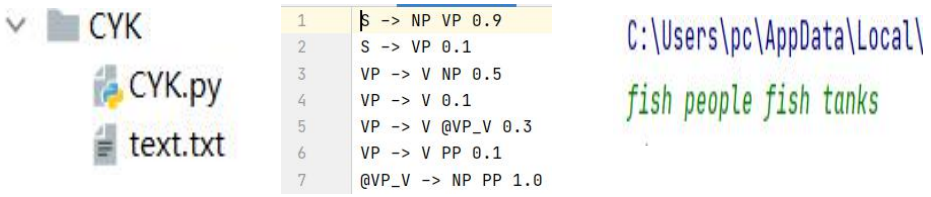
说明：

本次程序根据 CYK 算法实现了句子“fish people fish tanks”最可能的统计句法树，对应语法如下：（如需辨认其他句子，可以运行后的控制界面输入需要生成句法分析树的句子，并且在 text.txt 中输入对应语法）

S → NP VP	0.9		
S → VP	0.1	N → *people*	0.5
VP → V NP	0.5	N → *fish*	0.2
VP → V	0.1	N → *tanks*	0.2
VP → V @VP_V	0.3	N → *rods*	0.1
VP → V PP	0.1	V → *people*	0.1
@VP_V → NP PP	1.0	V → *fish*	0.6
NP → NP NP	0.1	V → *tanks*	0.3
NP → NP PP	0.2	P → *with*	1.0
NP → N	0.7		
PP → P NP	1.0		

本程序需要在运行后的控制界面输入需要生成统计句法树的句子，并且在程序同一目录下创建一个 text.txt 文件，并且将语法规则写入该文件中（每个语法一行，每个字符都要用空格隔开）



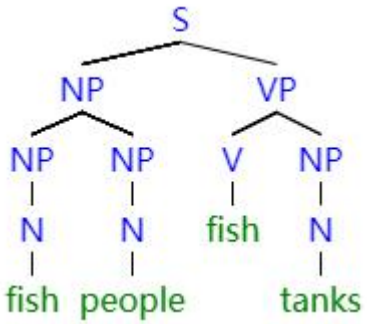
程序参考了网上的一些代码，将读入的语法和句子先经过 *create_target(str)* 函数将其格式化为字典，并且让每个主键对应一个字典含有其概率，节点值等信息，然后通过递归遍历完所有非终结字符的可能概率，得到最佳结果，最后 back 函数输出每一层树的节点的对应概率树，并且返回统计句法树串。

运行结果：

输入： *fish people fish tanks*

输出： **[S[NP[NP[N[fish]]][NP[N[people]]]][VP[V[fish]][NP[N[tanks]]]]]**

运行结果：



CYK

```
C:\Users\pc\AppData\Local\Programs\Python\Python37\python.exe [
请输入要构建统计语法书的句子：
fish people fish tanks
0  ('S', 0.00018521999999999994)
1    ('NP', 0.004899999999999999)
2        ('NP', 0.13999999999999999)
3            ('N', 0.2)
4                fish
2        ('NP', 0.35)
3            ('N', 0.5)
4                people
1    ('VP', 0.041999999999999996)
2        ('V', 0.6)
3            fish
2        ('NP', 0.13999999999999999)
3            ('N', 0.2)
4                tanks
[S[NP[NP[N[fish]]][NP[N[people]]]][VP[V[fish]][NP[N[tanks]]]]]

Process finished with exit code 0
```