不要觉得自然语言很复杂，不要觉得它比计算机编程语言更复杂，你想想，你在接触计算机编程语言时，是不是觉得很难，可当时你已经掌握了自然语言了呀。

自然语言和计算机编程语言各有特点。以下是我总结出来的不同点：  
（1）自然语言是用于人的交流的，为了产生和识别都不复杂，所以语法结构中，虽然允许嵌套，但不允许递归，且嵌套层级不会太多。计算机语言的语法结构，允许无穷多层的嵌套和递归，比如if-else可以无穷进行下去。

句子：主语 谓语 宾语 | ……

宾语：定语 宾语 | …… （不会出现： 宾语： 主语 谓语 宾语。每次嵌套，都是不同的语法结构，区别度大，容易识别）

用自然语言去描述一个if-else，我们会把它拆开，而不是嵌套在一起，因为自然语言的语法不允许这种嵌套。如果if-else分支太多，用自然语言表达出来会很难理解，我们常常会说“不好表达，要不我画张图给你吧”，这张图里可以描述出这些嵌套的层级关系。从这个角度看，计算机语言具有表达复杂逻辑的功能，但这样的代码依然不好读，对于维护而言，还是需要图描述的，图语言，图形编程，是一个发展方向，python利用换行符、缩紧来作为语法的一部分，某种意义上，实现了图形编程的思想。

（2）自然语言允许歧义，我们在日常交流中还会利用歧义，达到某些目的。而编程语言中，绝对不允许歧义。

（3）自然语言用于描述事物和事物间的关系，编程语言用于描述和操作数据。可以用数据来表示事物。

（4）自然语言中，有因为-所以，这个语法，用来表达事物的因果关系。而编程语言没有，编程语言的语法都是操作类的。

自然语言和计算机编程语言的相同点：

（1）它们的语句，按语法进行分析，都可以形成树状结构，这也给计算机处理这些语言提供了思路。输入一串文本，输出一个树，由非终结符到终结符。试想一下，是否有语法结构为图的语言？

（2）

对一个语言的有穷描述，有两种方式：

（1）文本（语法规则文件，EBNF）

（2）自动机

这两种描述是等价的。文本描述用于语句的生成，自动机用于语句的识别。

写语法规则文件时要注意的点：

（1）词法规则中，优先级问题，比如关键词的规则一定要在标志符的前面

（2）写语法规则时，要区分BNF和EBNF，有些工具可能不支持EBNF

（3）表达式的语法规则要注意：左递归、优先级和结合性。对于Antlr，可使用直接左递归（间接左递归不行），可以把所有的运算都用一个语法规则来涵盖，然后用最简洁的方式支持表达式的优先级和结合性，优先级可以靠前后顺序解决，结合性可以靠标志<assoc=right>解决。

expression

: primary

| expression bop='.'

( IDENTIFIER

| functionCall

| THIS

)

| expression '[' expression ']'

| functionCall

| expression postfix=('++' | '--')

| prefix=('+'|'-'|'++'|'--') expression

| prefix=('~'|'!') expression

| expression bop=('\*'|'/'|'%') expression

| expression bop=('+'|'-') expression

| expression ('<' '<' | '>' '>' '>' | '>' '>') expression

| expression bop=('<=' | '>=' | '>' | '<') expression

| expression bop=INSTANCEOF typeType

| expression bop=('==' | '!=') expression

| expression bop='&' expression

| expression bop='^' expression

| expression bop='|' expression

| expression bop='&&' expression

| expression bop='||' expression

| expression bop='?' expression ':' expression

| <assoc=right> expression

bop=('=' | '+=' | '-=' | '\*=' | '/=' | '&=' | '|=' | '^=' | '>>=' | '>>>=' | '<<=' | '%=')

expression

;