Scorpion Tokens

Token Classes Description

Keywords

- **Function:** Represents the keyword `func` used to declare functions.
- **ChildFunction:** Represents the keyword `cfunc` used to declare child functions.
- **Variable:** Represents the keyword `var` used to declare variables.
- **Constant:** Represents the keyword `const` used to declare constants.
- **Integer:** Represents the keyword `int` used to declare integer types.
- **Boolean:** Represents the keyword `bool` used to declare boolean types.
- **Array:** Represents the keyword `arr` used to declare arrays.
- **Tuple:** Represents the keyword `tuple` used to declare tuples.
- **Exception:** Represents the keyword `Exception` used for exception handling.
- **True:** Represents the keyword `true` for boolean true.
- **False:** Represents the keyword `false` for boolean false.
- **If:** Represents the keyword `if` for conditional statements.
- **Elseif:** Represents the keyword `elseif` for additional conditional statements.
- **Else:** Represents the keyword `else` for alternative conditional statements.
- **Return:** Represents the keyword `return` for returning values from functions.
- **Void:** Represents the keyword `void` for functions with no return type.
- **Try:** Represents the keyword `try` for exception handling.
- **Throw:** Represents the keyword `throw` for throwing exceptions.
- **Catch:** Represents the keyword `catch` for catching exceptions.
- **Print:** Represents the keyword `print` for printing output.
- **For:** Represents the keyword `for` for loop constructs.
- **While:** Represents the keyword `while` for while loop constructs.
- **Length:** Represents the keyword `len` for obtaining the length of arrays or strings.
- **Slice:** Represents the keyword `slice` for slicing operations.
- **Break:** Represents the keyword `break` for breaking out of loops.
- **Continue:** Represents the keyword `continue` for continuing to the next iteration of a loop.
- **Or:** Represents the keyword `or` for logical OR operations.
- **And:** Represents the keyword `and` for logical AND operations.
- **Not:** Represents the keyword `not` for logical NOT operations.
- **String_k:** Represents the keyword `string` for string type.
- **Cons:** Represents the keyword `cons` for constructing data structures.
- **Head:** Represents the keyword `head` for accessing the head of a list.
- **Tail:** Represents the keyword `tail` for accessing the tail of a list.
- **Format:** Represents the keyword `format` for string formatting.
- **Substr:** Represents the keyword `substr` for substring operations.
- **Type:** Represents the keyword `type` for defining custom types.
- **Main:** Represents the keyword `main` for the main function.

Identifiers and Literals

- **Identifier:** Represents identifiers (variable names, function names, etc.) in the code.

- **Number:** Represents numeric literals in the code.
- **Char:** Represents character literals in the code.
- **String:** Represents string literals in the code.

Operators

Binary Operators

- **Assign (`=`):** Represents the assignment operator.
- **Equal (`==`):** Represents the equality operator.
- **Plus (`+`):** Represents the addition operator.
- **Increment (`++`):** Represents the increment operator.
- **Minus (`-`):** Represents the subtraction operator.
- **Decrement (`--`):** Represents the decrement operator.
- **Star (`*`):** Represents the multiplication operator.
- **Slash (`/`):** Represents the division operator.
- **Mod (`%`):** Represents the modulo operator.
- **Less (`<`):** Represents the less than operator.
- **LessEqual (`<=`):** Represents the less than or equal to operator.
- **Greater (`>`):** Represents the greater than operator.
- **GreaterEqual (`>=`):** Represents the greater than or equal to operator.
- **NotEqual (`!=`):** Represents the not equal to operator.
- **PlusEqual (`+=`):** Represents the addition assignment operator.
- **SlashEqual (`/=`):** Represents the division assignment operator.
- **StarEqual (`*=`):** Represents the multiplication assignment operator.
- **MinusEqual (`-=`):** Represents the subtraction assignment operator.
- **ModEqual (`%=`):** Represents the modulo assignment operator.
- **Power (``):** Represents the exponentiation operator.
- **BitwiseNot (`~`):** Represents the bitwise NOT operator.
- **BitwiseAnd (`&`):** Represents the bitwise AND operator.
- **BitwiseOr (`|`):** Represents the bitwise OR operator.
- **AndEqual (`&=`):** Represents the bitwise AND assignment operator.
- **OrEqual (`|=`):** Represents the bitwise OR assignment operator.
- **LeftShift (`<<`):** Represents the left shift operator.
- **RightShift (`>>`):** Represents the right shift operator.
- **LeftShiftEqual (`<<=`):** Represents the left shift assignment operator.
- **RightShiftEqual (`>>=`):** Represents the right shift assignment operator.

Unary Operators

- **Bang (`!`):** Represents the logical NOT operator.
- **Minus (`-`):** Represents the unary minus operator.
- **Plus (`+`):** Represents the unary plus operator.
- **Dot (`.`):** Represents the dot operator.

Delimiters

- **Comma (`,`):** Separates elements in a list or function arguments.
- **Colon (`:`):** Used in various contexts, such as in dictionaries, for loop syntax, etc.
- **Semicolon (`;`):** Separates statements in some programming languages.

- **Left Parenthesis (`(`):** Represents the opening of a grouping or function call.
- **Right Parenthesis (`)`):** Represents the closing of a grouping or function call.
- **Left Brace (`{`):** Represents the opening of a block of code or a dictionary.
- **Right Brace (`}`):** Represents the closing of a block of code or a dictionary.
- **Left Bracket (`[`):** Represents the opening of a list or array.
- **Right Bracket (`]`):** Represents the closing of a list or array.

Special Tokens

- **Illegal:** Represents an error token.
- **EOF:** Represents the end of the file.