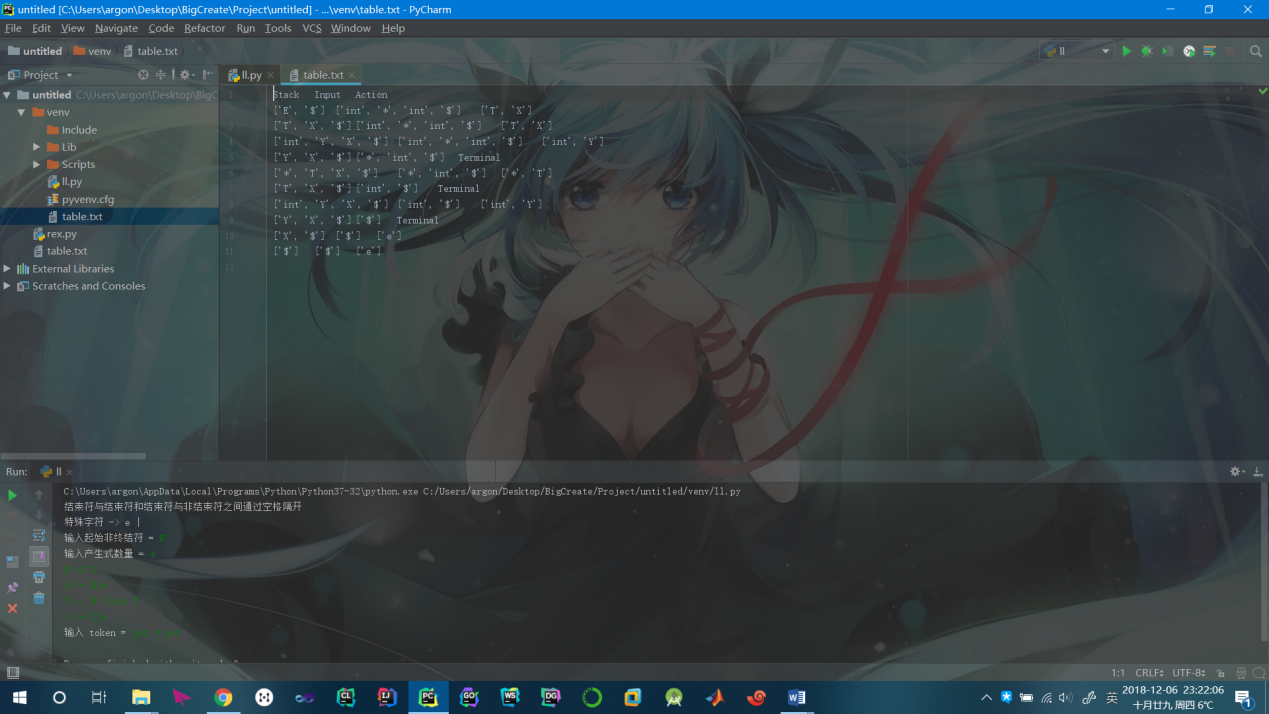
1. 使用说明

使用Python 3.x 运行代码 ll.py，依次输入起始非终结符、产生式数量、产生式和token串，token串分析表格在同一目录下的table.txt文件中，table.txt文件中(‘e’表示空)。



示例输入

结束符与结束符和结束符与非结束符之间通过空格隔开

特殊字符 -> e |

输入起始非终结符 = E

输入产生式数量 = 4

E->T X

X->+ E|e

T->( E )|int Y

Y->\* T|e

输入 token = int \* int

示例输出

Stack Input Action

['E', '$'] ['int', '\*', 'int', '$'] ['T', 'X']

['T', 'X', '$'] ['int', '\*', 'int', '$'] ['T', 'X']

['int', 'Y', 'X', '$'] ['int', '\*', 'int', '$'] ['int', 'Y']

['Y', 'X', '$'] ['\*', 'int', '$'] Terminal

['\*', 'T', 'X', '$'] ['\*', 'int', '$'] ['\*', 'T']

['T', 'X', '$'] ['int', '$'] Terminal

['int', 'Y', 'X', '$'] ['int', '$'] ['int', 'Y']

['Y', 'X', '$'] ['$'] Terminal

['X', '$'] ['$'] ['e']

['$'] ['$'] ['e']

1. 生成流程
2. 切割输入字符串为终结符和非终结符
3. 对每一个非终结符，对每一个以该非终结符为产生式左部的产生式的有部第一个字符（设该字符为target），如果该字符是终结符，将该字符添加进该非终结符的 first集，如果该字符是非终结符，更新target，递归地重复本过程，如果target的 first 集包含空，右部顺延一个字符更新target，迭代地重复本过程。
4. 对于每一个产生式的右部的每一个非终结符，如果该非终结符后无字符或者该非终结符后的字符是 first 集包含空的非终结符，将产生式左部的 follow 集添加进该非终结符的 follow 集；如果该非终结符后面的字符是终结符，添加该终结符进入非终结符的 follow 集；如果该非终结符后面的字符是非终结符，将后者的 first 集（不含空）添加进前者的 follow集，每轮对产生式的迭代在最后计算所有 follow 集中元素数量，当元素数量不变时，迭代终止。
5. 构建二维字典，第一维为非终结符，第二维为终结符，二维字典中的每个值（字典[非终结符][终结符]）为产生式的右部，非终结符指某个产生式的左部，终结符指产生式右部第一个字符（如果是非终结符则是其 first 集）。同时，对于产生式左部非终结符的 first 集包含空的非终结符，在以其 follow 集为终结符的位置写入空。
6. 使用二维字典对 token 串进行分析。使用起始非终结符和’$’初始化栈，对每一个 token ，当栈顶是非终结符时在字典中找到相应的值，更新栈，当栈顶时终结符时，stack和input同时去掉开头的元素。重复本过程，直到 stack 中只剩下’$’，迭代终止，分析完成。