Gemini-Script Documentation

A lightweight programming language with static typing

Overview

Gemini-Script (GS) is a lightweight and modern programming language with static typing, designed to be simple and intuitive. Its syntax is inspired by languages like Python and JavaScript but with a more robust flow control structure, as well as support for basic types such as `Number`, `String`, and `Boolean`.

Gemini-Script allows for variable manipulation, flow control, functions, lists, and more. The language focuses on simplicity and code clarity.

Basic Syntax

1. Variables

In Gemini-Script (GS), variables are declared using the `let` keyword, followed by the variable type, name, and value.

let <Type> <variable_name> = <value>

Variable Types:

- `Number`: Numeric type used for integers.
- `String`: Text type used for strings (text between quotes).

- `Boolean`: Logical type used for True or False values.

Example:

```
let Number age = 30
let String name = "Alice"
let Boolean is_active = True
```

2. Control Flow

Conditional `if`

The `if` statement checks if a condition is true and executes the corresponding block of code.

```
if (<condition>)
    <statements>
```

Example:

```
if (True)
    print("This will print because the condition is True")
```

Conditional `else`

The `else` block executes when the `if` condition is false.

```
if (<condition>)
    <statements>
else
    <statements>
```

Example:

```
if (False)
    print("This will not print")
else
    print("This will print because the condition is False")
```

3. Functions

Functions can be defined using the `func` keyword, followed by the function name and body.

```
func <function_name>()
    <statements>
```

Example:

```
func greet()
    print("Hello, Gemini-Script!")
```

To call the function:

```
greet()
```

4. Printing

The `print()` function is used to display values or variables to the console.

```
print(<value>)
```

Example:

```
print("Hello, Gemini-Script!")
```

5. Variable Assignment

Variables can be reassigned after their declaration using the `=` operator.

```
<variable_name> = <new_value>
```

Example:

```
age = 31
name = "Bob"
```

6. List Manipulation

You can manipulate lists with the `push` and `pop` commands.

- `push(<value>)`: Adds a value to the list.
- `pop()`: Removes the last value from the list.

Example:

```
let List numbers = []
push(numbers, 10)
push(numbers, 20)
pop(numbers)
```

Types

1. `Number`

Used for storing integer values.

Example:

let Number age = 25

2. `String`

Used for storing text between quotes.

Example:

let String name = "Gemini"

3. `Boolean`

Used for storing logical values: True or False.

Example:

let Boolean is_active = True

Control Flow

1. `if`

The `if` statement checks a boolean condition and executes the corresponding block of code.

if (<condition>)

    <statements>

Example:

if (True)

    print("Condition is true")

2. `else`

The `else` statement is used to execute an alternative block of code when the `if` condition is false.

```
if (<condition>)
    <statements>
else
    <statements>
```

Example:

```
if (False)
    print("This will not execute")
else
    print("This will execute")
```

Errors and Debugging

Gemini-Script has a simple error control structure, including type checks and undeclared variable handling.

Example errors:

- Type Error: Attempting to assign a value of an incompatible type to a variable.

```
let Number age = "twenty"  # Error: Invalid value for age. Expected a Number.
```

- Undeclared Variable Error:

```
print(undeclared_variable)  # Error: undeclared_variable not defined.
```

Examples

1. Hello World

```
let String greeting = "Hello, Gemini-Script!"

print(greeting)
```

2. Simple Function

```
func greet()
    print("Hello!")
greet()
```

3. Flow Control with if and else

```
let Boolean is_logged_in = True

if (is_logged_in)
    print("Welcome!")
else
    print("Please log in.")
```