**题目要求：**

**本次学习的语法是选择语句和循环语句，需要注意的是本次使用的语法做了一些改进，不是纯粹的python2语法。**

**需要结合上次课四则运算的解析程序**

1. **示例程序位于example3/**

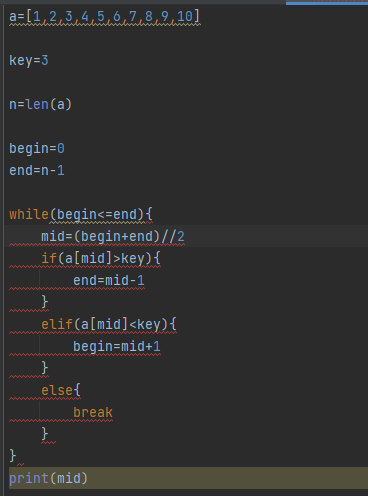
**（2）需要进行解析的文件为binary\_search.py和select\_sort.py，分别对应二分查找和选择排序。**

**（3）需要完成以下内容的解析**

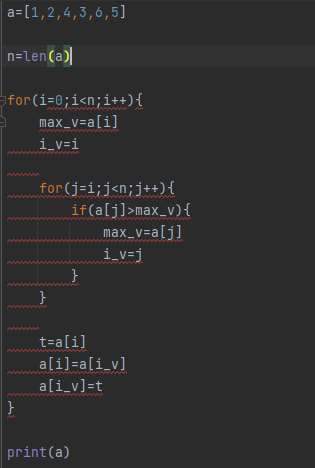
* **if**
* **while**
* **for**

**（4）解析结果以语法树的形式呈现**

**binary\_search.py文件内容如下：**



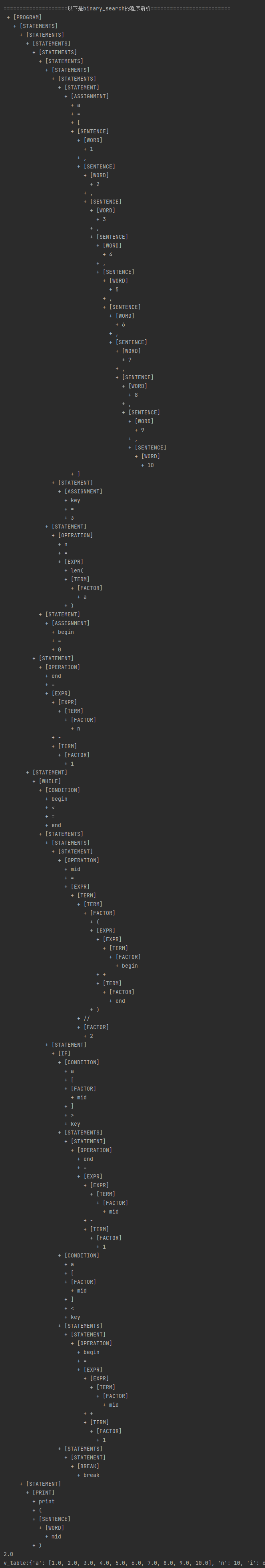
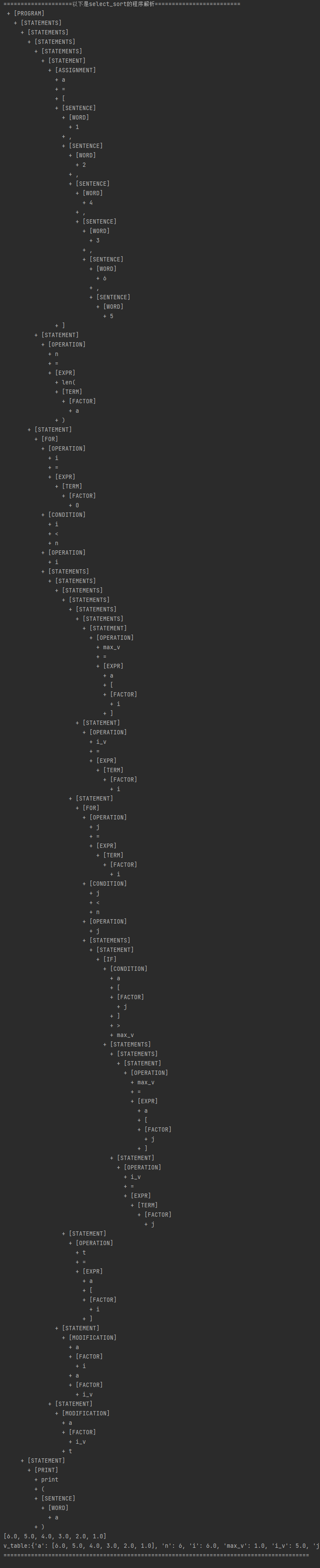
**select\_sort.py文件内容如下：**



**程序说明：**

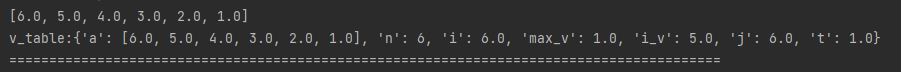
1. 打开main.py文件，确保source中的所有代码在同一目录下
2. 确保已经安装了PLY库
3. 运行main.py文件
4. 对binary\_search.py和select\_sort.py文件中的程序段进行解析，结果以语法树的形式展现，并展示print的结果以及所有变量的最终值字典，解析结果如下图：

第一张图是select\_sort.py的运行结果，第二张是binary\_search.py的运行结果

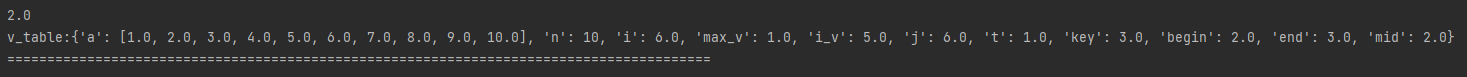


两个程序的print结果以及对应的最后v\_table内容如下：

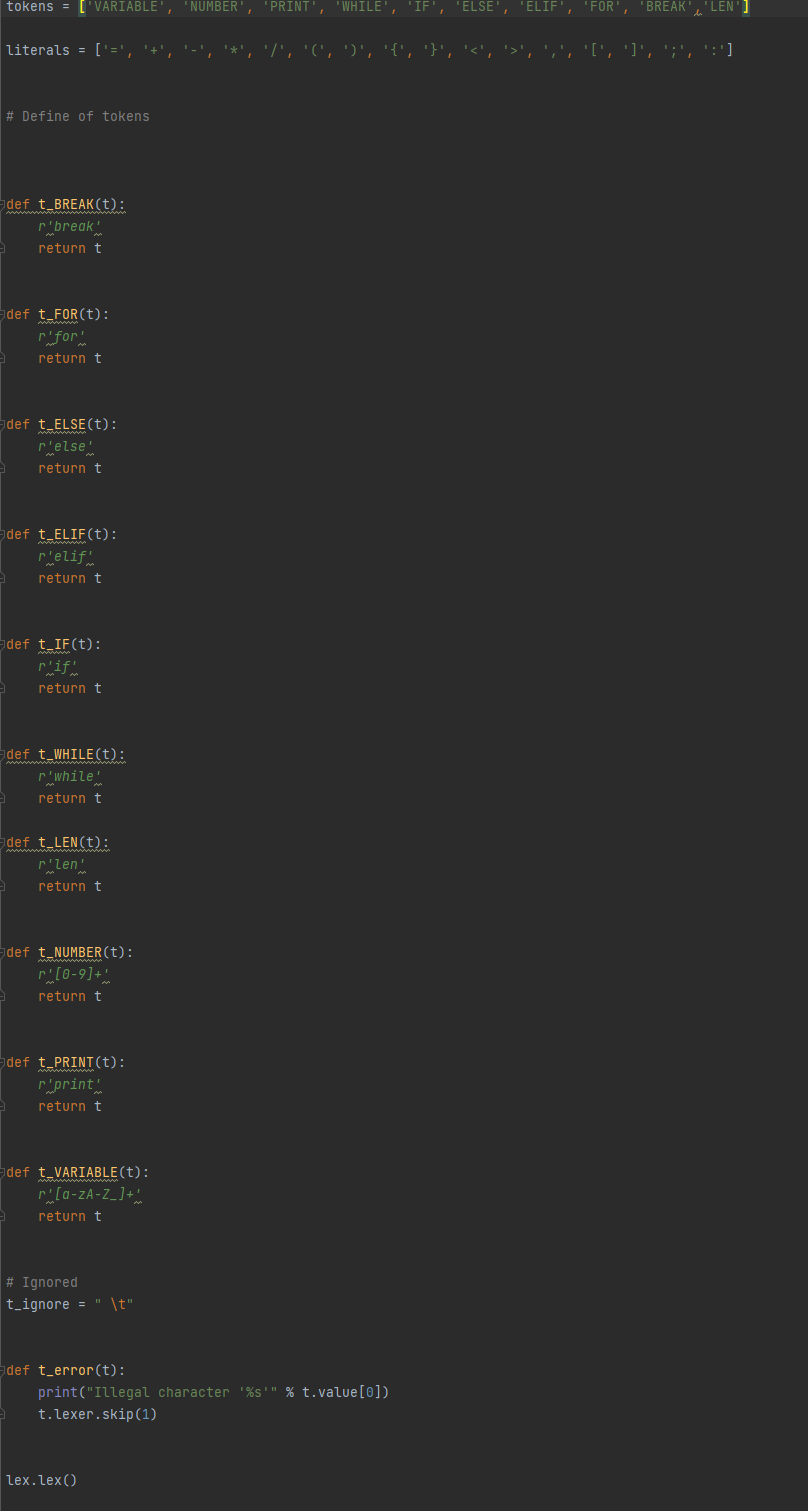
选择排序的结果：



二分查找的结果：



1. 对Lexer程序定义的token规则的解释



定义的token对应到待解析语句中的一些关键字如len()，while，for等的匹配和使用，literals中定义了需要使用的标点符号。

1. Yacc语法规则的设计

设计的语法规则展开后如下所示：

Grammar

Rule 0 S' -> program

Rule 1 program -> statements

Rule 2 statements -> statements statement

Rule 3 statements -> statement

Rule 4 statement -> assignment

Rule 5 statement -> operation

Rule 6 statement -> print

Rule 7 statement -> modification

Rule 8 statement -> iF

Rule 9 statement -> whilE

Rule 10 statement -> for

Rule 11 statement -> break

Rule 12 break -> BREAK statements

Rule 13 break -> BREAK

Rule 14 for -> FOR ( operation ; condition ; operation ) { statements }

Rule 15 condition -> VARIABLE > VARIABLE

Rule 16 condition -> VARIABLE < VARIABLE

Rule 17 condition -> VARIABLE > NUMBER

Rule 18 condition -> VARIABLE < NUMBER

Rule 19 condition -> VARIABLE < = VARIABLE

Rule 20 condition -> VARIABLE [ factor ] > VARIABLE

Rule 21 condition -> VARIABLE [ factor ] < VARIABLE

Rule 22 iF -> IF ( condition ) { statements }

Rule 23 iF -> IF ( condition ) { statements } ELIF ( condition ) { statements } ELSE { statements }

Rule 24 whilE -> WHILE ( condition ) { statements }

Rule 25 assignment -> VARIABLE = NUMBER

Rule 26 assignment -> VARIABLE = [ sentence ]

Rule 27 modification -> VARIABLE [ factor ] = VARIABLE [ factor ]

Rule 28 modification -> VARIABLE [ factor ] = VARIABLE

Rule 29 operation -> VARIABLE = expression

Rule 30 operation -> VARIABLE + +

Rule 31 expression -> expression + term

Rule 32 expression -> expression - term

Rule 33 expression -> term

Rule 34 expression -> VARIABLE [ factor ]

Rule 35 expression -> LEN ( term )

Rule 36 term -> term \* factor

Rule 37 term -> term / factor

Rule 38 term -> term / / factor

Rule 39 term -> factor

Rule 40 factor -> VARIABLE

Rule 41 factor -> ( expression )

Rule 42 factor -> NUMBER

Rule 43 print -> PRINT ( sentence )

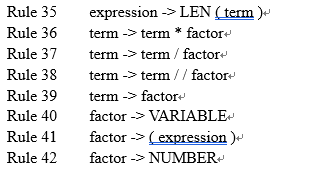
Rule 44 sentence -> word , sentence

Rule 45 sentence -> word

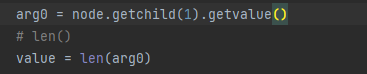
Rule 46 word -> NUMBER

Rule 47 word -> VARIABLE

1. Translation的关键部分逻辑设计
2. len()函数的求值和赋值



求长度的部分规则在这，可以发现term最终可以归约到某一个变量或者数字上去。

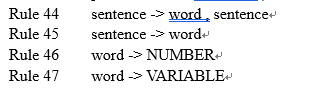


对孩子结点取值计算长度就行了

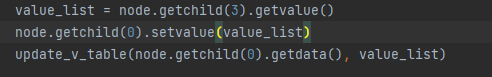
1. 对变量进行数组类型赋值

规则如下





对应到translation的代码：



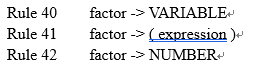
先把list从子结点的值中取出来，然后再赋给变量，更新v\_table

1. 给某一变量赋予数组的某一元素值

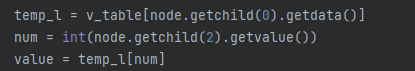
规则：







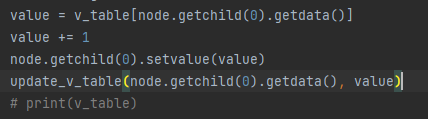
代码：



获取list的变量名，获取下标，得到值，然后赋给变量

1. 自加符号



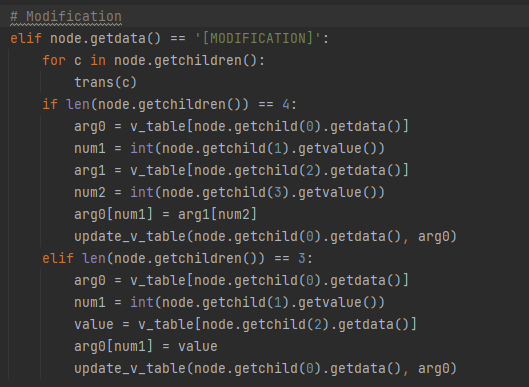


1. 对数组某一元素值进行修改

规则：



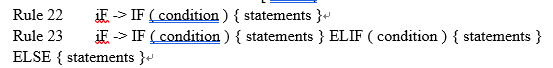
代码：



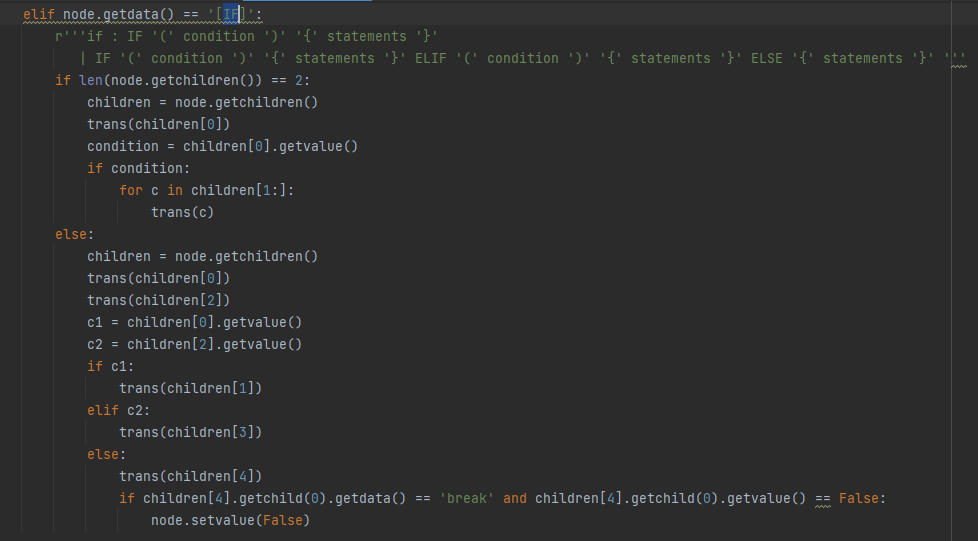
主要依赖的还是子结点的传值和从v\_table获取变量值

1. If 语句的解析

规则：



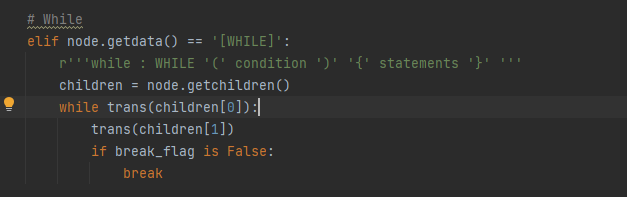
代码：



通过对子结点的trans调用，判断是否满足条件，决定执行哪一个分支，并视break\_flag的情况决定是否要break

1. While语句的解析

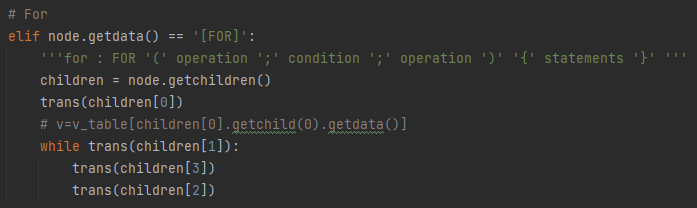




子结点不断递归调用，condition判断是否满足，看break\_flag决定是否终止。

1. For语句的解析





按照正常for循环时的执行顺序写出对应代码即可，不多作赘述。

1. Break语句的解析

设置一个break\_flag，当出现break时，就将该值由True改为False，然后让循环语句去判断是否要break即可。

