SER502 Project - NatLang
An English-Inspired Programming Language

Contribution: Team 8
- Thrupthi Hosahalli Manjunatha
- Mrunal Kapure
- Ayush Desai
- Piyush Sharma

Ira A. Fulton Schools of Engineering
Arizona State University

Introduction

NatLang

- **Overview** : The beginner friendly, English inspired programming language.
- **Problem**: Traditional programming languages can be intimidating for beginners.
- **Solution** : Creating a language that bridges natural English and programming.
- **Target audience**: Non-programmers, students, and beginners.
- Logical thinking and programming approaches through natural language like syntax.

Language Design

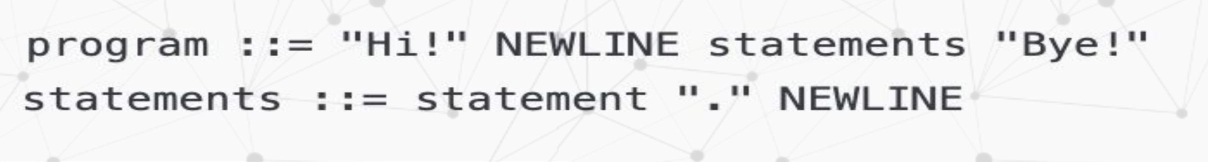
- Conversational approach (Hi/Bye structure).
- English-like syntax for readability.
- Simplified keywords that match intuitive understanding.
- Focus on educational value over performance.
- Designed to make coding more accessible and human-readable.
- The design focuses on 3 things - Simplicity, Expressiveness, Fun!

Language Features

- Program structure (Hi/Bye framework)
- Variable declarations and assignments
- Control structures (conditionals, loops)
- Output Capabilities
- Expressions and operators
- Execution flow : Input program → Tokenizer → Parser → Evaluator → Output!

Syntax Showcase

- NatLang vs. traditional languages
- English-like syntax maps to programming concepts



```
program ::= "Hi!" NEWLINE statements "Bye!"
statements ::= statement "." NEWLINE
```

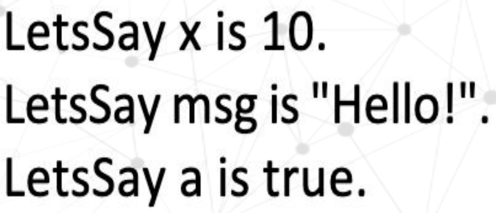
- ✅ Variable declarations
- ✅ Aliases (references)
- ✅ Assignments & Expressions
- ✅ Output with `Show`
- ✅ If-Else & Ternary conditions
- ✅ `ForAll` and `Until` loops
- ✅ Logical, Arithmetic, and Comparison ops

Syntax Comparison

Traditional Code	NatLang Equivalent
x = 5	LetsSay x is 5.
if (x > 5) { … }	When x IsGreaterThan 5 Then ...
print(x + y * z)	Show x plus y times z.
while (counter < 5) { … }	Until counter IsEqualTo 5: ...

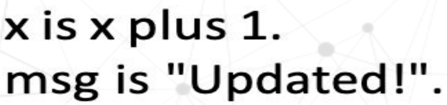
Supported Structure

Variable Declarations



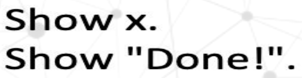
```
LetsSay x is 10.
LetsSay msg is "Hello!".
LetsSay a is true.
```

Assignment



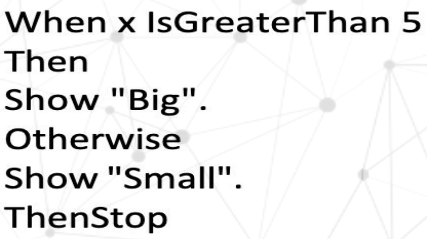
```
x is x plus 1.
msg is "Updated!".
```

Output



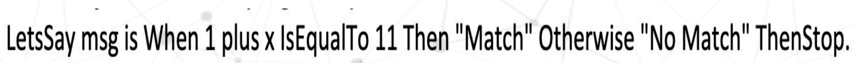
```
Show x.
Show "Done!".
```

If-Else Condition



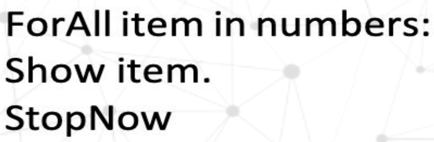
```
When x IsGreaterThan 5
Then
Show "Big".
Otherwise
Show "Small".
ThenStop
```

Ternary Condition



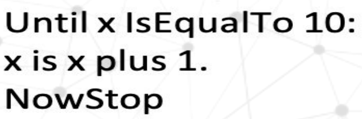
```
LetsSay msg is When 1 plus x IsEqualTo 11 Then "Match" Otherwise "No Match" ThenStop.
```

For Loop



```
ForAll item in numbers:
Show item.
StopNow
```

Until Loop



```
Until x IsEqualTo 10:
x is x plus 1.
NowStop
```

Operators & Expressions

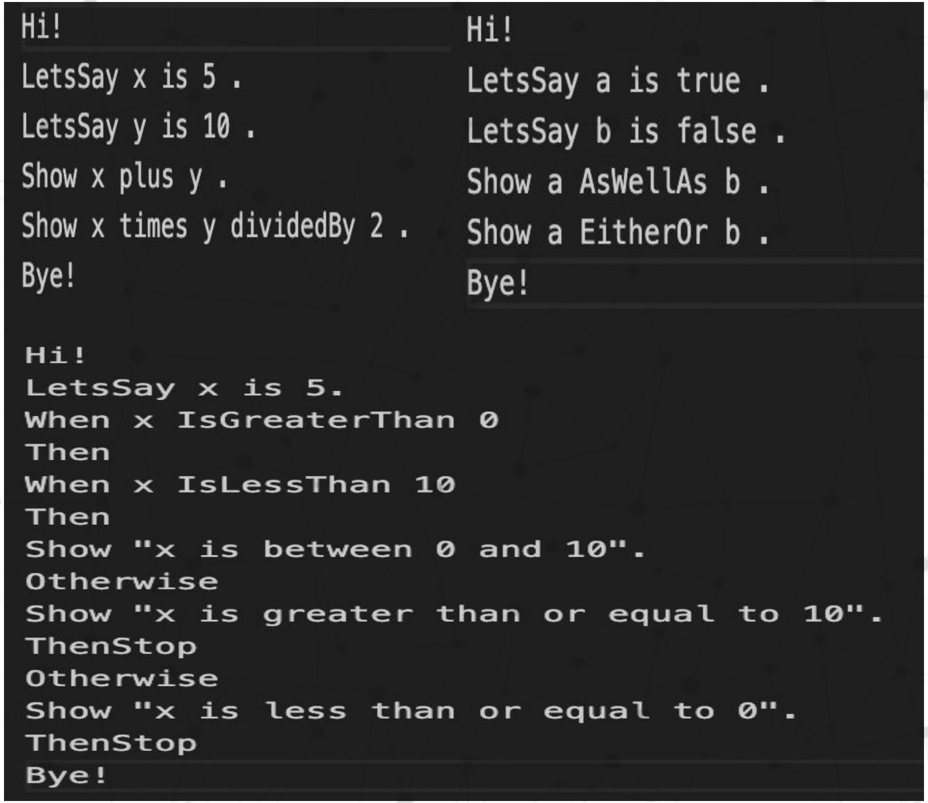
Arithmetic Operators

Operator	Symbol	Precedence
Parentheses	(...)	Highest
Multiplicative	times, dividedBy	High
Additive	plus, minus	Medium
Logical	AsWellAs, EitherOr	Low

Comparison Operators

Used in When or Until conditions:

- IsEqualTo
- IsNotEqualTo
- IsGreaterThan
- IsLessThan
- IsAtLeast
- IsAtMost
- IsNot



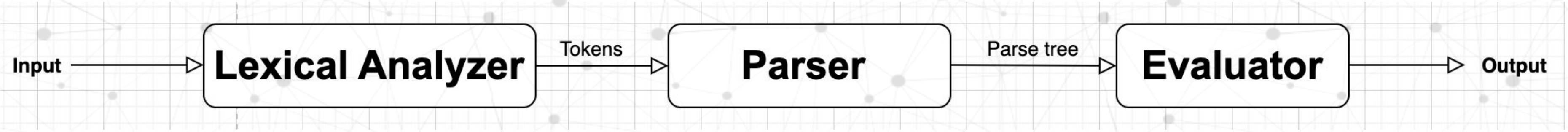
```
Hi!
LetsSay x is 5 .
LetsSay y is 10 .
Show x plus y .
Show x times y dividedBy 2 .
Bye!
```

```
Hi!
LetsSay a is true .
LetsSay b is false .
Show a AsWellAs b .
Show a EitherOr b .
Bye!
```

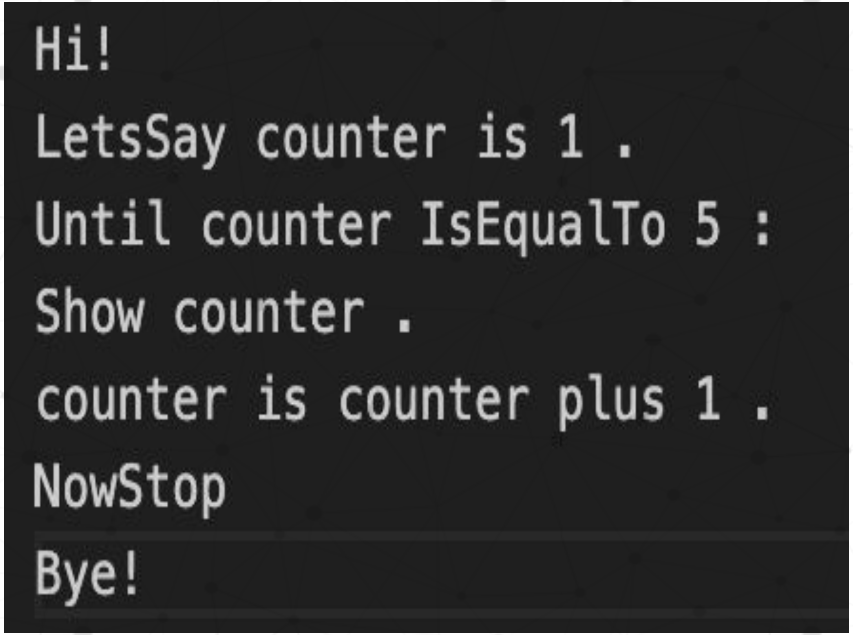
```
Hi!
LetsSay x is 5.
When x IsGreaterThan 0
Then
When x IsLessThan 10
Then
Show "x is between 0 and 10".
Otherwise
Show "x is greater than or equal to 10".
ThenStop
Otherwise
Show "x is less than or equal to 0".
ThenStop
Bye!
```

Technical Implementation

- **Lexical Analyzer (Python)**: Tokenizes NatLang input.
- **Parser (Prolog – DCG)**: Validates grammar and generates parse tree.
- **Evaluator/Interpreter (Python)**: Traverses parse tree and executes logic.



Example Walkthrough



```
Hi!
LetsSay counter is 1 .
Until counter IsEqualTo 5 :
Show counter .
counter is counter plus 1 .
NowStop
Bye!
```

Tokenizer :

['Hi!', '\n', 'LetsSay', counter, 'is', 1, '.', '\n', 'Until', counter, 'IsEqualTo', 5, ':', '\n', 'Show', counter, '.', '\n', counter, 'is', counter, 'plus', 1, '.', '\n', 'NowStop', '\n', 'Bye!'].

Parser :

ParseTree = *program(*hi,[*declare(*counter,*number(*1)), *until(condition(identifier(*counter),'IsEqualTo',*number(*5)),[*output(identifier(*counter)), *assign(*counter,*operator(*plus,*identifier(*counter),*number(*1))))]),bye)

Evaluator :

1
2
3
4

SER502 Project - NatLang
DEMO

Team 8

- Thrupthi Hosahalli Manjunatha
- Mrunal Kapure
- Ayush Desai
- Piyush Sharma

Conclusion

- Implemented complete toolchain: Lexer, Parser, and Interpreter.

- Designed intuitive control structures and expressions that mirror natural conversations.

- Emphasizes understanding concepts over memorizing syntax.

- Demonstrates how programming languages can adapt to humans rather than the reverse.

Thank You!