概述

代码搜索引擎采用的检索方式有三种方式，分别是原始字符串匹配、词法分析Token串匹配、语法分析树中的结构匹配。

下面我们先依次介绍三种解析方式，然后再说明我们如何将三种解析方式的搜索结果进行结合。

方法一：原始字符串匹配

直接将源代码进行字符串精确匹配，只进行简单的预处理

例如：

Source code：

```
# 这是一个注释
for i in range(10):
    print('abcde')
```

Char string:

```
for i in range(10): print('abcde')
```

方法二：词法分析，Token串匹配

词法分析：将一段字符序列（如一段程序代码），转化成为一个token序列。

下面是一个源代码经过词法分析转换成Token串的样例：

Source code:

```
# 这是一个注释
for i in range(10):
    print('abcde')
```

Token string:

```
Comment Keyword_for Identifier Keyword_in Keyword_range
LeftParen Number RightParen Colon Keyword_print LeftParen
String RightParen
```

进行词法分析的程序或者函数叫作词法分析器（lexical analyzer，简称lexer）。

我们使用Python内置的词法分析器: https://docs.python.org/3/library/tokenize.html.

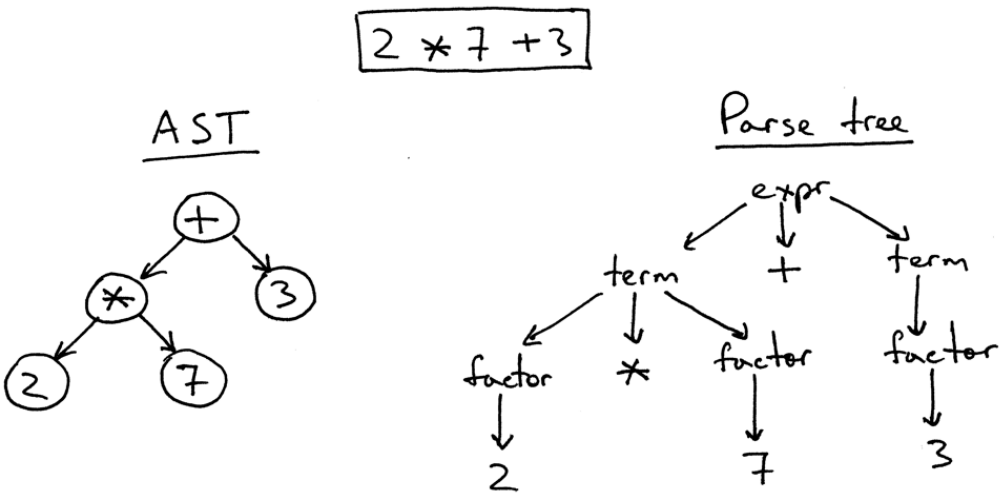
方法三：语法分析，语法树结构匹配

语法分析器基于特定语言的语法，将Token序列（由词法分析器生成）转换成一个抽象语法树。

抽象语法树（Abstract Syntax Tree，AST），或简称语法树（Syntax tree），是源代码语法结构的一种抽象表示。它以树状的形式表现编程语言的语法结构，树上的每个节点都表示源代码中的一种结构。之所以说语法是“抽象”的，是因为这里的语法并不会表示出真实语法中出现的每个细节。比如，嵌套括号被隐含在树的结构中，并没有以节点的形式呈现。

具体概念请参考维基百科抽象语法树，以及其他相关资料。

以下是表达式 2 × 7 + 3, 转换成语法树的例子：



在进行语法分析时，我们采用的工具如下：
1. Python ast : Python自带的语法分析器模块，通过parse()函数生成抽 象语法树，并提供对抽象语法树的遍历。链接如下：https://docs.python.org/3.7/library/ast.html.
2. C++ Clang : 提供了C、C++等语言的语法分析器。链接如下：https://clang.llvm.org/.

在进行语法分析得到语法树之后，我们采用论文 Similarity Evaluation on Tree-structured Data 中提出的方法，先将语法树转换成二叉树，这里我们对二叉树的所有叶结点进行结点填充，使每个叶节点自身也能够构成一个两层子树，然后取二叉树中的每个两层子树，当做一个索引词。

例如：
Source code:

```
a = 2 + 3 * 4
```

AST Token:

```
Module_Assign_0 Assign_Name_0 Name_Store_BinOp Store_0_0
BinOp_Num_0 Num_0_Add Add_0_BinOp BinOp_Num_0 Num_0_Mult
Mult_0_Num Num_0_0
```

其中，每个两层子树有三个结点，我们将三个结点用“_”连接起来，当做索引中的一个词。

最终方法：三种搜索方式结合：

我们最终采用的搜索方式，是将上述三种检索方式结合起来。

我们将原始字符、词法分析串、语法分析串三种解析结果合在一起，通过ElasticSearch，建立索引。
即数据库中的所有代码，都存在以上三种解析方式的索引。

实际使用过程中，传来一个查询query，我们会对该查询执行三种解析，得到三种不同的查询串，再用每一种查询串进行ElasticSearch的查询，然后将三种查询串搜索的结果进行合并，采取的合并策略是：原始字符串精确匹配的结果，优先级最高，排在最前面；词法分析的Token串查询的结果，优先级次之，依次排在后面；语法树的匹配结果，排在字符串结果和词法分析结果的后面。与此同时，在合并的过程中，进行去重，即字符串精确匹配的结

果，可能也在词法分析串搜索中也会出现，则去除词法分析中的这些返回结果。

注意：我们采用的搜索结合方式，针对查询较短代码时，返回的结果中通常是很多精确匹配的结果，以及词法分析串匹配的结果；而针对查询的代码较长时，精确匹配的结果往往很少，此时返回的应该大部分是语法结构的匹配结果。

建议：在前端查询界面，给予用户一定的使用提示，提示用户查询时，尽量输入**符合基本语法**的代码片段（例如，简单的括号要配对起来），这样才能返回较好的搜索结果。