

Oferece um grau de licença poética sem precedentes na composição e estrutura de programas



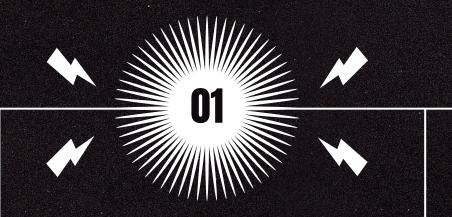
# ROCKSTAR?

Rockstar é uma linguagem de programação projetada para criar programas que também são hair metal power ballads.

### **ROCKSTAR FEATURES**

**GRAU POÉTICO case-insensitive** Com exceção de variáveis (Tommy = 1337)Tommy was a big bad brother. de nomes próprios **ECMAScript type system Comments** Exceto undefined, usa-se Isso é rock; cabe ao público "mysterious" em vez disso encontrar o significado.

# IMPLEMENTAÇÃO, Python Lex-Yacc



#### LEX\_py

Usado para quebrar o código em tokens especificados por regras de expressão regular



#### YACC\_py

Reconhece a sintaxe especificada na forma de uma gramática livre de contexto

# Yacc, "Yet Another Compiler Compiler"

Cada regra gramatical é definida por uma função Python em que a docstring para essa função contém a especificação gramatical livre de contexto apropriada. As declarações que compõem o corpo da função implementam as ações semânticas da regra. Cada função aceita um único argumento p que é uma sequência contendo os valores de cada símbolo gramatical na regra correspondente.

```
def p_expression_plus(p):
    'expression : expression PLUS term'
    # ^ ^ ^ ^ ^ ^ ^
# p[0]    p[1]    p[2] p[3]

p[0] = p[1] + p[3]
```

## **PARSING**

Por exemplo, se você quiser analisar expressões aritméticas simples, você pode primeiro escrever uma especificação gramatical livre de contexto.

: NUMBER

expression · )

factor

#### **PARSING**

Sempre que uma regra gramatical específica é reconhecida, a ação descreve o que fazer.

```
Action
Grammar
expression0 : expression1 + term
                                   expression0.val = expression1.val + term.val
                                    expression0.val = expression1.val - term.val
             expression1 - term
                                    expression0.val = term.val
             term
                                    term0.val = term1.val * factor.val
term0
            : term1 * factor
             term1 / factor
                                    term0.val = term1.val / factor.val
                                    term0.val = factor.val
             factor
                                    factor.val = int(NUMBER.lexval)
factor
            : NUMBER
              ( expression )
                                    factor.val = expression.val
```

# BRGADÔL



#### PERGUNTAS?

github.com/Durfan/ufsj-rockstar

CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon**, and infographics & images by **Freepik** 

Please keep this slide for attribution