Gerador de Analisador Sintático

Ferramenta para gerar o código de analisadores sintáticos automaticamente.

Recebe as regras sintáticas como entrada.

Usados em conjunto com geradores de analisadores léxicos.

Gerador de Analisador Sintático - Exemplos

Yacc	– parser em C (usado com o Lex)
Bison	– parser em C (usado com o Flex)
JavaCC	– parser em Java
ANTLR	– parser em diversas linguagens

Parser em Python:

PLY	Cotovia	Lrparsing
PLYPlus	Pyleri	

PLY – Python Lex-Yacc

Ferramenta que ajuda a criar programas em Python para analisar e interpretar um arquivo de entrada.

Possui 2 módulos:

- lex.py : analisador léxico
- yacc.py : analisador sintático

PLY – yacc.py

módulo que ajuda a criar programas em Python para definir um parser para tokens gerado pelo um analisador léxico.

Valida uma sequencia de tokens usando **regras sintáticas.**

Inicia o processo através da função **parse()** .

lexer = lex.lex()

PLY – Formato do Arquivo .py

```
import ply.yacc as yacc

% import dos tokens
from mylex.py import tokens

% regras sintáticas

parser = yacc.yacc()
```

PLY – Formato do Arquivo .py

Regras sintáticas

- Definir uma função para cada **não-terminal** gramática;

- A função será iniciada por t_ seguida do nome do não-terminal e irá receber como parâmetro o token sendo avaliado.

- O corpo da função define as regras sintáticas.

PLY – Formato do Arquivo .py - Exemplo

Regras sintáticas

considerando a regras F -> (E) | a *:*

Token (definido no Léxico)

```
t_AP = r‘\(‘
t_FP = r’\)’
t_A = r’a’
```

PLY – Formato do Arquivo .py - Exemplo

Regras sintáticas

considerando a regras F -> (E) | a *:*

Definido no sintático

```
def t_f(t):
    ‘’’ f : AP e FP
            | A
            ’’’
```

PLY - Exemplo

GRAMÁTICA

NT = { EXP, OP, ID , NUM }

T = { +, -, *, / }

SI = EXP

Regras = {

EXP -> ID OP EXP

NUM OP EXP

ID

NUM

OP -> + | - | * | /

}

PLY – Exemplo de Arquivo mylex.py

```
import ply.lex as lex
tokens = ( NUM, ID, SOMA, SUB, MULT, DIV )
t_SOMA = r'\+'
t_SUB = r'-'
t_MULT = r'\*'
t_DIV = r'/'
def t_NUM(t):
        r'{[0-9]}+"."{DIGIT}* '
def t_ID(t):
        r'[a-z][a-z0-9]*'
lexer = lex.lex()
```

PLY – Exemplo de Arquivo myparser.py

```
import ply.yacc as yacc
From mylex import tokens


def t_exp(t)
        ‘’’exp : ID op exp
                | NUM op exp
                | ID
                | NUM
                ‘’’
```

```
def t_op(t)
        ‘’’op : SOMA
                | SUB
                | MULT
                | DIV
                ‘’’


parser = yacc.yacc(start=‘exp’)
```