## lua table parser说明文档：

PyLuaTblParser类成员函数介绍：

### def \_\_init\_\_(self): #构造函数

self.dd = {} #存储该类中的dict

self.flag = True #标记该类当前存的是list还是dict,True表示dict，False表示list

self.ll = [] #存储该类的list

### def remove\_comments(str): # 去注释

（本函数代码较多，取部分介绍）

对于去注释的思路是一个个字符查看，判断当前字符所处的域是什么，

分三种，引号域，注释域，代码域，当处于代码域时可能转换为引号域或者

注释域，判断字符是否出现‘或“或多级长括号，如果出现进入引号域，否则

还是代码域，如果出现 — 注释起始符，进行判断是否处于长括号，如果是长括号

则是多行注释，否则单行注释，之后进入注释域，注释域所有字符省略。所以需要

两个bool变量来表示当前所处的域，并且记录结束标识，比如引号域左边为单引号，

那么出现非转义单引号则结束引号域，如果是多级长括号，则当出现对应多级长括号

时结束引号域，进入代码域，注释处理也是一样；

### def check\_long\_branch(str, index): #检测长括号

检测从index索引开始是否出现长括号，如果出现是几级，

如果不出现返回-1，出现返回长括号标识的字符数，比如 [===[

返回5；

### def convert\_char(str,index): #转义处理

该函数处理引号域内的转义字符，比如\n,\t,\r等，返回转义之后的字符

‘\n’, ’\r’, ’\t’，因为我们参数接受是原样字符，需要自己做词法分析；

### def name\_check(self,str): #检测变量名是否合法

因为在Lua中 table有三种表示，name = exp, [exp] = exp, exp,当key是字符串时并且

没有用[‘ ‘]括起，该字符串需要满足变量名定义规则，就是字母或者下划线开始，

只能由字母下划线和数字组成；满足返回True，否则返回False

### def remove\_space(str): # 去除左右空格和回车

由于在代码域中可以用任意类型的空格回车和制表符，所以当切割完成一个变量后，

需要左右删除空白字符才能继续处理；函数返回取去除之后的字符串；

### def \_\_str\_to\_dict(self,str): # 将一个lua\_table转换为dict

将一个lua\_table转换为一个dict字典，输入参数是一个切割好的table元素字符串，

我们判断是否有处于非引号域和不在大括号内的=号，如果有转换为dict，否则索引

递增，当有=号时获得key和value后再个个分析获得dict中对应的key和value，

在分析key时判断有无[ ] 号，如果有去掉之后当作exp分析，否则就是一个name字符串；

### def \_\_str\_to\_list(self,str): # 将一个lua table转换为list

道理同上；

### def test\_input(self,sstr): #测试输入

由于在计算数值型变量真正的值时，我会做简单处理后调用eval内置函数，

所以为了防止eval代码注入，需要对输入的参数做检测，只能由数字，运算符

和E或者P字母，其他则非法；

### def str\_to\_num(self,sstr): # str转number

计算数值型变量字符串获得真值

### def \_\_convert\_what(self,str): # 判断lua table转换成什么

判断该str应该转换为list还是dict, -1为list, 正值为dict 返回值是 = 的索引

### def check\_value(self,str): #检查value类型

当得到一个lua\_table的value后检查table的value是否为字符串，如果是返回转义后字符串，否则返回 False

### def char\_recovery(self,str): #将python字符串转换为lua\_table时，需要防转义

当将一个dict或者list转为lua\_table时，对于字符串我统一使用双引号，所以对于本来就存在的双引号，需要加转义符，还有对于\转义符号，也需要防转义，一个原则，就是再次load

时得到结果不变

### def \_\_str\_ana(self,str): # 一个 table 的 value 元素

分析一个lua\_table的value元素类型，可能是一个嵌套table，也可能是一个字符串或者

布尔型或者nil或者数值型，该函数进行分析后直接返回应该存在python中的值

### def \_\_list\_ana(self,ll): #将list解析成一个lua table字符串

该函数将一个python中的list转换为lua中的table，由于是list，所以直接换成exp的形式

对于数值型或者nil或者bool型，直接返回对应的字符串，如果是字符串，需要加双引号，并且防转义，对于list中包含嵌套，直接会自动递归调用

### def \_\_dict\_ana(self,di): #将dict解析成一个lua table字符串

该函数与\_\_list\_ana(self,ll)相对应，相互调用，只不过该函数处理dict，需要返回[exp] = exp

的形式；

### def \_\_mycopy(self, dd): # 实现自定义深拷贝

该函数在loadDict或dumpDict时被调用，由于只处理number和str两种key，

所以需要自己实现深拷贝，并且防止引用造成的级联修改；

### def load(self,str): #将lua\_table转为python中的dict或list

该函数直接调用\_\_str\_ana(self,str)成员函数；

### def dump(self): #将python的dict或list转为lua\_table

根据该类中存的类型，调用\_\_dict\_ana(self,dd)或者\_\_list\_ana(self,ll)函数

### def loadDict(self,d): #加载一个dict到类

将一个dict加载到该PyLuaTbParser类中，然后dump或者dumpDict；

### def dumpDict(self): #将该类中存的dict深拷贝出来返回

返回该类中dict，并且不是引用返回

### def loadLuaTable(self, f): #从一个文件加载lua\_table

与load函数类似，只不过要转的字符串来源于文件，多一个文件读取操作

### def dumpLuaTable(self,f): #将类的对应的lua\_table写到文件

与dump函数类似，只是将返回值写入文件，多一个文件写操作

### def \_\_getitem\_\_(self, item): #重载[] 索引运算符,没有赋值,可以读

重载索引运算符，使得该类可以像dict或者list一样用a[index]形式获得值

### def \_\_setitem\_\_(self, key, value): #重载[] 索引运算符,有赋值,可以写

重载赋值运算符，使得该类可以被赋值通过索引，比如a[index] = newvalue

可以像dict一样直接赋新值；

### def update(self,d): #用字典d更新类中数据

直接调用深拷贝\_\_mycopy(self,dl)直接覆盖类中数据；