CE305-ASSIGNMENT-TWO

REPORT

BORISLAV KOLEV

STUDENT ID: 1801355

Main Features:

- Compiles code from external text filed to .py files.
- Language supports variable declaration and assignment, with type checking.
- Language can use if, while and function statements.
- Supports all basic arithmetic and Boolean operation on variables.
- Having some error handling exceptions.
- Generates an Abstract Syntax Tree in the console.

Language Syntax:

- Can use both Integers and Float values. Conversion is done implicitly.
- Statements of "while", "if" and "else if" are structured with conditions in brackets '(' ')', and the code is executed inside curly brackets '{' '}'. Only "else" does not have conditions bracket and the code is directly executed in the curly brackets.
- Declaration, assignment, function, function return, while statements and expressions must end with ';'. Only when using the "if" statements we put ';' at the end of an "else" statement.

Variables:

- Three types of variables are supported:
  - Int
  - Float
  - Bool
- Integer values are whole positive or negative numbers.
- Float values are and positive and negative numbers with some decimal value attacked after the '.'
- Boolean values are either "True" or "False".
- Variables must be declared before they are to be used in any way.
- To declare a variable, we do: type variable-name; int val;
- After declaration we can either assign a value to it or use it in other statements. To assign a value we use '=' like: variable-name = value; val = 10;
- Variables can be assigned expressions and need to be surrounded by brackets like:

bool variable-name = (expression); bool a = (5<6);

- Variables can use arithmetic operations with the “=” character to denote that the variable will use itself in the calculation. Meaning that the variable after the “=” will use itself once plus the other variable that is declared. Example:
  value += 4;   is equal to   value = value + 4;

Functions:

- Functions are declared with the type, the name, variables inside brackets for that function, then all content of the function in curly brackets ‘{‘ ‘}’ like:
  type variable-name(variables) {code};   int func(int n) { code };
- A return statement must be used to return values from the function if it’s type is set to “int”, “float” or “bool”.
- Functions have a fourth type supported “void”, where all the code inside the curly brackets is executed but a return statement is not needed.

Expressions:

While Loop Statements:

- Loops are structured like:
  while (variable or condition) { expressions; };   while (a <= 4) { code };
- Basic Boolean values can be used in the brackets
- When writing statements in the curly brackets, no indentation is required.

If-Else Statements:

- If-Else statements are structured like:
  If (variable or condition) { expression; }
  else if(variable or condition) {expression; }
  else { expression; };
- An If-Else statement must always end with an “else” statement and a ‘;’ at the end of the curly brackets.
- There can be many “else if” statements but only one “else” statement.
- No indentation is required like in the while loops.

Operations:

Arithmetic Operations:

- Number variables can have these operations applied:
  - + Add
  - - Subtract
  - * Multiply
  - / Divide
  - ^ Power
  - % Modulo
- Multiple operation can be applied together:
  4 * 2 + 3 ^ 3 – 15 / 3

Boolean Logic:

- Numbers can be compared using these operations for Boolean result:
    - > Greater Than
    - < Less Than
    - >= Greater Than or Equal To
    - <= Less Than or Equal To
- All variables can be compared using:
    - == Equal To
    - != Not Equal To
- We can use multiple Boolean logic using comparators like:
    - && And
    - || Or

Error Handling:

- There are a couple of error handling exceptions created in the code. They are:
    - DuplicateDefinition which throws an error when it finds duplicates in the scope.
    - IncorrectDataType which throws an error for excepting a different data type.
    - Keyword which throws an error when a keyword that is reserved has been used as an identifier.
    - UndefinedFunction throws an error when the function has not been defined in the scope.
    - UndefinedFunctionReturn throws an error when a non-void data type function does not have a return statement.
    - UndefinedVariable throws an error when a variables has not been defined in the scope.
    - And UnsupportedDataType throws an error when the user tries to use a data type that is currently not yet implemented or supported.

Language Tokens:

- Add – the + character. Identifier for addition.
- Subtract - the – character. Identifier for subtraction.
- Multiply – the * character. Identifier for multiplication.
- Divide – the / character. Identifier for division
- Power – the ^ character. Identifier for power calculation.
- Modulo – the % character for modulo calculation.
- Assign – uses = to denote when a variable is being assigned.
- AssignPlus – usus the += for a variable operation of addition assignment.
- AssignMinus – uses the -= for a variable operation of subtraction assignment.
- AssignMultiply – uses the *= for a variable operation of multiplication assignment.
- AsssignDivide – uses the /= for a variable operation of division assignment.
- AssignPower – uses the ^= for a variable operation of power assignment.
- AssignModulo – uses the %= for a variable operation of modulo assignment.
- Not – uses ! meaning not equal to comparison.
- GreaterThan – uses > greater than character for comparison.

- LessThan – uses < less than character for comparison.
- Equals – uses == symbol for equals to comparison.
- NegativeEquals – uses != symbol for does no equal to comparison.
- GreaterThanEquals – uses >= symbol for a greater than or equals to comparison.
- LessThanEquals – uses <= symbol for a less than or equal to comparison.
- Or – uses || for a logical OR symbol
- And – uses && for a logical AND symbol
- Int – int any string of number characters from 0-9
- Float – float string of numbers followed by a '.' And at least one number after
- Bool – bool represent a True/False value
- Void – void is data type that does not require any return statement
- If – if used to denote the beginning of an If statement
- Else – else used to denote the beginning of an else statement
- While – while used to denote the beginning of a while loop
- Return – return used to return a value from a function
- LeftPara – uses (character for encapsulate expression for variable or statement
- RightPara – uses) character for end of encapsulation
- LeftBracket – uses { character for nesting
- RightBracket – uses } character for end of nesting
- Comma- uses ',' character to separate multiple inputs
- Variable – the name for the variable
- Number – the number for the variable
- EndOfLine – uses ; character to signal the end of the statement
- Space – is any blank space, which is ignored by the program