الاسم: ابراهيم شكري السعيد الجزار

Logo

Description automatically generatedCompilers…

Creating Parse Table.

**Language Grammar to Create Parse Table?**

    program -> statement\_list

    statement\_list -> statement ';' | statement\_list ';'

    statement -> assignment | print | if\_statement | while\_loop

    assignment -> identifier '=' expression

    print -> 'print' expression ';'

    if\_statement -> 'if' expression '{' statement\_list '}'

    while\_loop -> 'while' expression '{' statement\_list '}'

    expression -> term | number op number

    term -> number | identifier | string | '(' expression ')'

    number -> digit

    identifier -> '[a-zA-Z\_][a-zA-Z0-9\_]\*'

    op -> '+' | '-' | '\*' | '/'

    string -> '[a-zA-Z\_][a-zA-Z0-9\_]\*'

    digit -> '[0-9]+'

**Terminals and Non-Terminals:**

Terminals: {';', '=', 'print', 'if', '{', '}', 'while', '+', '-', '', '/', '(', ')', '[a-zA-Z\_][a-zA-Z0-9\_]', '[0-9]+'}

Non-terminals: {program, statement\_list, statement, assignment, print, if\_statement, while\_loop, expression, term}

**Constructing Parse Table using First and Follow Set:**

First Set:

First(program) = First(statement\_list) = First(statement) = {identifier, print, if, while}

First(assignment) = {identifier}

First(print) = {'print'}

First(if\_statement) = {'if'}

First(while\_loop) = {'while'}

First(expression) = {number, identifier, string, '('}

First(term) = {number, identifier, string, '('}

First(number) = {digit}

First(identifier) = {'[a-zA-Z\_]'}

First(op) = {'+', '-', '\*', '/'}

First(string) = {'[a-zA-Z\_]'}

First(digit) = {'[0-9]'}

Follow Set:

Follow(program) = $

Follow(statement\_list) = Follow(program) = $

Follow(statement) = {';', '}', Follow(statement\_list)}

Follow(assignment) = {';', '}', Follow(statement\_list)}

Follow(print) = {';', '}', Follow(statement\_list)}

Follow(if\_statement) = {Follow(statement\_list), '}'}

Follow(while\_loop) = {Follow(statement\_list), '}'}

Follow(expression) = {')', Follow(term)}

Follow(term) = {op, ')', Follow(expression)}

Follow(number) = {op, ')', Follow(term)}

Follow(identifier) = {op, '=', ')', Follow(term)}

Follow(op) = {number, identifier, string, '('}

Follow(string) = {op, ')', Follow(term)}

Follow(digit) = {op, ')', Follow(term)}

**Code for Constructing Parse Table:**

parse\_table = {

    'program': {

        'identifier': 'statement\_list',

        'print': 'statement\_list',

        'if': 'statement\_list',

        'while': 'statement\_list',

        '$': '$'

    },

    'statement\_list': {

        'identifier': 'statement ; statement\_list',

        'print': 'statement ; statement\_list',

        'if': 'statement ; statement\_list',

        'while': 'statement ; statement\_list',

        ';': 'statement\_list',

        '{': 'statement\_list { statement\_list } statement\_list'

    },

    'statement': {

        'identifier': 'assignment',

        'print': 'print expression ;',

        'if': 'if expression { statement\_list }',

        'while': 'while expression { statement\_list }'

    },

    'assignment': {

        'identifier': 'identifier = expression ;'

    },

    'print': {

        'print': 'print expression ;'

    },

    'if\_statement': {

        'if': 'if expression { statement\_list }'

    },

    'while\_loop': {

        'while': 'while expression { statement\_list }'

    },

    'expression': {

        'identifier': 'term',

        '(': 'term',

        'number': 'term',

    },

    'term': {

        'identifier': 'identifier',

        'string': 'string',

        '(': '( expression )',

        'number': 'number'

    },

    'number': {

        'digit': 'digit'

    },

    'identifier': {

        'identifier': 'identifier ( expression ) op term',

        '(': '( expression ) op term'

    },

    'op': {

        '+': '+ term',

        '-': '- term',

        '\*': '\* term',

        '/': '/ term'

    },

    'string': {

        'identifier': 'identifier',

        'string': 'string'

    },

    'digit': {'digit' :'digit' }

}

df = pd.DataFrame.from\_dict(parse\_table, orient='index')

df.index.name = 'Non-terminal'

df.columns.name = 'Terminal'

df.fillna('', inplace=True)

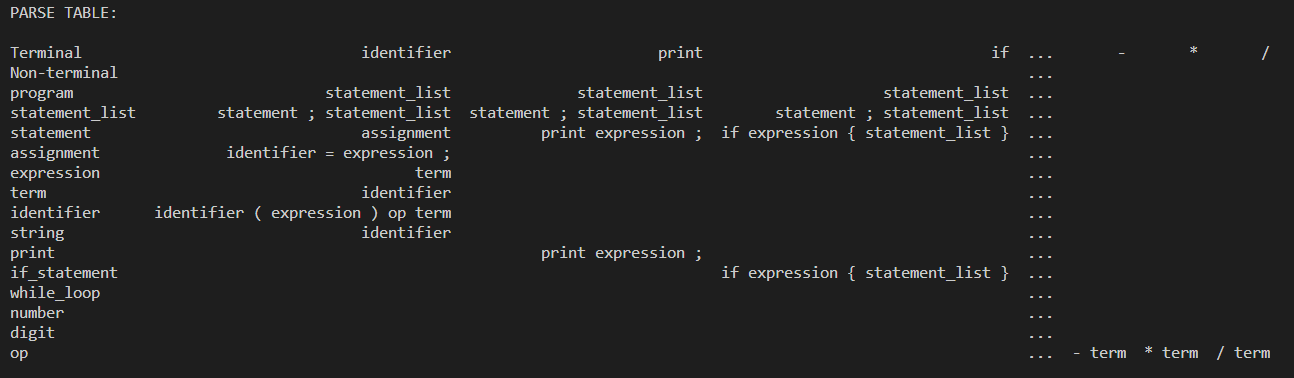
df = df.rename(columns=lambda x: 'digit' if x.isdigit() else x)

print()

print("PARSE TABLE:")

print()

print(df)

**Code Output:**