# 程序设计报告

## 总体设计

1. GrammarNode 作为语法树的节点
   1. GrammarNode.\_\_init\_\_() 解析该语句
   2. GrammarNode.indent 该语句缩进的空格数量
   3. GrammarNode.command 该语句执行的动作
   4. GrammarNode.value 该语句的动作对象（如有）
   5. GrammarNode.child 该语句的第一个子节点
   6. GrammarNode.parent该语句的父亲节点
   7. GrammarNode.brother 该语句的下一个兄弟
2. Parser 读取脚本文件，构建语法树并解释执行
   1. Parser.\_\_init\_\_() 读取脚本文件，构建语法树，使用栈结构根据缩进判断语句在语法树中的层级，每新建一个语法树节点就初始化其兄弟和父亲节点信息
   2. Parser.execute() 根据语法树中的语句执行脚本，判断关键字执行响应动作或跳转
   3. Parser.text 读入的脚本文件
   4. Parser.parse\_tree 根据脚本文件构建的语法树

## 源程序清单

源程序：

DSLParser.py

前端页面：

templates/

static/

server.py

示例脚本：

shopping.script

torture.script

test.script

测试脚本：

test.py

## 程序测试

测试环境：Windows10 WSL (Ubuntu 20.04.1 LTS)

测试脚本：test.py

测试用例：test.script

测试的功能：输入输出，分支控制

测试结果：

## 记法说明

变量：以$符号开头的数字（只能在input语句中定义），示例：$1，$2

常量：以双引号开头、结尾的字符串，示例：”这是一个示例”

注释：以#号开头的行，示例：# 这是一行注释

输入：关键字input后加变量名，以空格分隔，示例：input $1

输出：关键字print或println后加变量名，以空格分隔，示例：print “here”

终止：关键字end结束执行

重试：关键字retry重新执行该脚本

分支开始：关键字switch后加变量名，以空格分隔，表示以该变量的值作为判断条件，示例：switch $1

分支判断：关键字case后加变量名或常量名或other，以空格分隔，表示当switch中的变量值与该值匹配时执行case中的语句，示例：case $1，case “1”，case other

代码结构：以4个空格为单位的缩进，缩进后的代码为缩进之前代码的子语句，switch与case，case与执行代码块之间均需要缩进

示例：请见示例脚本