SER502-Porygon-Team11

Anmol Mallikarjun Nemagouda
Chandrakanth Dhanunjai Chintala
Kaumudi Degekar Gulbarga
Rakshilkumar Modi

Porygon-Team11

—

SER 502

—

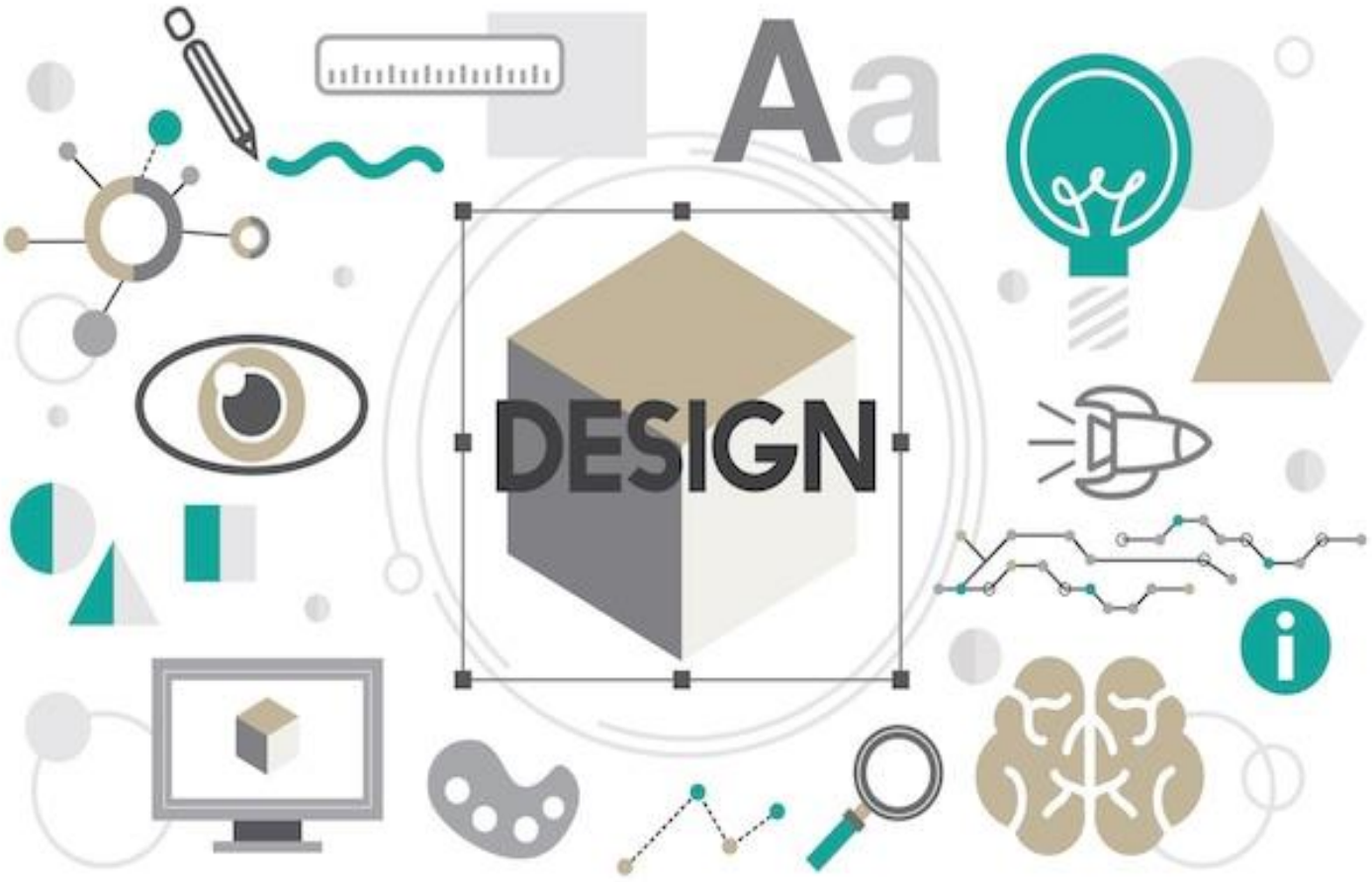
Dr. Ajay Bansal

Overview

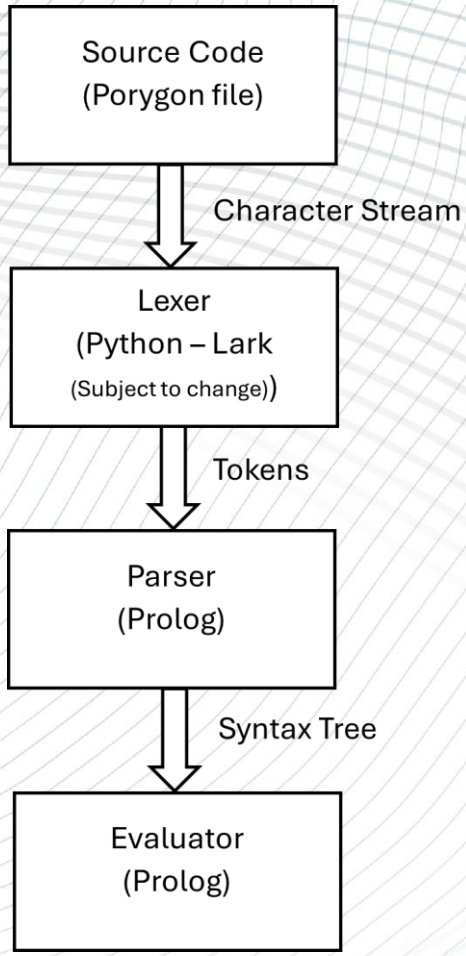
- About the Language.
- Design.
- Design Components.
- Features.
- Grammar

About the Language

- **Name:** Porygon
- **Extension:** .prgn
- **Paradigm:** Imperative
- **Programming languages used:** Python and Prolog.


Aa
DESIGN

Future Project Pipeline



Design Components

- **Script:** A script that helps to run the Porygon program file.
- **Source Code:** This is the source code for the newly developed language that needs to be executed.
- **Lexer:** We will be using Python (Lark- *subject to change*) as a Lexer for this project. It accepts the source code as an input and breaks down the given code into tokens. The tokens will be generated as a list which will then be passed on to Prolog for parsing.
- **Parser:** We be using Prolog as a parser for this project. It will take the token list from the lexer and generate a parse tree as per the defined grammar.
- **Evaluator:** This is used to evaluate the parse tree generated by the parser and execute the instructions.

Feature List

Declarations:

- Constant declaration.
- Plain declaration.
- Variable declaration.

Commands:

- If
- If else
- If else ladder
- For value in range (for loop variation).
- While loop.
- Assignment.
- Print statement.
- For loop.

Assignment:

- Initial assignment.
- Declarative assignment.
- Shorthand Assignment.

Operations:

- Arithmetic (addition, subtraction, multiplication, division, increment, decrement).
- Modulus.
- Square.
- Square root.
- Exponent.
- Cube.
- Cube root.
- String length.
- Boolean operations.
- Ternary operator.
- Parenthesis.

Feature List

General Program rules:

- Each program begins and ends with curly braces '{}'.
- Each declaration and command line must end with a semi colon ';'.
- Variable names can only start with a lower-case letter, it can be alphanumeric and contain underscore '_' but cannot end with an underscore. It can also contain upper-case letters.
- String Values must be enclosed within double-quotes (" ").

Data types:

- Integer (int).
- String.
- Boolean (bool).
- Floating point numbers (float).

Keywords (subject to change):

- Const.
- If.
- Else.
- For.
- In.
- Not.
- And.
- Or.
- Elseif.
- While.
- True.
- False.
- Print.

Grammar

```
%%%%%%%%%%%%%%%%%%%%% CODE BEGIN BLOCK %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%% Program starts with '{' and ends with '}' %%%%%%%%%%%%%%%%

Block ::= '{',DeclarationList,CommandList , '}'.

%%%%%%%%%%%%%%%%%%%%%%%%% DECLARATION %%%%%%%%%%%%%%%%%%%%%%%%%
%Declaration can be multiple if follows
%'const' variable type, varible name = value/expression;
%variable Type (String, int, float, bool), variable name;
%variable Type, variable name = value/expression;



DeclarationList ::= Declaration ,';', DeclarationList
    | Declaration,';'.
Declaration ::= Constant_assignment
    | Declarative_assignment
    | Plain_declaration.
```

```
Constant_assignment ::= 'const' 'int' Variable_name = Numbers
    | 'const' 'string' Variable_name = '\"', String_Value, '\"'
    | 'const' 'bool' Variable_name = Bool_value
    | 'const' 'float' Variable_name = Float_value
    | 'const' 'int' Variable_name '=' Expression
    | 'const' 'string' Variable_name '=' Expression
    | 'const' 'bool' Variable_name '=' Expression
    | 'const' 'float' Variable_name '=' Expression.
Declarative_assignment ::= 'int' Variable_name = Numbers
    | 'string' Variable_name = '\"', String_Value, '\"'
    | 'bool' Variable_name = Bool_value
    | 'float' Variable_name = Float_value
    | 'int' Variable_name '=' Expression
    | 'string' Variable_name '=' Expression
    | 'bool' Variable_name '=' Expression
    | 'float' Variable_name '=' Expression.
Plain_declaration ::= 'int' Variable_name
    | 'string' Variable_name
    | 'bool' Variable_name
    | 'float' Variable_name.
```

Grammar

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% COMMANDS BLOCK %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Commands can be multi line or single line, and it but multiple
% We are using Assignment, ternary operator, print command, string length, just if command, if else command, if else ladder command, while,for for in range.

% Out of which Assignment, ternary operator, print command, string length this are single line commands ending with semicolon
% and rest and block commands enclosed within {}.



CommandList ::= Plain_command,';',CommandList
    | Plain_command, ';'.
Plain_command ::=  Assignment
    | Ternary_operator
    | Print
    | String_length
    | If_command
    | If_else_command
    | If_else_ladder_command
    | While_command
    | For_command
    | For_in_range_command.
```

```
% Assignment can be done three ways, giving value to a variable directly and also using syntactic sugar, declaring a variable and giving it a value.

Assignment ::= Initial_assignment
    | Declarative_assignment
    | Shorthand_assignment.

Shorthand_assignment ::=  Variable_name,'+=',Expression
    | Variable_name,'-=',Expression
    | Variable_name,'*=',Expression
    | Variable_name,'/=',Expression
    | Variable_name,'%=',Expression
    | Variable_name,'^=',Expression.

Initial_assignment ::= Variable_name, '=', Expression.
```

Grammar

```
% Expression include arthimetic operations(addition,subtraction,mutliplication,divison, Extra(Modulus,Exponent,Sqaure,SqaureRoot,Cube,CubeRoot).
% Expression also includes Assignment of variables and values.

Expression ::= Term, '+', Expression
    | Term, '-', Expression
    | Term.
Term ::= Factor, '*', Term
    | Factor, '/', Term
    | Factor, '%', Term
    | Factor.
Factor ::= Exponent, '^', Factor
    | Exponent.
Exponent ::= Square
    | SqaureRoot
    | Cube
    | CubeRoot
    | '(',Expression,')'
    | Initial_assignment
    | Variable_name
    | Numbers.
SqaureRoot ::= 'sqrt', '(' Expression ')'.
CubeRoot ::= 'cbrt', '(' Expression ')'.
Sqaure ::= 'sq' ,'(' Expression ')'.
```

```
% Boolean condition include logical operators('and','or','not'), [Extra(Comparison operators), also taking care of condition when variable is intialised]

Bool_condition ::= And_condition, 'or', Bool_condition
    | And_condition.
And_condition ::= Condition, 'and', And_condition
    | Condition.
Condition ::= 'not', Bool_condition
    | '(',Bool_condition,')'
    | Initial_assignment
    | 'not', Expression
    | Expression, '==', Expression
    | Expression, '<', Expression
    | Expression, '>', Expression
    | Expression, '<=', Expression
    | Expression, '>=', Expression
    | Bool_value.
```

Grammar

```
%%%%%%%%%%%%%%%% IF STATEMENTS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
If_command :- If_part.
If_else_command :- If_part, Else_part.
If_else_ladder_command :- If_part, Elseif_part, Else_part.

If_part :- 'if','(',Bool_condition,')','{',CommandList,'}' .
Elseif_part :- 'else if', '(',Bool_condition,')' ,'{',CommandList,'}', Elseif_part.
Elseif_part :- 'else if', '(',Bool_condition,')' ,'{',CommandList,'}'.
Else :- 'else' ,'{',CommandList,'}'.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%WHILE STATEMENT%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
While_command :- 'while','(',Bool_condition,')', '{',CommandList,'}'.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%FOR STATEMENT%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

For_command :- 'for','(',Assignment,';','Bool_condition',';',Variable_updation,')','{',CommandList,'}'.


Variable_updation :- Increment_operation
    | Decrement_Operation
    | Variable_name '=' Expression.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% INCREMENT OPERTORS(Post and Pre)%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


Increment_operation :- Variable_name,'++'
    | '++',Variable_name.



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% DECREMENT OPERTORS(Post and Pre)%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


Decrement_Operation :- Variable_name,'--'
    | '--',Variable_name.



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%FOR IN RANGE STATEMENT%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


For_in_range_command :- 'for',Variable_name,'in','range', '(',Range_Val,',',Range_Val,')','{',CommandList,'}'.
Range_Val ::= Variable_name
    | Numbers.
```

Grammar

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%SINGLE  OPERATIONS%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


Ternary_operator ::= '(',Bool_condition, ')','?',Expression ':' Expression
    |  '(',Bool_condition, ')','?','{',CommandList,'}' ':' '{',CommandList,'}'.

Print ::= 'print','(',Expression,')'.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Finding length of String %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
String_length ::= 'length','(',String_Value,')'.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%% STRING VALUE SHOULD be with double qoutes and can be alphanumeric %%%%%%%%%%%%%%%%%%%%
String_Value ::= '\"',Alphanumeric,'\"'.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Float Values can be decimal as well as integers %%%%%%%%%%%%%%%%%%%%%
Float_value ::= Numbers,'.', Numbers
    | Numbers.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Bool Values can be true, false, 0 and 1 %%%%%%%%%%%%%%%%%%%%%
Bool_value ::= 'true' | 'false' | '0' | '1'.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%% Variable_name should always start with lowercase and can be followed by Uppercase_letter or digit or '_', but it can't end with '_'.
Variable_name ::= Lowercase_letter
    | Lowercase_letter, { Lowercase_letter | Uppercase_letter | Digit | '_' }, Lowercase_letter
    | Lowercase_letter, { Lowercase_letter | Uppercase_letter | Digit | '_' }, Uppercase_letter
    | Lowercase_letter, { Lowercase_letter | Uppercase_letter | Digit | '_' }, Digit .

%%%%%%%%%%%%%%%%%%%%%%%%%% Alphanumeric characters are combination of letters, symbols and digits.
Alphanumeric ::= Character, Alphanumeric
    | Character.
Character ::= Letter | Numbers | Special.
Numbers ::= Digit, Numbers
    | Digit.
Letter ::= Lowercase_letter | Uppercase_letter.
Lowercase_letter ::= 'a' | 'b' | 'c' | 'd' | 'e' | 'f' | 'g' | 'h' | 'i' | 'j' | 'k' | 'l' | 'm' | 'n' | 'o' | 'p' | 'q' | 'r' | 's' | 't' | 'u' | 'v' | 'w' | 'x' | 'y' |
Uppercase_letter ::= 'A' | 'B' | 'C' | 'D' | 'E' | 'F' | 'G' | 'H' | 'I' | 'J' | 'K' | 'L' | 'M' | 'N' | 'O' | 'P' | 'Q' | 'R' | 'S' | 'T' | 'U' | 'V' | 'W' | 'X' | 'Y' |
Digit ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'.
Special ::= '!' | '@' | '#' | '$' | '%' | '^' | '&' | '*' | '(' | ')' | '-' | '+' | '=' | '{' | '}' | '[' | ']' | ':' | ';' | ',' | '.' | '<' | '>' | '/' | '?' | '~' | '`
```