

NOOBIE2 Programming Language

Complete Documentation

Luca Tarantola

June 2025

Contents

1	Introduction	3
2	Simple NOOBIE	3
2.1	Data Types	3
2.2	Basic Commands	3
2.2.1	SAY - Display Output	3
2.2.2	CREATE - Variable Declaration	4
2.2.3	LISTEN - User Input	4
2.2.4	CHANGE - Modify Variables	4
2.2.5	WHILE - Loop Execution	5
2.2.6	EXIT - Program Termination	5
2.3	Basic Conditionals	5
2.4	Simple Examples	6
2.4.1	Hello World Program	6
2.4.2	Personal Greeting	6
2.4.3	Simple Calculator	6
2.4.4	Counting Loop	7
2.4.5	Program with User Control	7
3	Advanced NOOBIE	8
3.1	Advanced Variable Operations	8
3.1.1	Type Conversion	8
3.1.2	Variable Manipulation	9
3.2	Mathematical Operations	10
3.3	String Operations	10
3.4	Random Number Generation	11
3.5	Advanced Program Control	12
3.5.1	Conditional Program Termination	12
3.5.2	Menu-Driven Programs with EXIT	12
3.6	Advanced Syntax Features	13

3.6.1	Special Syntax Elements	13
3.6.2	Comments	13
3.7	Complex Examples	14
3.7.1	Advanced Calculator	14
3.7.2	Number Guessing Game	15
3.8	Best Practices	16
4	Quick Reference	17
4.1	Command Summary	17
4.2	Tips for Success	17
5	Conclusion	18

1 Introduction

NOOBIE is a simple, intuitive programming language specifically designed for beginners who are taking their first steps into the world of programming. Unlike complex languages that can overwhelm newcomers, NOOBIE focuses on readability, simplicity (is case insensitive), and educational value.

The language emphasizes natural language constructs, making it easy for students to understand programming concepts without getting bogged down in complex syntax. NOOBIE version 2.2 supports fundamental programming constructs including variables, data types, user input/output, conditionals, and basic operations.

This documentation is divided into two main sections: **Simple NOOBIE** covers the basic concepts and commands needed to write your first programs, while **Advanced NOOBIE** explores more sophisticated features for users ready to tackle complex programming challenges.

2 Simple NOOBIE

This section covers the fundamental concepts and commands that every NOOBIE2 programmer should master first.

2.1 Data Types

NOOBIE supports five basic data types that cover most programming needs:

- **INT**: Integer numbers for whole values (e.g., 42, -10, 0)
- **FLOAT**: Decimal numbers for precise calculations (e.g., 3.14, -0.5, 2.0)
- **STR**: Text strings for words and sentences (e.g., "Hello", 'World')
- **CHAR**: Single characters for individual letters or symbols (e.g., 'A', '5', '!')
- **BOOL**: Boolean values for true/false logic (true, false, null)

2.2 Basic Commands

2.2.1 SAY - Display Output

The SAY command displays text or variable values to the user.

Syntax: SAY "message" or SAY `variable_name`

```
1 SAY "Hello , World!"
2 SAY name
3 SAY "Your age is: @age"
4 SAY "{5+5}"
```

The Reserved Word "end": The word `end` is a special reserved variable in NOOBIE2 that represents a newline character. When used in output commands like `SAY`, it automatically inserts a line break. This is particularly useful for formatting output across multiple lines. You can use `end` anywhere in your output where you want to create a new line, making your program's output more readable and well-formatted.

```
1 SAY "First line" end "Second line" end
2 SAY "Hello" end end "This has two line breaks"
```

2.2.2 CREATE - Variable Declaration

The `CREATE` command declares new variables with optional initial values.

Syntax: `CREATE [CONST] <type> <name> [value]`

```
1 CREATE INT age 25
2 CREATE STR name "John"
3 CREATE STR surname Mark
4 CREATE CONST FLOAT pi 3.14159
5 CREATE BOOL is_student
```

If the value of the variable is not specified, it will be automatically set as: `0` for `INT`, `0.0` for `FLOAT`, `""` for `STR`, `'\0'` for `CHAR`, `null` for `BOOL`

2.2.3 LISTEN - User Input

The `LISTEN` command gets input from the user and stores it in a variable.

Syntax: `LISTEN <type> [variable_name] "prompt"`

```
1 LISTEN STR name "Enter your name: "
2 LISTEN INT age "How old are you? "
3 LISTEN BOOL "Are you a student? (true/false): "
4 LISTEN INT num prompt_variable
```

Variable Name Omission: If you don't specify a variable name in the `LISTEN` command, NOOBIE2 will automatically create a variable called `listened` to store the user's input. This is useful for quick input operations where you don't need a specific variable name.

```
1 LISTEN STR "Enter something: "
2 SAY "You entered: " listened
```

2.2.4 CHANGE - Modify Variables

The `CHANGE` command updates the value of existing variables.

Syntax: `CHANGE <variable> <new_value>`

```
1 CHANGE age 26
2 CHANGE message "Updated text"
3 CHANGE is_active true
4 CHANGE name @new_name
5 CHANGE sum {current_value + sum}
```

2.2.5 WHILE - Loop Execution

The WHILE command repeats a block of code as long as a condition remains true.

Syntax: WHILE <condition> DO <code> ENDO

```
1 CREATE INT counter 1
2 WHILE counter <= 5 DO
3     SAY "Count: " counter
4     INCREMENT counter
5 ENDO
```

2.2.6 EXIT - Program Termination

The EXIT command immediately terminates the program execution. This is useful for ending programs early based on certain conditions or user input.

Syntax: EXIT or EXIT "message"

```
1 # Simple exit
2 EXIT
3
4 # Exit with message
5 EXIT "Program terminated by user"
6
7 # Conditional exit
8 CREATE STR user_choice
9 LISTEN STR user_choice "Do you want to continue? (
    yes/no): "
10 IF user_choice == "no" DO
11     EXIT "Goodbye!"
12 ENDO
```

2.3 Basic Conditionals

Conditional statements allow your program to make decisions based on different conditions.

Syntax:

```
1 IF <condition> DO
2     <statements>
3 ELSE
4     <statements>
5 ENDO
```

Example:

```
1 CREATE INT age
2 LISTEN INT age "Enter your age: "
3 IF age >= 18 DO
4     SAY "You are an adult!" end
5 ELSE
6     SAY "You are a minor." end
7 ENDO
```

2.4 Simple Examples

2.4.1 Hello World Program

```
1 # My first NOOBIE2 program
2 SAY "Hello, World!" end
3 SAY "Welcome to NOOBIE2 programming!" end
```

2.4.2 Personal Greeting

```
1 # Personal greeting program
2 CREATE STR name
3 CREATE INT age
4
5 LISTEN STR name "What's your name? "
6 LISTEN INT age "How old are you? "
7
8 SAY "Hello, " name "!" end
9 SAY "You are " age " years old." end
```

2.4.3 Simple Calculator

```
1 # Basic addition calculator
2 CREATE INT num1
3 CREATE INT num2
4
5 LISTEN INT num1 "Enter first number: "
6 LISTEN INT num2 "Enter second number: "
7
8 SAY "The sum is: {num1 + num2}" end
```

2.4.4 Counting Loop

```
1 # Count from 1 to 10
2 CREATE INT number 1
3
4 SAY "Counting from 1 to 10:" end
5 WHILE number <= 10 DO
6     SAY number end
7     INCREMENT number
8 ENDO
9 SAY "Finished counting!"
```

2.4.5 Program with User Control

```
1 # Program that lets user exit early
2 CREATE STR response
3
4 SAY "Welcome to the greeting program!"
5 LISTEN STR response "Press Enter to continue or type
    'exit' to quit: "
6
7 IF response == "exit" DO
8     EXIT "Thanks for visiting!"
9 ENDO
10
11 SAY "Hello! Nice to meet you!" end
12 SAY "Program completed successfully." end
```

3 Advanced NOOBIE

This section covers more sophisticated features for users who have mastered the basics and are ready for complex programming challenges.

3.1 Advanced Variable Operations

3.1.1 Type Conversion

Convert variables between different data types using the `CONVERT` command.

Syntax: `CONVERT <variable> <new_type>`

The `CONVERT` command can also use the `?` operator to get the type of another variable and use it for conversion:

```

1 CREATE INT number 42
2 CONVERT number STR
3 SAY "Number as string: " number
4
5 CREATE STR text "123"
6 CONVERT text INT
7 SAY "Text as integer: " text
8
9 # Using ? operator to get variable type
10 CREATE FLOAT decimal 3.14
11 CREATE INT whole 10
12 CONVERT whole ?decimal # Converts whole to FLOAT
    type
13 SAY "Whole as float: " whole " (type: " ?whole ")"
```

Type Conversion Rules:

- **INT to FLOAT:** Direct conversion (e.g., $42 \rightarrow 42.0$)
- **INT to CHAR:** Converts to ASCII character if value is 0-127
- **INT to STR:** Converts to string representation (e.g., $42 \rightarrow "42"$)
- **INT to BOOL:** 0 becomes false, any other value becomes true
- **FLOAT to INT:** Truncates decimal part (e.g., $3.14 \rightarrow 3$)
- **FLOAT to CHAR:** Converts integer part to ASCII if 0-127
- **FLOAT to STR:** Converts to string representation
- **FLOAT to BOOL:** 0.0 becomes false, any other value becomes true

- **CHAR to INT:** Converts to ASCII value (e.g., 'A' → 65)
- **CHAR to FLOAT:** ASCII value as float
- **CHAR to STR:** Character becomes single-character string
- **CHAR to BOOL:** Based on ASCII value (0, space, tab, '0' are false)
- **BOOL to INT:** true → 1, false → 0
- **BOOL to FLOAT:** true → 1.0, false → 0.0
- **BOOL to STR:** true → "true", false → "false"
- **BOOL to CHAR:** true → '1', false → '0'
- **STR to INT:** Converts to string length
- **STR to FLOAT:** String length as float
- **STR to BOOL:** Empty/whitespace strings are false, others true
- **STR to CHAR:** Works only for single-character strings

3.1.2 Variable Manipulation

Advanced commands for manipulating variable values:

```

1 # Increment and decrement
2 CREATE INT counter 5
3 INCREMENT counter      # counter becomes 6
4 DECREMENT counter      # counter becomes 5
5
6 # Swap variables
7 CREATE STR first "Hello"
8 CREATE STR second "World"
9 SWAP first second      # first="World", second="Hello"
10
11 # Reset and delete
12 RESET counter          # Sets to default value for
    type
13 DEL unused_variable    # Removes variable from memory

```

3.2 Mathematical Operations

NOOBIE2 supports different operators:

- **Arithmetic Operators:** +, -, *, /, %, **
- **Relational Operators** <, >, !=, ==, >=, <=
- **Logical Operators** and, or, not, xor
- **Value Operators** @ (expand the value of a variable), ? (expand the type of a variable, only in strings), {} (evaluate expression in strings)
- **Precision Control:** ROUND for floating-point precision

```
1 CREATE FLOAT pi 3.14159265
2 CREATE INT base 2
3 CREATE INT exponent 8
4
5 SAY "Pi rounded to 2 places: "
6 ROUND pi 2
7 SAY pi
8
9 SAY "2 to the power of 8: {base ** exponent} " end
10 SAY "17 divided by 5: {17 / 5} " end
11 SAY "17 modulo 5: {17 % 5} " end
```

3.3 String Operations

Advanced string manipulation capabilities:

```
1 CREATE STR message "Hello World"
2
3 # String transformations
4 UPPERCASE message      # "HELLO WORLD"
5 SAY "Uppercase: " message
6
7 LOWERCASE message      # "hello world"
8 SAY "Lowercase: " message
9
10 REVERSE message        # "dlrow olleh"
11 SAY "Reversed: " message
```

3.4 Random Number Generation

Generate random values for games, simulations, and testing:

Syntax: RANDOM <type> <min> <max> [variable]

```
1 # Generate random integer between 1 and 100
2 RANDOM INT 1 100 dice_roll
3 SAY "You rolled: " dice_roll end
4
5 # Generate random float between 0.0 and 1.0
6 RANDOM FLOAT 0.0 1.0 probability
7 SAY "Random probability: " probability end
8
9 # Generate random character (ASCII 65-90 for
   uppercase letters)
10 RANDOM CHAR 65 90 random_letter
11 SAY "Random letter: " random_letter end
```

Boolean Random Generation: For BOOL type, the range parameters have special meanings:

- **Range 1-2:** Generates only true or false (excludes null)
- **Range 1-3:** Generates true, false, or null (includes all BOOL values)

```
1 # Generate random boolean (true/false only)
2 RANDOM BOOL 1 2 coin_flip
3 IF coin_flip DO
4     SAY "Heads!" end
5 ELSE
6     SAY "Tails!" end
7 ENDO
8
9 # Generate random boolean including null
10 RANDOM BOOL 1 3 three_state
11 IF three_state == true DO
12     SAY "True state" end
13 ELSE
14     IF three_state == false DO
15         SAY "False state" end
```

```
16     ELSE
17         SAY "Null state" end
18     ENDO
19 ENDO
```

3.5 Advanced Program Control

3.5.1 Conditional Program Termination

The EXIT command can be used strategically in complex programs to handle error conditions or user-requested termination:

```
1 # Error handling with EXIT
2 CREATE INT divisor
3 LISTEN INT divisor "Enter a divisor: "
4
5 IF divisor == 0 DO
6     SAY "Error: Cannot divide by zero!" end
7     EXIT "Program terminated due to invalid input"
8 ENDO
9
10 SAY "Result: {100 / divisor}" end
```

3.5.2 Menu-Driven Programs with EXIT

```
1 # Interactive menu program
2 CREATE STR choice ""
3
4 WHILE choice != "4" DO
5     SAY "=== Main Menu ==="
6     SAY "1. Say Hello" end
7     SAY "2. Calculate Sum" end
8     SAY "3. Show Random Number" end
9     SAY "4. Exit Program" end
10
11     LISTEN STR choice "Choose an option: "
12
13     IF choice == "1" DO
14         SAY "Hello, User!" end
```

```
15     ELSE
16         IF choice == "2" DO
17             CREATE INT a
18             CREATE INT b
19             LISTEN INT a "First number: "
20             LISTEN INT b "Second number: "
21             SAY "Sum: {a + b}" end
22         ELSE
23             IF choice == "3" DO
24                 RANDOM INT 1 10 rand_num
25                 SAY "Random number: " rand_num end
26             ELSE
27                 IF choice == "4" DO
28                     EXIT "Thank you for using the
program!"
29                 ELSE
30                     SAY "Invalid choice. Please try
again." end
31                 ENDO
32             ENDO
33         ENDO
34     ENDO
35 ENDO
```

3.6 Advanced Syntax Features

3.6.1 Special Syntax Elements

- **Direct Variable Use:** Simply use the variable name in expressions and output
- **String Variable Output:** Use quotes with variable names for formatted output
- **Mathematical Expressions:** {expression} (evaluates math in strings)
- **Special Variables:** @end (represents newline character)

3.6.2 Comments

NOOBIE2 supports single-line comments:

```
1 # This is a single-line comment
```

```
2 CREATE INT x 10 # Inline comment
```

3.7 Complex Examples

3.7.1 Advanced Calculator

```
1 # Advanced calculator with multiple operations and
  loop
2 CREATE STR continue_calc "yes"
3
4 SAY "=== Advanced Calculator ===" end
5
6 WHILE continue_calc == "yes" DO
7     LISTEN FLOAT num1 "Enter first number: "
8     LISTEN CHAR operation "Enter operation (+, -, *,
  /, **, %): "
9     LISTEN INT num2 "Enter second number: "
10
11     IF operation == "+" DO
12         SAY "Result: " {num1 + num2} end
13     ELSE
14         IF operation == "-" DO
15             SAY "Result: " {num1 - num2} end
16         ELSE
17             IF operation == "*" DO
18                 SAY "Result: " {num1 * num2} end
19             ELSE
20                 IF operation == "/" DO
21                     IF num2 != 0 DO
22                         SAY "Result: " {num1 / num2}
23
24                     ELSE
25                         SAY "Error: Division by zero
  due to error"
26
27                     EXIT "Calculator terminated
  due to error"
28                 ENDIF
29             ENDIF
30         ENDIF
31     ENDIF
32 ENDWHILE
33 END
```

```
27         ELSE
28             SAY "Unknown operation!" end
29         ENDO
30     ENDO
31 ENDO
32 ENDO
33
34     LISTEN STR continue_calc "Continue? (yes/no): "
35     IF continue_calc == "no" DO
36         EXIT "Calculator session ended by user"
37     ENDO
38 ENDO
39
40 SAY "Thanks for using the calculator!"
```

3.7.2 Number Guessing Game

```
1 # Number guessing game with loops
2 CREATE INT secret_number
3 CREATE INT guess
4 CREATE INT attempts 0
5 CREATE BOOL game_won false
6
7 RANDOM INT 1 100 secret_number
8 SAY "=== Number Guessing Game ===" end
9 SAY "I'm thinking of a number between 1 and 100!"
   end
10 SAY "Type -1 to quit at any time." end
11
12 # Game loop
13 WHILE game_won == false DO
14     LISTEN INT guess "Enter your guess: "
15
16     IF guess == -1 DO
17         EXIT "Thanks for playing! The number was "
           secret_number
```

```
18     ENDO
19
20     INCREMENT attempts
21
22     IF guess == secret_number DO
23         CHANGE game_won true
24         SAY "Congratulations! You guessed it in "
attempts " attempt(s)!" end
25     ELSE
26         IF guess < secret_number DO
27             SAY "Too low! Try again." end
28         ELSE
29             SAY "Too high! Try again." end
30         ENDO
31     ENDO
32 ENDO
33
34 SAY "Thanks for playing!"
```

3.8 Best Practices

1. **Use descriptive variable names:** Choose names that clearly indicate the variable's purpose
2. **Comment your code:** Explain complex logic and important sections
3. **Initialize variables:** Always give variables initial values when possible
4. **Handle edge cases:** Check for division by zero, invalid input, etc.
5. **Use constants for fixed values:** CREATE CONST for values that shouldn't change
6. **Organize your code:** Group related operations together
7. **Test thoroughly:** Try different inputs to ensure your program works correctly
8. **Use EXIT strategically:** Provide clean program termination with informative messages
9. **Handle errors gracefully:** Use EXIT to prevent crashes from invalid operations

4 Quick Reference

4.1 Command Summary

- SAY - Display output
- CREATE - Declare variables
- LISTEN - Get user input
- CHANGE - Modify variables
- CONVERT - Change variable types
- IF/ELSE/END - Conditional execution
- WHILE/END - Loop execution
- RANDOM - Generate random values
- EXIT - Terminate program
- INCREMENT/DECREMENT - Modify numeric values
- UPPERCASE/LOWERCASE/REVERSE - String operations
- SWAP - Exchange variable values
- DEL - Delete variables
- RESET - Reset to default values
- ROUND - Round floating-point numbers

4.2 Tips for Success

- Variables are case-insensitive in NOOBIE2
- Use `{}` for mathematical expressions within strings
- The `@end` variable represents a newline character
- Always test your programs with different inputs
- Start simple and gradually add complexity
- No `@` symbol needed for variable access in most contexts
- Use EXIT to provide clean program termination
- Include helpful messages with EXIT for better user experience

5 Conclusion

NOOBIE2 provides a gentle introduction to programming concepts while offering enough power for meaningful projects. By mastering the Simple NOOBIE concepts first, then progressing to Advanced NOOBIE features, students can build a solid foundation in programming logic and problem-solving.

The language's natural syntax and comprehensive error handling make it an ideal choice for educational environments, coding bootcamps, and self-directed learning. As students become comfortable with NOOBIE2, they'll find the transition to more complex programming languages much smoother.

Remember: every expert programmer started as a beginner. NOOBIE2 is designed to make that journey as enjoyable and productive as possible. Happy coding!