

# Nex Programming Language

A powerful, interpreted programming language built with Python.

# Team Members

Jomar Lazaro

Jarren Aceret

Derek Luis Lagadon

Gabriel Andrei Adajar

# Introduction

**Nex** features:

- Clean syntax
- Dynamic typing
- Modular architecture

# Getting Started

## Prerequisites

- Python 3.x

## Running the Shell

To start the interactive REPL:

```
python shell.py
```

# Variables & Data Types

Nex supports dynamic typing. Use `VAR` to declare variables.

```
VAR age = 25  
VAR pi = 3.14  
VAR name = "Nex"  
VAR numbers = [1, 2, 3]
```

# Arithmetic Operators

- `+` : Addition
- `-` : Subtraction
- `*` : Multiplication
- `/` : Division
- `^` : Power

# Comparison & Logical Operators

Comparison	Logical
<code>==</code> Equal	<code>AND</code> Logical AND
<code>!=</code> Not Equal	<code>OR</code> Logical OR
<code>&lt;</code> Less than	<code>NOT</code> Logical NOT
<code>&gt;</code> Greater than	
<code>&lt;=</code> Less or equal	
<code>&gt;=</code> Greater or equal	

# Control Flow: If-Elif-Else

```
VAR x = 10

IF x > 5 THEN
    PRINT("Greater than 5")
ELIF x == 5 THEN
    PRINT("Equal to 5")
ELSE
    PRINT("Less than 5")
END
```

# Loops

## For Loops

```
FOR i = 0 TO 4 THEN  
    PRINT(i)  
END
```

## While Loops

```
VAR i = 0  
WHILE i < 5 THEN  
    PRINT(i)  
    VAR i = i + 1  
END
```

# Functions

## Block Syntax

```
FUN add(a, b)
    RETURN a + b
END
```

## Arrow Syntax

```
FUN multiply(a, b) -> a * b
```

# Built-in Functions

- `PRINT(value)`
- `INPUT()`, `INPUT_INT()`
- `LEN(list)`, `APPEND(list, value)`, `POP(list, index)`
- `IS_NUM(value)`, `IS_STR(value)`, `IS_LIST(value)`
- `RUN(filename)`

# Project Structure

The project is modularized for maintainability:

- **Core:** `nex.py`, `shell.py`
- **Parsing:** `lexer.py`, `parser.py`, `tokens.py`
- **Execution:** `interpreter.py`, `values.py`, `nodes.py`
- **Utils:** `errors.py`, `constants.py`

# Project Highlights

- Demonstrates deep understanding of programming concepts
- Includes advanced features or optimizations

# Conclusion

The **Nex** programming language project successfully demonstrates the implementation of a functional interpreter using Python.

## Key Takeaways:

- Practical application of lexing, parsing, and interpreting.
- Understanding of language design principles.
- Creation of a modular and extensible codebase.

# Thank You!

Start coding with **Nex** today.