**题目要求：**

1. **利用PLY实现简单的Python程序的解析**
2. **示例程序位于example/**

**（2）需要进行解析的文件为example.py**

**（3）需要完成以下内容的解析**

* **赋值语句**
* **完整的四则运算**
* **print语句**

**四则运算的无二义性下文法大致如下：**

**expr -> expr + term | term**

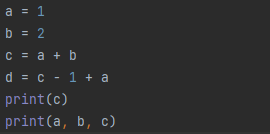
**term -> term \* factor | factor**

**factor -> id | (expr)**

**（不需要消除二义性）**

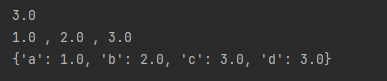
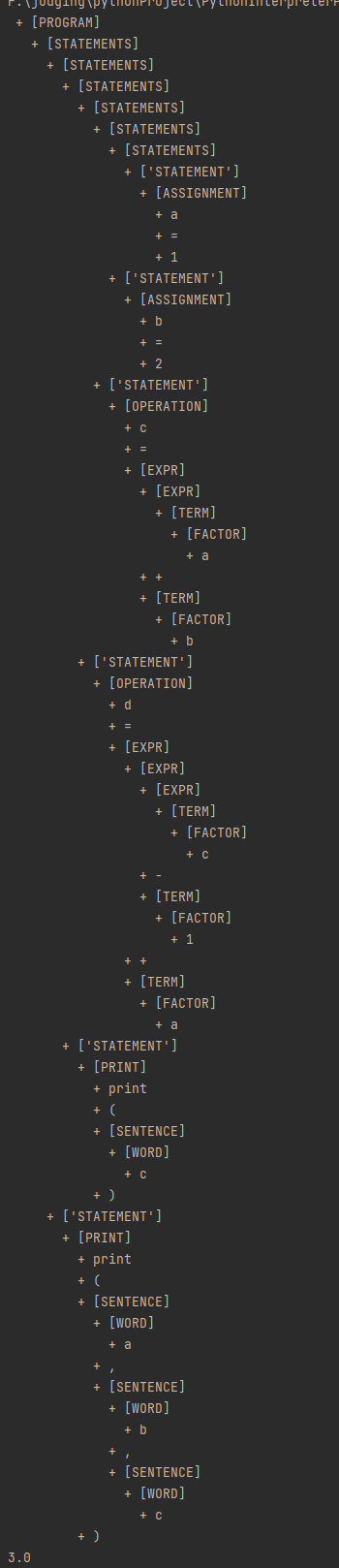
**（4）解析结果以语法树的形式呈现**

**example.py文件内容如下：**



**程序说明：**

1. 打开main.py文件，确保source中的所有代码在同一目录下
2. 确保已经安装了PLY库
3. 运行main.py文件
4. 对example.py文件中的程序段进行解析，结果以语法树的形式展现，并展示print的结果以及所有变量的最终值字典，解析结果如下图：

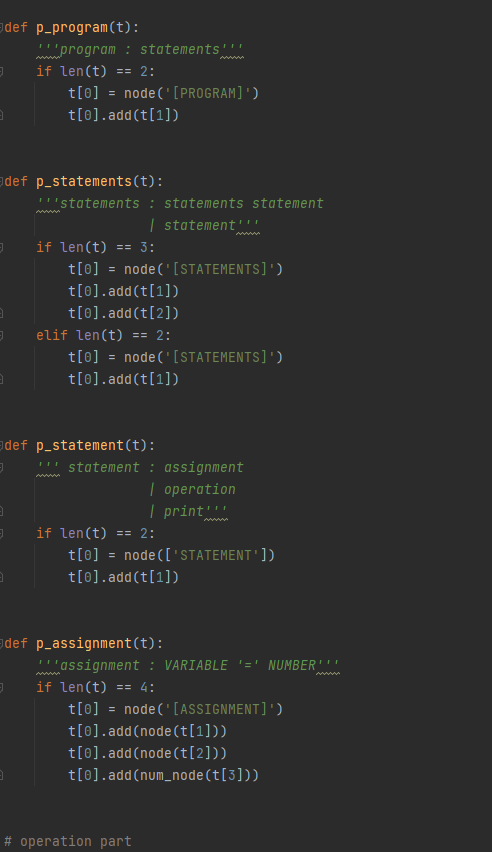


（第一张图是语法树的结构，具体的结点含义后文讲解。第二张图是print的结果，也就是c的最终值，还有a, b, c的最终值，以及所有变量的最终值）

显然解析结果和题目中需要完成的测试结果等价。

1. 对yacc程序定义的文法规则的解释

总体结构和assignment部分不多作赘述，结构较为简单，如下图：



Operation部分的文法结构如下：

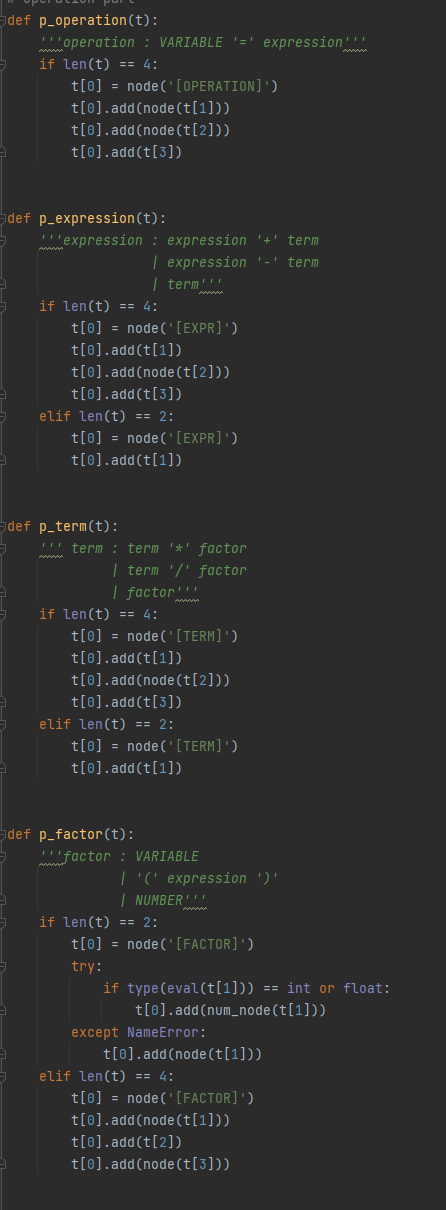
文法：

Operation -> VARIABLE = expression

Expression -> expression + term | expression – term | term

Term -> term \*factor | term / factor | factor

Factor -> NUMBER | VARIABLE | ( expression )



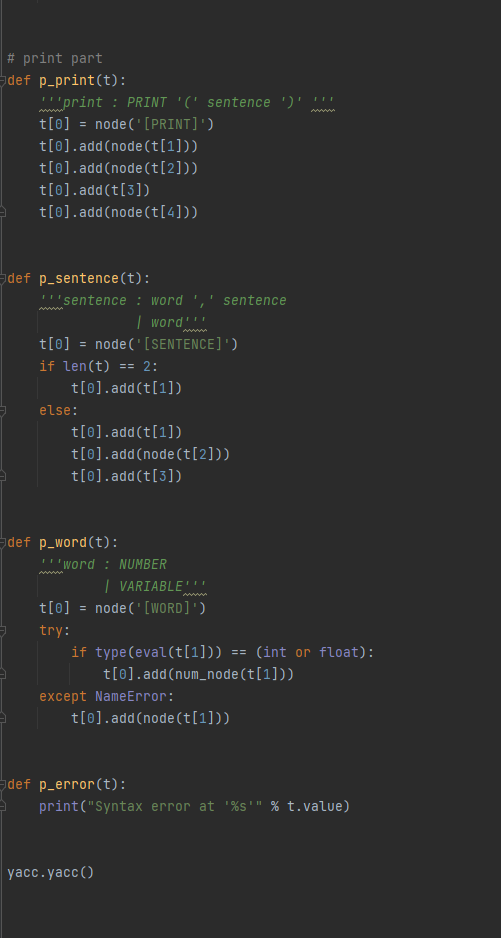
Print部分的文法结构如下图：

文法：

Print -> PRINT ( sentence )

Sentence -> word, sentence | word

Word -> NUMBER | VARIABLE



1. Translation部分

理解各层的node里面应该是什么值就行，比如说operation部分各层的node的value就是当前的运算结果，而print部分sentence层的node的value就是待输出的值组成的列表，然后对各层执行相应的setvalue，更新v\_table等操作即可。