一、字符串表达式转化分离成数字与运算符的集合

利用python的正则表达式进行识别，找出输入字符串里面运算字符的序号，存储在一个list里面，再根据这个list的运算符序号将运算符与运算数字进行分离，分离后的结果存放在新的一个list里面，供后面的运算提前做准备。

二、中缀表达式转后缀表达式

在这个步骤中，利用了栈这种数据结构，实现了中缀表达式到后缀表达式的转化。首先创建一个运算符栈，将前面已经分开的表达式元素进行处理，从第一个开始，遇到数字直接输出。当遇到运算符，若优先级比栈顶操作符的优先级低，则直接输出，若比栈顶元素优先级高，则压入运算符栈里面。其中有一个特殊的地方，又括号具有最低的优先级，而左括号具有最高的优先级。经过一系列的循环，可以在输出的list里面得到一个后缀的表达式，再将其中的括号运算符去掉。

三、计算后缀表达式

得到的后缀表达式类似于 “2, 3 ,+, 3 ,\*, 6, -” 这样的表达式，这个表达式其实是中缀表达式 “(2+3)\*3-6” 转化得到的，可以看到后缀表达式的运算符是在后面的，不便于我们的理解，但是有一个优点就是便于计算机的计算，可以利用一个计算栈很简单地计算，而不用考虑里面的优先级。这这里面，首先创建一个运算栈，不断将后缀表达式的元素入栈，遇到数字直接入栈，遇到操作符则弹出栈顶两个元素，将这个运算符作用与这两个数字，然后再将结果入栈，这样一直循环下去，直到最后运算栈里面只剩下一个数字，这个数字也就是该表达式的运算结果。

四、表达式树的构造

有了后缀表达式，构建表达式树已经是一件简单的事情。实际上，表达式树的三种遍历分别对应着表达式的三种不同的表示方法。前缀表达式是表达式树的先序遍历，中缀表达式是表达式树的中序遍历，后缀表达式是表达式树的后序遍历。可以利用一个数据栈以及一个后缀表达式很方便地构造出一棵表达式树。利用python建立一个结点类Node，其中有三个元素，一个数值以及两个指针，分别指向左子树以及右子树。

遍历后缀表达式的各个元素，遇到数字直接入栈，遇到运算符则从栈中弹出两个元素，以这个运算符为根节点，两个数字为左右子树建立一颗树，然后再将这棵树入栈，把这棵树当成一个元素处理，这样当遍历到后缀表达式的最后一个运算符，在栈中就剩下整个表达式的表达式树。

五、方程的运算

在求解方程的过程中，用到了python里面的sympy符号运算的函数库，接受一个方程的字符串，然后调用solve()函数进行求解。

六、微分的计算

计算微分的时候同样使用到了sympy这个函数库，调用diff()函数进行求解，在求解微分之前，先对字符串进行预处理，使得输入符合这个函数的要求

七、表达式树的图形化实现

在表达式的图形化实现中，利用了networkx这个库，把树当成是一种特殊的有向图来进行处理，构造出一个图，再用python的绘图函数库matplotlib对这个特殊的图进行绘制，同时根据树的深度，将里面的分支距离处理好，最终生成一个可视化的树结构。