Année universitaire 2016-2017
Université de Caen Basse-Normandie

Rapport numéro 2 : Comment utiliser Lex et Yacc pour notre IDE ?

Alexis Carreau
Thomas Lécluse
Emma Mauger
Théo Sarrazin
L2 Informatique

Table des matières

1 Lex

1.1 Théorie

Lex peut :
— convertir les chaînes de caractères en Tokens
— reconnaître les rôles des éléments du code grâce à des expressions régulières qui correspondent à des paternes
— renvoyer où il a trouvé tel élément et à quoi il correspond.

1.2 Pratique

Pour faire fonctionner Lex, nous devons lui spécifier une liste de token (qui doit s'appeler "tokens"), qui sont une représentation numérique du code. Ainsi que des expréssions régulières qui correspondent à chacun des token(s), permettant à Lex de les identifier.
On lui passe donc une liste de token qu'il doit retrouver, et il va retourner tout ce qu'il va trouver avec les expressions régulières.

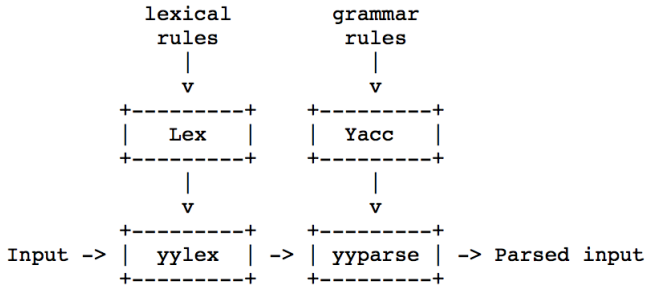
2 Yacc

2.1 Théorie

— Yacc récupère la liste de token renvoyée par Lex
— On lui passe une grammaire : façon d'assembler les token(s)
— Analyse les token(s) et créé un arbre syntaxique qui impose une structure hiérarchique des token(s).
— Dès qu'il trouve un élément qui correspond à la grammaire qu'on lui a passé, il (dans notre cas) exécute une action.

2.2 Pratique

On commence donc par définir une grammaire, contenue dans une docstring d'une fonction (ici python), qui correspond à la syntaxe du langage que nous souhaitons traiter. Yacc va parcourir la liste de token renvoyée par Lex et dès qu'il rencontrera une syntaxe correspondante, il exécutera la fonction dans laquelle est contenue la docstring.



3 Et dans notre projet ?

Lex et Yacc nous permettrons de vérifier la syntaxe du code ainsi que de colorer les mots clefs.