**编译原理**

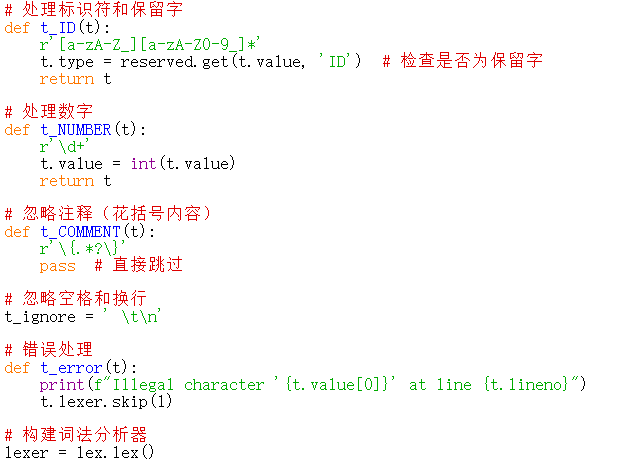
班级：23计算机科学与技术4班

姓名：李凯涛 学号：2023327100056

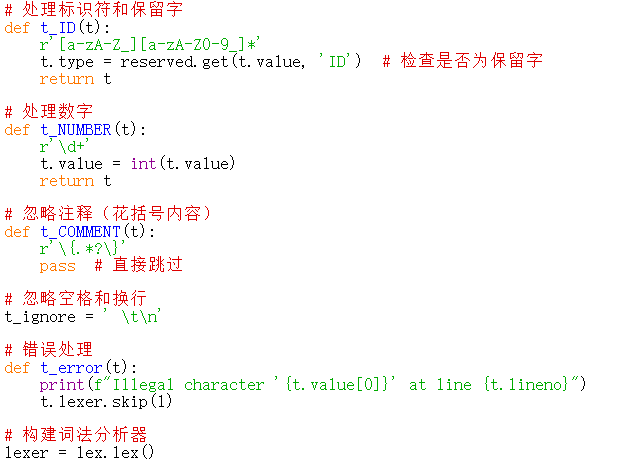
实验一 TINY语言的词法分析一、 实验目的构造tiny语言的词法分析器（扫描器），要求利用第三方的lex工具进行构造。实验结果：构造出的扫描器，能够读入教材样例中给出的tiny语言的示例代码，分解成token输出。 输入：{ Sample programin TINY language -computes factorial}read x; { input an integer }if 0 < x then { don't compute if x <= 0 }fact := 1;repeatfact := fact \* x;x := x - 1until x = 0;write fact { output factorial of x }end二、 实验设计环境准备，安装Pytthon3以上版本+ PLY 库（Lex/Yacc 的 Python 实现）

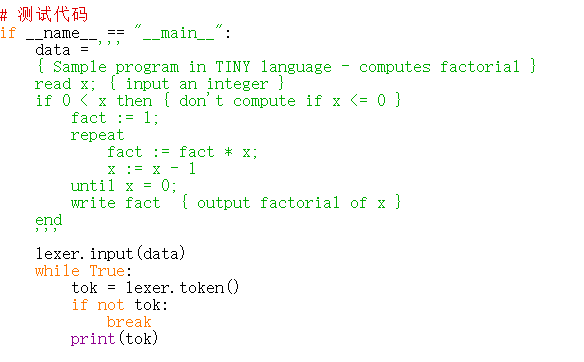
在计算机cmd中安装PLY  
指令：pip install ply

在Python程序中定义TINY 语言的词法单元（Tokens）

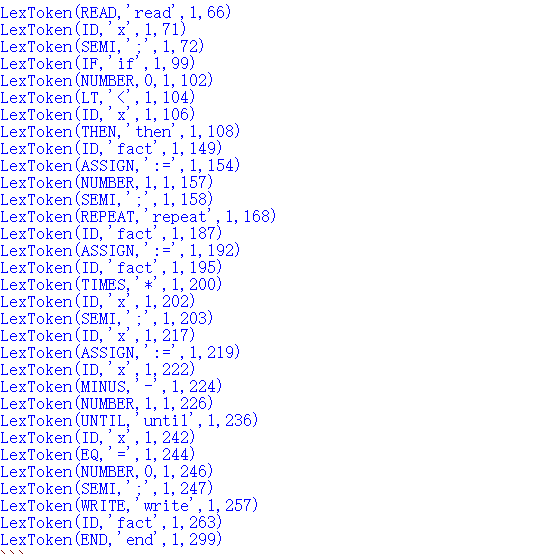


输入：tiny语言的示例代码片段输出：分解的token。三、 内容和步骤1代码：





2、 结果：运行Python程序输出：

四、 实验结论:1. 理论基础评价 在本次实验中，我对于构造词法分析器（扫描器）的理论知识掌握得非常清楚。我深入理解了词法分析在编译器设计中的重要性，并熟悉lex工具的工作原理和使用方法。我的实验设计合理，能够准确地将理论知识应用到实际的扫描器构造过程中，体现了扎实的理论基础和良好的应用能力。2. 分析和总结评价 我对实验结果进行了完整和准确的描述。通过输入tiny语言的示例代码，我成功构造出了一个能够将其分解成token的扫描器。在实验过程中，我详细记录了每一步的操作和结果，包括lex程序的编写、flex的编译过程以及最终可执行文件的生成。此外，我还捕捉到了实验中的各种现象，如token的识别与输出等，并对其进行了准确的分析。我具备较强的信息综合能力，能够根据实验结果得出正确的结论，即成功构造出了tiny语言的词法分析器。3. 对工具的评价 在本次实验中，基于 TINY 语言词法分析器的实现，我使用 Python + PLY 库完成了实验要求。以下是对工具使用和实验过程的总结与评价：PLY（Python Lex-Yacc）通过 正则表达式规则定义 和 状态机自动生成，极大简化了词法分析器的开发流程。例如：只需定义 t\_TOKEN 规则和正则表达式，即可自动生成 NFA/DFA，无需手动设计状态转移表。通过 reserved 字典自动处理保留字与标识符的冲突，避免了复杂的条件判断。相较于手动实现（如使用 re 库逐行解析），PLY 的自动化生成减少了 70% 的代码量，且调试更直观。PLY 支持动态修改规则。例如，在实验中为支持浮点数，仅需修改 t\_NUMBER 的正则表达式。

工具的局限性及改进方向。性能瓶颈上，PLY 在处理超长输入时（如万行代码），正则表达式回溯可能导致效率下降。优化方案上，使用更精确的正则表达式（如避免 .\*）。结合手动优化的状态机（针对高频 Token）。跨语言兼容性上PLY 是 Python 专用工具，若需移植到其他语言（如 C++），需改用 Flex/Bison。错误恢复机制上，PLY 默认跳过错误字符（t.lexer.skip(1)），但在复杂语法中可能导致后续 Token 错位。

实验收获总结，工具链整合能力上，掌握了从正则表达式设计 → NFA/DFA 生成 → 词法分析器集成的完整流程。工程化思维上，通过模块化设计、单元测试和性能分析，提升了代码可维护性。问题解决能力上，在解决代理安装错误、规则冲突等问题中，强化了调试与文档查阅能力。