保存成功4import subprocess

Shift + Ctrl + V

粘贴时添加一个换行符粘贴时添加一个换行符

粘贴时添加一个换行符粘贴时添加一个换行符

008\_人工智能(AGI) 008\_人工智能(AGI)

008\_人工智能(AGI)

# 008\_人工智能(AGI)

| 项 | 值 | | ------------ | ----------------------------------- | | Provider | Anthropic | | API Key | 刚才复制的 Key |

| Base URL | `https://api.moonshot.cn/anthropic` |

| Model | 保持默认即可（会自动指向 kimi-k2） |

| Browser Tool | 建议关闭（Disable） |

1保留原有所有功能

2第1个是我的源文件

3第2个是你给的代码修改后的目标文件

4用你修改后的代码替换进去之后第1张图片红框处文件\_\_version参数自动加点功能没有了

5请比较一下你给的代码出了什么问题

6重新给出修改代码

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dest\_dir = [

r'''多行路径字符串'''.replace('\n', ''), # 源文件路径

r'''多行路径字符串'''.replace('\n', '')

]\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# 创建包含多个多行字符串的列表

source\_path = [

"""

这是另一个

多行字符串示例

包含三个段落

""",

"""

第一行

第二行

第三行

""",

"""

C:\Users\Administrator\AppData\Roaming\Listary\UserProfile

\Settings\SearchHistory.json

""",

"""

C:\Users\Administrator\AppData\Roaming\IrfanView\i\_view64.ini

""",

"""

C:\Users\Administrator\AppData\Roaming\Ditto\005\_211\_Ditto!2\_数据备份.db

"""

]

# 访问并打印第一个多行字符串

print(source\_path[0])

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

包含三个段落

""",

"""

第一行

第二行

第三行

""",

"""

C:\Users\Administrator\AppData\Roaming\Listary\UserProfile

\Settings\SearchHistory.json

""",

"""

C:\Users\Administrator\AppData\Roaming\IrfanView\i\_view64.ini

""",

"""

C:\Users\Administrator\AppData\Roaming\Ditto\005\_211\_Ditto!2\_数据备份.db

"""

]

# 访问并打印第一个多行字符串

print(source\_path[0])$sourcePath = "C:\Users\Administrator\AppData\Roaming\Listary\UserProfile\Settings\PathHistory.json"

$destinationPath = "E:\备份盘\8000\_大文件夹\009\_备份文件夹\_自\005\_209\_Listary!2\_设置\自定义设置\PathHistory.json"

Copy-Item -Path $sourcePath -Destination $destinationPath -Force

# 复制单个文件（无需-Recurse参数）

Copy-Item -Path $sourcePath -Destination $destinationPath -Force

# 如果需要复制整个目录（包括子项），使用以下命令

# Copy-Item -Path "C:\源目录" -Destination "E:\目标目录" -Recurse -Force if source\_name and not lang:

base\_name = os.path.basename(source\_name)

# 检查源文件名中是否包含常见语言标识

if 'py' in base\_name or 'python' in base\_name.lower():

lang = 'py'

elif 'ps' in base\_name or 'powershell' in base\_name.lower():

lang = 'ps'

else:

lang = '' # 无匹配时留空

# 提取前缀（优先使用参数，其次从源文件名提取）

if source\_name:

parts = base\_name.split('\_')

prefix = prefix or (parts[0] if parts else "")

prefix = prefix.replace('\_', '\_') # 将连字符转换为下划线

prefix = '\_'.join(filter(None, prefix.split())) # 处理多个连字符或空格

# 提取版本号（优先使用参数，其次从源文件提取）

if source\_name and not version:

parts = base\_name.split('\_')

for part in parts:

if 'v' in part and '.' in part:

version = part[1:]

elif '.' in part and all(p.isdigit() for p in part.split('.')):

version = part

# 提取日期（严格匹配8位数字）

if source\_name and not (year and month and day):

parts = base\_name.split('\_')

for part in parts:

if len(part) == 8 and part.isdigit():

year, month, day = part[:4], part[4:6], part[6:8]

break

# 日期处理（优先使用参数，其次从源文件提取，最后用当前日期）

date\_part = f"{year}{month.zfill(2)}{day.zfill(2)}" if all([year, month, day]) else datetime.now().strftime("%Y%m%d")

version = version or "1.0" # 版本号默认值

# 直接使用传入的 version 参数，去除格式转换逻辑

version\_formatted = version

# 组合文件名组件（仅当lang有效时包含语言部分）

components = filter(None, [prefix, lang, f"v{version\_formatted}", date\_part])

return "\_".join(components)

\_——————————\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

def generate\_filename(prefix="", lang="", version="", year="", month="", day="", source\_name=None):

"""标准化文件名生成（增强源文件信息提取，优化语言标识优先级）"""

# 从源文件提取语言标识（优先于参数）

if source\_name and not lang:

base\_name = os.path.basename(source\_name)

# 检查源文件名中是否包含常见语言标识

if 'py' in base\_name or 'python' in base\_name.lower():

lang = 'py'

elif 'ps' in base\_name or 'powershell' in base\_name.lower():

lang = 'ps'

else:

lang = '' # 无匹配时留空

# 提取前缀（优先使用参数，其次从源文件名提取）

if source\_name:

parts = base\_name.split('\_')

prefix = prefix or (parts[0] if parts else "")

prefix = prefix.replace('\_', '\_') # 将连字符转换为下划线

prefix = '\_'.join(filter(None, prefix.split())) # 处理多个连字符或空格

# 提取版本号（优先使用参数，其次从源文件提取）

if source\_name and not version:

parts = base\_name.split('\_')

for part in parts:

if 'v' in part and '.' in part:

version = part[1:]

elif '.' in part and all(p.isdigit() for p in part.split('.')):

version = part

# 提取日期（严格匹配8位数字）

if source\_name and not (year and month and day):

parts = base\_name.split('\_')

for part in parts:

if len(part) == 8 and part.isdigit():

year, month, day = part[:4], part[4:6], part[6:8]

break

# 日期处理（优先使用参数，其次从源文件提取，最后用当前日期）

date\_part = f"{year}{month.zfill(2)}{day.zfill(2)}" if all([year, month, day]) else datetime.now().strftime("%Y%m%d")

version = version or "1.0" # 版本号默认值

# 直接使用传入的 version 参数，去除格式转换逻辑

version\_formatted = version

# 组合文件名组件（仅当lang有效时包含语言部分）

components = filter(None, [prefix, lang, f"v{version\_formatted}", date\_part])

return "\_".join(components)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1在version参数中版本格式转换逻辑中强行加入下划线"\_"

2version参数用第2张图片中的写法

3得到第3张图片中效果

4解决报错并给出完整代码

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2在第1张图片中\_\_lang参数留空脚本执行成功重命名

3在第2张图片中出现文件"old\_py\_v2\_0\_20250622"多出语言类型"py"报错

4在重命名文件操作中中\_\_lang参数默认值更改为最优先继承源文件

5解决报错并给出完整代码

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# 替换 [自定义路径] 为你想安装的实际目录，比如 D:\my\_project\lib

C:\Users\Administrator\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundation.Python.3.13\_qbz5n2kfra8p0\python.exe \_m pip install \_\_target=[自定义路径] psutil

E:\备份盘\8000\_大文件夹\009\_备份文件夹\_自\005\_211\_Ditto!2\_数据备份\005\_211\_Ditto!2\_数据备份.db

Python

005\_211\_Ditto!2\_数据备份

005\_420\_VS Code

8911586460流2

008\_人工智能(AGI)E:\备份盘\带0文件夹\005\_计算机科学、程式、资料,硬件\005\_210\_电脑软件!1\_使用手册\6.00\_\_技术\_应用科学\_电\6.55\_网络工具\_电\6.55!D\_Download

xcopy文件\_本地覆盖云端配置\_T0

E:\备份盘\带0文件夹\005\_计算机科学、程式、资料,硬件\005\_400\_电脑编程!1\BAT\云端\_本地配置\_互传\_BAT\xcopy文件\_本地覆盖云端配置\_T0.bat

"E:\备份盘\带0文件夹\005\_计算机科学、程式、资料,硬件\005\_400\_电脑编程!1\BAT\云端\_本地配置\_互传\_BAT\005\_411\_PowerShell ISE\_云端覆盖本地配置\_T0\_bat.bat"

云端\_本地配置\_互传\_BATE:\备份盘\带0文件夹\005\_计算机科学、程式、资料,硬件\005\_400\_电脑编程!1\BAT\云端\_本地配置\_互传\_BATE:\备份盘\带0文件夹\005\_计算机科学、程式、资料,硬件\005\_400\_电脑编程!1\BAT\云端\_本地配置\_互传\_BAT

"D:\下载中\360 极速浏览器X.lnk"

M005\_212\_QTTabBar

# 创建单个空文件

New\_Item \_Path "E:\备份盘\带0文件夹\005\_计算机科学、程式、资料,硬件\005\_400\_电脑编程!1\BAT\example.txt"

# 简化语法（ni 是 New\_Item 的别名）

ni \_Path "C:\Temp\newfile.csv" \_Type File

# 批量创建多个文件

1..5 | ForEach\_Object { New\_Item \_Path "C:\Temp\file$\_.txt" \_ItemType File }

Set\_ExecutionPolicy \_ExecutionPolicy RemoteSigned \_Scope CurrentUser

E:\备份盘\带0文件夹\005\_计算机科学、程式、资料,硬件\005\_400\_电脑编程!1\BAT\云端\_本地配置\_互传\_BAT\ xcopy文件\_云端覆盖本地配置\_T0.bat

xcopy "" /S /E /Y /I

005\_411\_PowerShell ISE命令

CurrentUserC:\Users\Administrator\AppData\Roaming\Microsoft\Office\Recent

C:\Users\Administrator\AppData\Roaming\Microsoft\Windows\Recent

800\_文学&100\_哲学与心理学

300\_社会科学

400\_语言

600\_技术应用科学

700\_艺术与休闲

800\_文学

900\_历史、地理与传记

## 2关闭txt文档

1不触发保存选项对话框,强制保存多个 TXT文档

2同时关闭全部多个TXT文档

3增加一段代码

4保留原有全部功能

## 3临时脚本

import win32gui

import win32con

import time

def force\_save\_and\_close\_all\_txt():

txt\_hwnds = []

def callback(hwnd, extra):

if win32gui.IsWindowVisible(hwnd):

title = win32gui.GetWindowText(hwnd)

if title.endswith(".txt - 记事本"):

txt\_hwnds.append(hwnd)

return True

win32gui.EnumWindows(callback, None)

for hwnd in txt\_hwnds:

# 模拟保存

win32gui.SendMessage(hwnd, win32con.WM\_KEYDOWN, win32con.VK\_CONTROL, 0)

win32gui.SendMessage(hwnd, win32con.WM\_KEYDOWN, ord('S'), 0)

win32gui.SendMessage(hwnd, win32con.WM\_KEYUP, ord('S'), 0)

win32gui.SendMessage(hwnd, win32con.WM\_KEYUP, win32con.VK\_CONTROL, 0)

time.sleep(0.5)

# 模拟关闭

win32gui.SendMessage(hwnd, win32con.WM\_SYSCOMMAND, win32con.SC\_CLOSE, 0)

time.sleep(0.3) # 给关闭操作留时间

print(f"已尝试处理 {len(txt\_hwnds)} 个 TXT 窗口")

# 005\_

Life

Hotkey

Apply Last Search

Ctrl + C

Copy Selection

Decrease Transparency %

Delete all non used clips

EMail, Clip Export As Attachment

EMail, Content As Attachment

EMail, Content In Body

Shift + Esc

Force Close Window

Gmail

Increase Transparency %

Invert Case

# 005\_400\_电脑编程

005\_410\_PowerShell

005\_420\_VS\_Code

005\_209\_Listary

005\_212\_QTTabBar

005\_210\_电脑软件

005\_210\_电脑软件

# 005\_420\_VS\_Code

git config --global user.name "kid1997"

git config --global user.email "kid1997@139.com"

——————————\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

005\_420\_VS Code

005\_410\_PowerShell

005\_411\_PowerShell ISE \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## 扩展

这些是用来增强 VS Code 使用体验的扩展插件，每条含义如下：

表格

复制

扩展名 功能说明

QuickDraw 快速访问常用命令（快捷命令面板增强）

Open I 在 VS Code 内智能打开 URL（网页预览）

Qwen 接入阿里巴巴的「通义千问」大模型

Set Workspace Extension 设置工作区扩展（可能是配置插件）

Unzip 直接在 VS Code 中解压 zip 文件

Deepseek Web 在 VS Code 中使用 Deepseek 聊天功能

LLMText 把网页内容转为 Markdown 格式，供 LLM（如 Perplexity）使用

o3 Mini 在 VS Code 中使用 OpenAI 的 o3-mini 模型

# 005\_430\_Python

# 融合后的列表：包含原多行字符串 + 处理后的路径字符串

source\_path = [

r'''C:\Users\Administrator\AppData\Roaming\Listary\UserProfile

\Settings\PathHistory.json'''.replace('\n', ''), # 源文件路径

r'''C:\Users\Administrator\AppData\Roaming\Listary\UserProfile\Settings

\Preferences.json'''.replace('\n', ''),

r'''C:\Users\Administrator\AppData\Roaming\Listary\UserProfile\Settings

\SearchHistory.json'''.replace('\n', ''),

r'''C:\Users\Administrator\AppData\Roaming\IrfanView

\i\_view64.ini'''.replace('\n', ''),

r'''C:\Users\Administrator\AppData\Roaming\Ditto

\005\_211\_Ditto!2\_数据备份.db'''.replace('\n', '')

]

dest\_dir = [

r'''E:\备份盘\8000\_大文件夹\009\_备份文件夹\_自\005\_209\_Listary!2\_设置

\自定义设置\\'''.replace('\n', ''), # 目标文件夹路径

r'''E:\备份盘\8000\_大文件夹\009\_备份文件夹\_自

\005\_238\_irfanview!2\_设置备份\\'''.replace('\n', ''),

r'''E:\备份盘\8000\_大文件夹\009\_备份文件夹\_自

\005\_211\_Ditto!2\_数据备份\\'''.replace('\n', '')

]

# 验证结果（可选）

print("列表长度：", len(source\_path)) # 输出 3，说明包含3个元素

print("路径字符串：", source\_path[1]) # 输出处理后的完整路径（无换行）

print("路径字符串：", dest\_dir[0]) # 输出处理后的完整路径（无换行）

j= print("列表长度：", len(source\_path))

for i in range(1, j):

# 执行复制操作，开启强制覆盖功能

copy\_file(source\_path[i-1], dest\_dir[i-1], overwrite=True)

## 关闭word文档并保存

import win32com.client

import time

import os

import subprocess

def close\_word\_with\_save\_option(file\_path=None):

r"""

关闭Word文档并触发保存选项对话框，之后执行关机操作。

"""

# 启动Word应用程序

word\_app = win32com.client.Dispatch("Word.Application")

# 显示Word窗口（若需要后台操作可设为False）

word\_app.Visible = True

try:

# 打开指定文档（若未打开）

if file\_path:

doc = word\_app.Documents.Open(FileName=file\_path)

# 等待文档加载

time.sleep(1)

else:

# 若未指定路径，操作当前激活的文档

doc = word\_app.ActiveDocument

# 关闭文档：参数SaveChanges=-2是wdPromptToSaveChanges，会触发保存提示对话框

# 原代码中-1是wdDoNotSaveChanges，已修正为-2以正确触发保存提示

doc.Close(SaveChanges=-2)

print("文档已关闭，保存选项已触发")

except Exception as e:

print(f"操作失败：{str(e)}")

finally:

# 退出Word应用程序（可选，若需保留其他文档可注释）

word\_app.Quit()

# 执行关机操作

try:

# 设置关机倒计时（秒）

shutdown\_delay = 10

print(f"将在{shutdown\_delay}秒后关机...")

print("若要取消关机，请打开命令提示符并输入：shutdown /a")

# 执行关机命令

subprocess.run(

f"shutdown /s /t {shutdown\_delay}",

shell=True,

check=True

)

except subprocess.CalledProcessError as e:

print(f"关机命令执行失败：{str(e)}")

# 示例：关闭指定路径的Word文档

if \_\_name\_\_ == "\_\_main\_\_":

# 替换为你的Word文档路径

doc\_path = r"" # 例如：r"C:\Documents\example.docx"

close\_word\_with\_save\_option(doc\_path)

## 关闭txt文本

def force\_save\_and\_close\_notepad():

# 记事本窗口类名

NOTEPAD\_CLASS = "Notepad"

txt\_hwnds = []

# 枚举所有记事本窗口

def enum\_callback(hwnd, extra):

if win32gui.GetClassName(hwnd) == NOTEPAD\_CLASS:

txt\_hwnds.append(hwnd)

return True

win32gui.EnumWindows(enum\_callback, None)

print(f"找到 {len(txt\_hwnds)} 个记事本窗口")

for hwnd in txt\_hwnds:

title = win32gui.GetWindowText(hwnd)

print(f"处理窗口: {title} (句柄: {hwnd})")

try:

# 1. 激活窗口

win32gui.SetForegroundWindow(hwnd)

win32gui.SetActiveWindow(hwnd)

time.sleep(0.3)

# 2. 强制保存 - 使用Ctrl+S快捷键

# 发送Ctrl+S组合键确保保存

win32api.keybd\_event(win32con.VK\_CONTROL, 0, 0, 0) # 按下Ctrl键

win32api.keybd\_event(ord('S'), 0, 0, 0) # 按下S键

time.sleep(0.1) # 等待按键响应

win32api.keybd\_event(

ord('S'), 0, win32con.KEYEVENTF\_KEYUP, 0) # 释放S键

win32api.keybd\_event(win32con.VK\_CONTROL, 0,

win32con.KEYEVENTF\_KEYUP, 0) # 释放Ctrl键

time.sleep(0.5) # 等待保存完成

# 3. 获取编辑控件并清空撤销缓冲区，避免关闭时提示

edit\_hwnd = win32gui.FindWindowEx(hwnd, None, "Edit", None)

if edit\_hwnd:

# 清空撤销缓冲区，让记事本认为没有未保存的更改

win32api.SendMessage(

edit\_hwnd, win32con.EM\_EMPTYUNDOBUFFER, 0, 0)

# 设置 Modified 标志为 False

win32api.SendMessage(edit\_hwnd, win32con.EM\_SETMODIFY, 0, 0)

time.sleep(0.2)

# 4. 发送关闭消息

result = win32api.SendMessage(hwnd, win32con.WM\_CLOSE, 0, 0)

if result == 0:

# 备用：通过进程关闭（针对无响应情况）

pid = win32api.GetWindowThreadProcessId(hwnd)[1]

try:

p = psutil.Process(pid)

p.terminate()

print(f"强制终止进程 {pid}")

except Exception as e:

print(f"终止进程失败: {e}")

time.sleep(0.3)

except Exception as e:

print(f"处理窗口时出错: {e}")

continue

print(f"已处理 {len(txt\_hwnds)} 个记事本窗口")

if \_\_name\_\_ == "\_\_main\_\_":

force\_save\_and\_close\_notepad()

## 合并后

import win32com.client

import win32gui

import win32con

import win32api

import psutil

import time

import pythoncom

import threading # 新增：确保线程独立性

def force\_save\_and\_close\_all\_word():

"""强制保存并关闭所有打开的Word文档(不显示保存对话框）"""

word\_app = None

try:

word\_app = win32com.client.GetActiveObject("Word.Application")

except win32com.client.DispatchException:

try:

word\_app = win32com.client.Dispatch("Word.Application")

except Exception as e:

print(f"创建或获取Word应用实例失败: {str(e)}")

return

except Exception as e:

print(f"获取Word应用实例时发生错误: {str(e)}")

return

try:

word\_app.Visible = True

pythoncom.CoInitialize() # 初始化COM线程

# 从最后一个文档开始关闭

for i in range(word\_app.Documents.Count, 0, -1):

try:

doc = word\_app.Documents(i)

doc.Close(SaveChanges=-1) # 强制保存关闭

print(f"已关闭Word文档: {doc.Name}")

time.sleep(0.5) # 延长延迟确保完成

except Exception as doc\_e:

print(f"处理文档出错: {str(doc\_e)}")

# 关闭Word应用

word\_app.Quit()

print("Word程序已退出")

except Exception as e:

print(f"Word操作错误: {str(e)}")

finally:

if word\_app:

del word\_app

pythoncom.CoUninitialize() # 释放COM线程

time.sleep(1) # 新增：等待Word完全退出，释放系统资源

def force\_save\_and\_close\_notepad():

"""修复后的记事本处理函数：增强窗口激活和关闭可靠性"""

NOTEPAD\_CLASS = "Notepad"

txt\_hwnds = []

# 枚举所有记事本窗口（确保线程状态正常）

def enum\_callback(hwnd, extra):

if win32gui.IsWindowVisible(hwnd) and win32gui.GetClassName(hwnd) == NOTEPAD\_CLASS:

txt\_hwnds.append(hwnd)

return True

# 关键修复1：确保枚举窗口时线程无COM干扰

win32gui.EnumWindows(enum\_callback, None)

print(f"找到 {len(txt\_hwnds)} 个记事本窗口（可见状态）")

for hwnd in txt\_hwnds:

title = win32gui.GetWindowText(hwnd)

print(f"处理窗口: {title} (句柄: {hwnd})")

try:

# 关键修复2：增强窗口激活（先还原窗口，再获取焦点）

win32gui.ShowWindow(hwnd, win32con.SW\_RESTORE) # 确保窗口不最小化

win32gui.SetForegroundWindow(hwnd) # 激活窗口

win32gui.SetActiveWindow(hwnd)

time.sleep(0.5) # 延长激活等待时间

# 强制保存（Ctrl+S）

win32api.keybd\_event(win32con.VK\_CONTROL, 0, 0, 0)

win32api.keybd\_event(ord('S'), 0, 0, 0)

time.sleep(0.2) # 延长按键等待

win32api.keybd\_event(ord('S'), 0, win32con.KEYEVENTF\_KEYUP, 0)

win32api.keybd\_event(win32con.VK\_CONTROL, 0, win32con.KEYEVENTF\_KEYUP, 0)

time.sleep(1) # 关键修复3：延长保存等待（确保保存完成）

# 清空撤销缓冲区，避免关闭提示

edit\_hwnd = win32gui.FindWindowEx(hwnd, None, "Edit", None)

if edit\_hwnd:

win32api.SendMessage(edit\_hwnd, win32con.EM\_EMPTYUNDOBUFFER, 0, 0)

win32api.SendMessage(edit\_hwnd, win32con.EM\_SETMODIFY, 0, 0)

time.sleep(0.3)

# 关键修复4：增强关闭逻辑（先发送关闭消息，再检查是否真的关闭）

# 发送WM\_CLOSE

win32api.SendMessage(hwnd, win32con.WM\_CLOSE, 0, 0)

time.sleep(0.5) # 等待关闭响应

# 检查窗口是否仍存在

if win32gui.IsWindow(hwnd):

# 强制终止进程

pid = win32api.GetWindowThreadProcessId(hwnd)[1]

p = psutil.Process(pid)

p.terminate()

print(f"记事本窗口未响应，已强制终止进程 {pid}")

else:

print(f"记事本窗口 {title} 已正常关闭")

time.sleep(0.3)

except Exception as e:

print(f"处理记事本出错: {str(e)}")

print(f"已处理 {len(txt\_hwnds)} 个记事本窗口")

if \_\_name\_\_ == "\_\_main\_\_":

print("开始处理Word文档...")

force\_save\_and\_close\_all\_word()

print("\n开始处理记事本窗口...")

force\_save\_and\_close\_notepad() # 此时系统资源已释放，记事本处理不受干扰

print("\n所有操作已完成")

## 使用方法

以下是该代码的详细使用方法：

### 一、功能说明

该代码用于通过Python操作Word应用程序，实现关闭指定Word文档并根据设置触发保存操作的功能。

### 二、环境准备

1. \*\*安装依赖库\*\*：代码使用了`win32com.client`模块，需要先安装`pywin32`库，安装命令如下：

```bash

pip install pywin32

```

2. \*\*运行环境\*\*：仅支持Windows系统（因依赖Windows的COM组件），且需在已安装Microsoft Word的电脑上运行。

### 三、代码参数与配置

1. \*\*核心函数\*\*：`close\_word\_with\_save\_option(file\_path=None)`

- 参数`file\_path`：可选参数，传入Word文档的完整路径（如`r"E:\文档\示例.docx"`）。若不传入，则操作当前激活的Word文档（即当前正在编辑的文档）。

def close\_word\_with\_save\_option(file\_path=None) 中，file\_path=None 表示给参数 file\_path 设置了一个默认值—— 当调用这个函数时，如果不传入 file\_path 的值，它会自动使用 None（空值）作为默认值。

2. \*\*保存选项说明\*\*：代码中`doc.Close(SaveChanges=-1)`的`-1`对应Word的“自动保存”模式（等同于`wdDoNotSaveChanges`为`0`、`wdPromptToSaveChanges`为`-2`，这里`-1`表示直接保存修改）。

检查SaveChanges参数：

-1：强制保存（不提示）

-2：提示保存（原代码注释描述与实际参数不符，若需要提示保存应改为-2）

0：不保存

### 四、使用步骤

1. \*\*修改文档路径（可选）\*\*：

- 若要关闭指定文档，在代码底部的`if \_\_name\_\_ == "\_\_main\_\_":`区块中，将`doc\_path`的值修改为目标文档的完整路径，例如：

```python

doc\_path = r"C:\Users\用户名\Documents\我的文档.docx"

```

- 若要关闭当前正在打开的文档，可将`doc\_path`留空，调用函数时不传参数：`close\_word\_with\_save\_option()`

2. \*\*运行代码\*\*：直接运行该Python脚本，程序会自动启动Word（若未启动）、打开指定文档（若传入路径），然后执行关闭并保存操作。

### 五、注意事项

1. \*\*文档路径格式\*\*：路径字符串前需加`r`（表示原始字符串），避免路径中的反斜杠`\`被解析为转义字符。

2. \*\*权限问题\*\*：确保目标文档未被其他程序锁定，且当前用户有读写该文档的权限。

3. \*\*Word窗口显示\*\*：代码中`word\_app.Visible = True`会显示Word窗口，若需后台操作，可改为`False`。

4. \*\*异常处理\*\*：若操作失败（如路径错误、文档损坏等），会打印错误信息，可根据提示排查问题。

### 六、示例

关闭`E:\工作\报告.docx`文档并保存：

```python

if \_\_name\_\_ == "\_\_main\_\_":

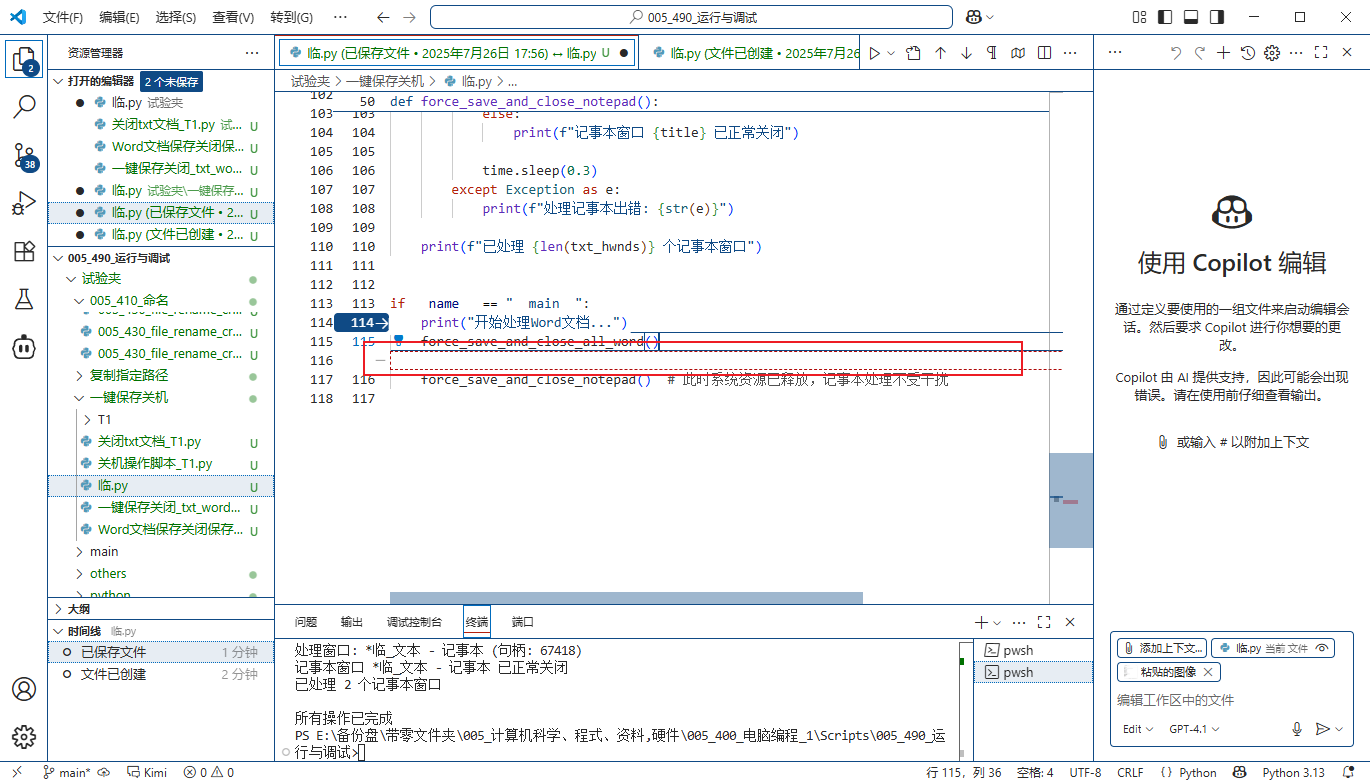
doc\_path = r"E:\工作\报告.docx"

close\_word\_with\_save\_option(doc\_path)

```

运行后，程序会自动处理该文档并关闭Word应用程序。

# 报错及解决办法



红框处的虚线是代码错误提示线

2. 语法缩进错误

删除行后，下面代码的缩进可能乱了（比如和上面 print 语句没对齐 ），Python 对缩进敏感，会触发语法检查。

解决：把 force\_save\_and\_close\_notepad() 这行的缩进，调整成和 print("开始处理Word文档...") 一样（一般是 4 个空格或 1 个 Tab ）。

3. 缓存的错误提示

有时候 VS Code 没及时刷新，删掉代码后还显示旧错误。

解决：按 Ctrl + S 保存文件，或者重启 VS Code 重新加载，红框会重新校验，没问题就消失啦～

简单说，红框是语法 / 依赖报错，不是代码没删完，

# 临

1不触发保存选项对话框,强制保存多个 TXT文档

2同时关闭全部多个TXT文档

3增加一段代码

4保留原有全部功能

# end