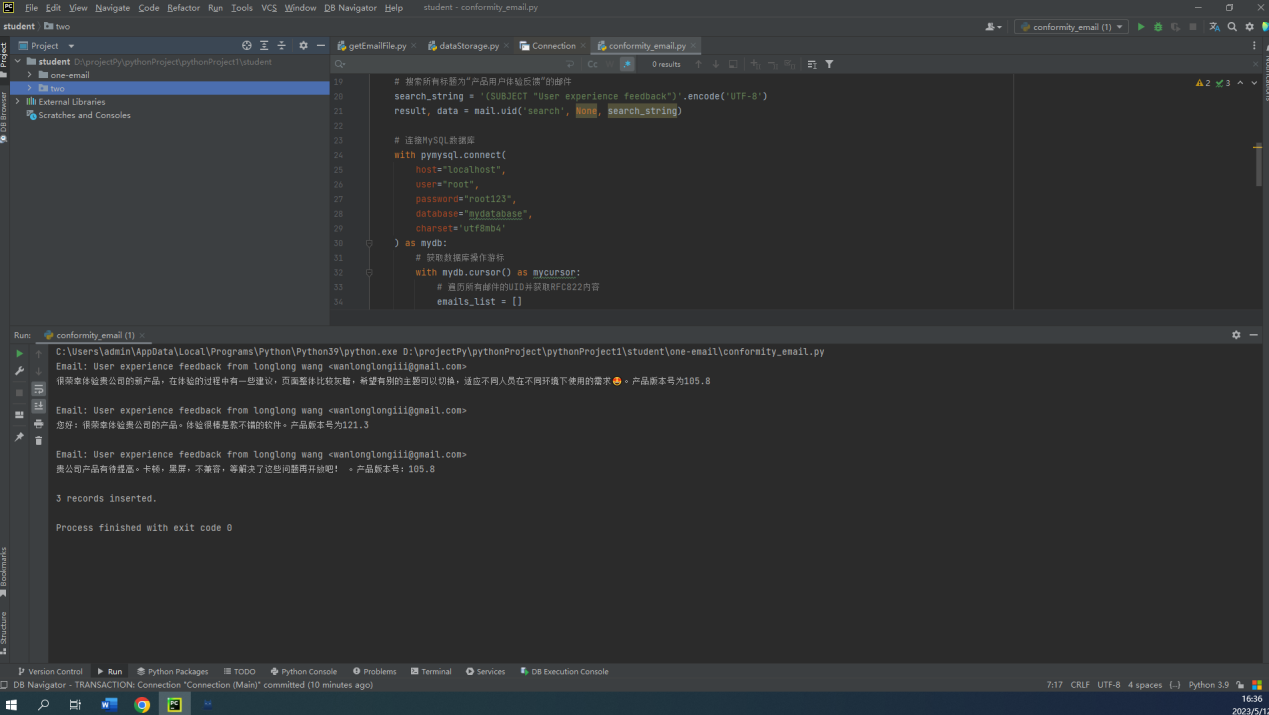
df = pd.DataFrame(result, columns=[i[0] for i in cursor.description])

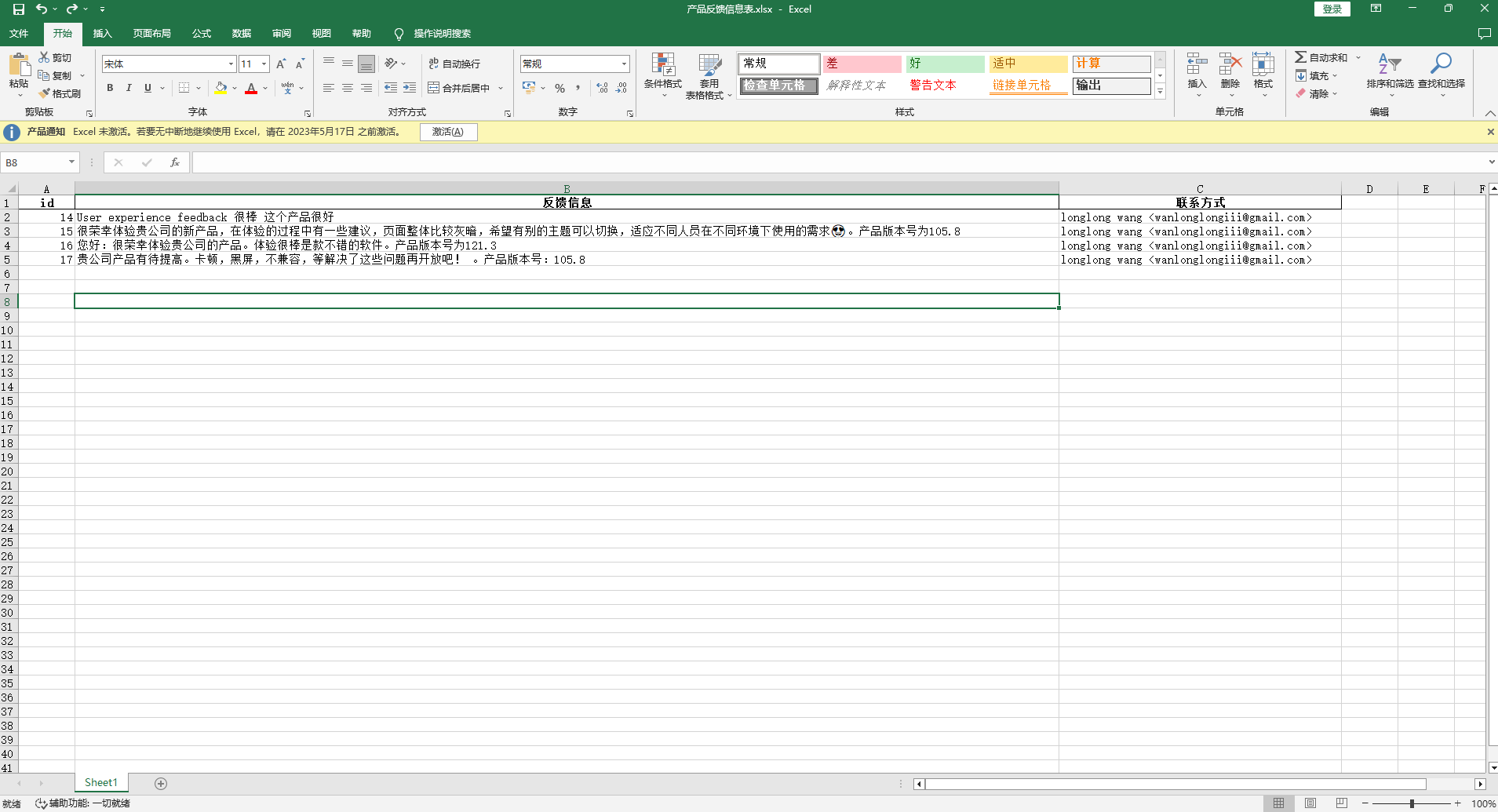
# 导出数据到Excel文件

df.to\_excel("产品反馈信息表.xlsx", index=False)

1. 运行截图

获取邮件中的信息-并存储到数据库，并且保存练习方式

将数据库中的数据获取存储到excel



1. **结果分析**

该代码段实现了从 IMAP 服务器中获取特定主题的邮件，并将邮件的正文和发件人存储到 MySQL 数据库中。以下是代码执行的结果分析：

连接到 IMAP 服务器：

成功连接到 IMAP 服务器 "imap.gmail.com"。

使用提供的用户名和密码成功进行身份验证。

选择收件箱：

成功选择收件箱 "inbox"。

搜索特定主题的邮件：

使用主题搜索字符串 '(SUBJECT "User experience feedback")' 进行搜索。

获取搜索结果，包含满足搜索条件的邮件 UID。

连接到 MySQL 数据库：

成功连接到本地 MySQL 数据库服务器。

使用提供的用户名和密码进行身份验证。

连接到名为 "mydatabase" 的数据库。

获取数据库操作游标：

成功获取数据库操作游标。

遍历邮件 UID 并获取邮件内容：

遍历搜索结果中的每个邮件 UID。

获取邮件的 RFC822 格式内容。

解析邮件内容，并提取主题、发件人和正文。

插入邮件正文到 MySQL 数据库：

将邮件正文和发件人信息存储在名为 "emails" 的表中。

使用插入语句 "INSERT INTO emails (body, address) VALUES (%s, %s)"。

执行插入操作，并将多个邮件的信息一次性插入数据库。

提交事务以确保数据持久化。

打印插入的记录数：

打印成功插入数据库的记录数。

关闭邮箱和退出登录：

成功关闭 IMAP 邮箱连接。

成功退出登录。

总结：

该代码段通过与 IMAP 服务器和 MySQL 数据库的交互，实现了从邮件服务器获取特定主题的邮件，并将邮件的正文和发件人信息存储到数据库中。执行结果表明，连接、搜索、解析、插入和提交等操作都顺利完成，并成功将邮件数据存储在数据库中。执行结果分析提供了对每个步骤执行的情况进行了详细说明，以及成功插入的记录数。这有助于确保代码的正确性和数据的完整性，并为进一步的分析和处理提供了基础。

1. **系统涵盖的主要知识点及完整源代码**

使用 pymysql 模块与 MySQL 数据库进行连接和操作。

使用 imaplib 模块与 IMAP 服务器进行连接和操作。

使用 email 模块解析电子邮件消息的主题、发件人和正文。

使用 decode\_header 函数对邮件主题和发件人进行解码，以处理可能的编码问题。

使用 IMAP 协议搜索特定主题的邮件，并获取邮件的 UID 和 RFC822 格式的内容。

将邮件的正文和发件人存储到列表中。

使用 SQL 语句将邮件正文插入到 MySQL 数据库中。

使用 commit() 方法提交数据库操作，确保数据持久化。

使用 print() 函数打印相关信息，如邮件主题、发件人、邮件正文和插入的记录数。

程序实现了从 IMAP 服务器获取特定主题的邮件，并将邮件的正文和发件人存储到 MySQL 数据库中。它展示了如何与数据库和邮件服务器进行交互，以及如何解析电子邮件消息的内容和元数据。代码中还包含一些错误处理和打印结果的步骤，以提供更好的可读性和用户反馈。

1. 源代码

email Fetch.py

1. # 1，导入所需的模块，包括imaplib、email和decode\_header。
2. # IMAP库，用于邮件的接收
3. import imaplib
4. # email库，用于解析邮件
5. import email
6. # 解码邮件头
7. from email.header import decode\_header
8. # 2，设置IMAP服务器和认证信息
9. # IMAP服务器
10. imap\_server = "imap.gmail.com"
11. # 用户名
12. imap\_username = "wanlonglongiii@gmail.com"
13. # 密码
14. imap\_password = "ovbiornryoqckujk"
15. # 3，连接IMAP服务器并登录
16. # SSL方式连接IMAP服务器
17. mail = imaplib.IMAP4\_SSL(imap\_server)
18. mail.login(imap\_username, imap\_password)
19. # 4，选择收件箱
20. mail.select("inbox")
21. # 5，搜索所有标题为“产品用户体验反馈”的邮件
22. # 搜索字符串
23. search\_string = '(SUBJECT "User experience feedback")'.encode('UTF-8')
24. # 搜索匹配邮件的UID
25. result, data = mail.uid('search', None, search\_string)
26. # 6，遍历所有邮件的UID并获取RFC822内容
27. for uid in data[0].split():
28. # 获取RFC822格式的邮件内容
29. result, data = mail.uid('fetch', uid, '(RFC822)')
30. # 获取邮件内容的第二部分
31. raw\_email = data[0][1]
32. # 解析邮件内容
33. email\_message = email.message\_from\_bytes(raw\_email)
34. # 邮件的主题和发件人。
35. # 解码邮件主题
36. email\_subject = decode\_header(email\_message["Subject"])[0][0]
37. # 解码邮件发件人
38. email\_from = decode\_header(email\_message["From"])[0][0]
39. print(f"Email: {email\_subject} from {email\_from}")
40. # 获取邮件正文
41. for part in email\_message.walk():
42. if part.get\_content\_type() == "text/plain":
43. # 解码邮件正文
44. email\_body = part.get\_payload(decode=True).decode()
45. # 打印邮件正文
46. print(email\_body)
47. # 7,关闭连接
48. # 关闭邮箱
49. mail.close()
50. # 退出登录
51. mail.logout()

dataStorage.py

# 导入pymysql模块

import pymysql

# 导入getNewEmailFile.py中的email\_body和email\_from

from getEmailFile import email\_body,email\_from

# 连接MySQL数据库，需要提供主机名（host）、用户名（user）、密码（password）、数据库名称（database）和字符集（charset

mydb = pymysql.connect(

host="localhost",

user="root",

password="root123",

database="mydatabase",

charset='utf8'

)

# 获取数据库操作游标

mycursor = mydb.cursor()

# 插入邮件正文到MySQL中 定义要执行的SQL语句，将邮件正文和发件人地址插入到名为emails的表中的body和address字段中

sql = "INSERT INTO emails (body, address) VALUES (%s, %s)"

# 定义要插入的数据，即邮件正文和发件人地址，使用元组（tuple）来存储

val = (email\_body,email\_from)

# 执行SQL语句，将数据插入到数据库中

mycursor.execute(sql, val)

# 提交事务，将数据插入到数据库中

mydb.commit()

# 打印插入的记录数

print(mycursor.rowcount, "record inserted.")

extractData.py

import pymysql

import pandas as pd

# 连接数据库

mydb = pymysql.connect(

host="localhost",

user="root",

password="root123",

database="mydatabase",

charset='utf8mb4'

)

# 创建游标

cursor = mydb.cursor()

# 查询数据

sql = "SELECT \* FROM emails;"

cursor.execute(sql)

# 获取查询结果

result = cursor.fetchall()

for row in result:

print(row)

# 创建DataFrame

df = pd.DataFrame(result, columns=[i[0] for i in cursor.description])

# 导出数据到Excel文件

df.to\_excel("产品反馈信息表.xlsx", index=False)

print("已经成功将信息写入到excel中")

# 关闭连接

mydb.close()