KanbunSE Programming Language

Comprehensive Developer's Guide

23302010090 软件工程 黄浩源

A Classical Chinese S-Expression Programming Language

Table of Contents

1 Introduction
- 1.1 Key Design Philosophy
- 1.2 Target Audience

2 Language Overview
- 2.1 Core Feature Matrix
- 2.2 Unique Language Features

3 Design and Implementation
- 3.1 Architecture Overview
  - 3.1.1 Component Breakdown
- 3.2 Execution Flow
- 3.3 Techincal Implementation Details
  - 3.3.1 Environment Chain Implementation
  - 3.3.2 Advanced Stringification System
  - 3.3.3 Comment Processing

4 Installation and Setup
- 4.1 System Requirements
- 4.2 Installation Steps
  - 4.2.1 Step 1: Install Dependencies
  - 4.2.2 Step 2: Project Setup
- 4.3 Project Structure

5 Data Types and Literals
- 5.1 Primitive Data Types
  - 5.1.1 Numeric Types
  - 5.1.2 String Literals
  - 5.1.3 Boolean Values
  - 5.1.4 List Types
    - 5.1.4.1 Object Types

6 Basic Syntax Reference
- 6.1 Variable Definition and Assignment
  - 6.1.1 Method 1: Classical Declaration Style
  - 6.1.2 Method 2: Naming Style
- 6.2 Arithmetic Operations
- 6.3 Comparison and Logical Operations
  - 6.3.1 Comparison Operations
  - 6.3.2 Logical Operations
- 6.4 Control Flow Structures
  - 6.4.1 Conditional Statements
  - 6.4.2 Pattern Matching

6.5 Loop Constructs
6.5.1 Fixed Iteration Loop
6.5.2 Indexed Loop
6.6 Function Definition and Invocation
6.6.1 Lambad Functions
6.6.2 Function Invocation
7 Object-Oriented Programming
7.1 Class Definition
7.1.1 Standalone Class
7.1.2 Inherited Class
7.2 Comprehensive OOP Example
7.3 Object System Features
7.3.1 Instance Creation
7.3.2 Attriubte Access and Modification
7.3.3 Method Invocation
7.3.4 Self-Reference in Methods
8 Advanced Features
8.1 List Operations and Manipulation
8.1.1 Creating Mutable Lists
8.1.2 List Operations
8.1.3 Nested Lists
8.2 Pattern Matching System
8.3 Chinese Numeral Integration
8.3.1 Supported Numeral Types
8.3.2 Automatic Conversion Examples
9 Example Programs
10 Error Handling
10.1 Error Categories and Messages
10.1.1 Variable Resolution Errors
10.1.2 Type Errors
10.1.3 Syntax Errors
10.1.4 Method Resolution Errors
10.1.5 Index Out of Bounds
11 Performance Notes
12 Conclusion
13 Appendix
13.1 Python Integration

1 Introduction

KanbunSE (**K**anbun **S**-**E**xpression) is an innovative programming language that bridges the gap between classical Chinese literature and modern computational thinking. Built on the foundations of *S-Expression* syntax, KanbunSE preserves the elegance and structure of Classical Chinese while providing comprehensive programming capabilities including object-oriented features, functional programming constructs, and advanced data manipulation.

1.1 Key Design Philosophy

- **Cultural Authenticity**
  Syntax closely follows Classical Chinese grammar patterns and linguistic structures
- **Modern Functionality**
  Full-featured programming language with contemporary software development capabilities
- **Educational Value**
  Demonstrates how programming concepts can be expressed through different linguistic paradigms
- **Practical Application**
  Production-ready interpreter with comprehensive error handling and debugging support

1.2 Target Audience

This guide is designed not only for a report in my handin package, but also for:

- Software developers interested in domain-specific languages
- Students studying compiler construction and language implementation
- Cultural & Computing enthusiasts
- Anyone curious about the intersection of classical Chinese literature and modern programming

---

2 Language Overview

2.1 Core Feature Matrix

Feature Category	Components	Status
Basic Language Constructs	Variables, Operators, Control Flow	✅ Complete
Data Types	Numbers, Strings, Lists, Booleans	✅ Complete
Functions	First-class functions, Closures, Recursion	✅ Complete
Object-Oriented	Classes, Inheritance, Methods, Instances	✅ Complete
Advanced Features	Pattern Matching, List Operations, Error Handling	✅ Complete
External Integration	Chinese Numeral Conversion	✅ Complete

2.2 Unique Language Features

1. **Bidirectional Chinese Numeral Support**
   - Automatic conversion between Arabic numerals and traditional Chinese numerals
   - Support for complex number expressions (e.g., 三百一十四點一五)
   - Integration with external Python conversion utility

2. **Classical Chinese Syntax**
   - Natural language-like programming statements
   - Subject-Verb-Object word order preservation (Partially, due to special grammar in Literal Chinese)
   - Contextually appropriate Classical Chinese grammar
3. **Comprehensive Object Model**
   - Full inheritance hierarchy support based on *struct* definition
   - Method overriding and polymorphism
   - Instance attribute management with proper scoping
4. **Advanced Error Reporting**
   - All error messages presented in Classical Chinese (Except for some native exceptions)
   - Contextual error informatino with environment state
   - Graceful degradation for syntax and runtime errors

---

3 Design and Implementation

3.1 Architecture Overview

The **KanbunSE** interpreter employs a multi-stage architecture designed for extensibility and maintainability:

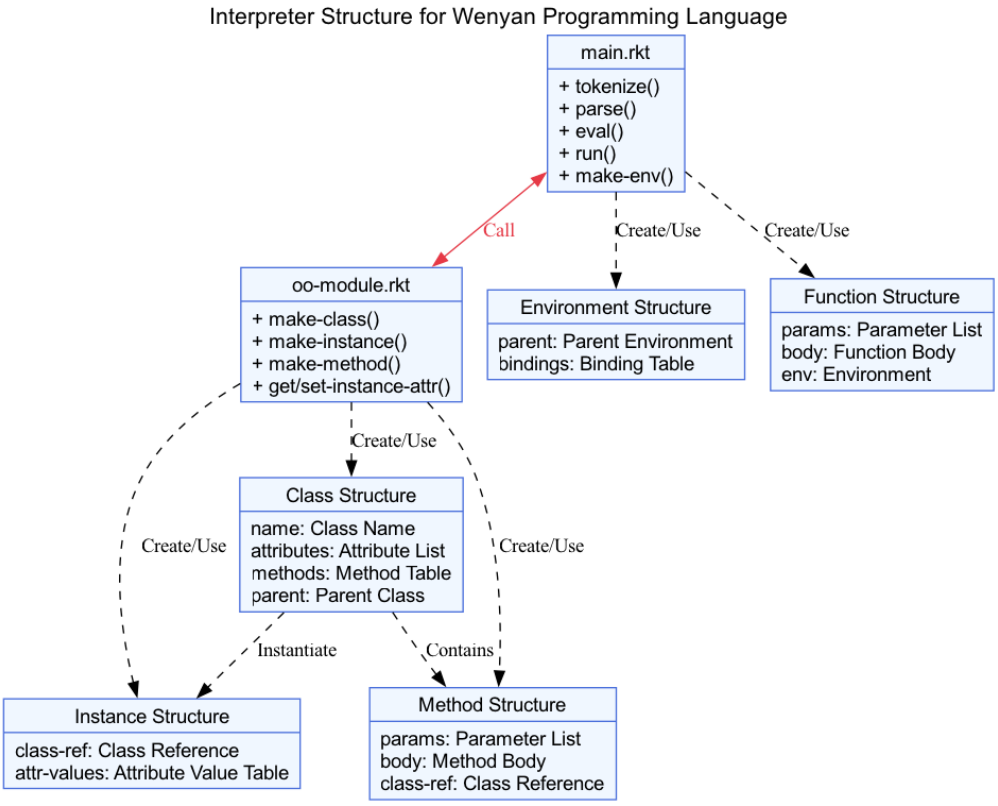

Figure 3.1.1

3.1.1 Component Breakdown

1. **Lexical Analysis Layer**
   - Unicode-aware tokenization for Chinese characters

- Comment removal and preprocessing
- Token classification and validation

2. **Parsing Layer**
   - Recursive descent parser for S-Expressions
   - AST (Abstract Syntax Tree) generation
   - Syntax error detection and reporting
3. **Evaluation Engine**
   - Environment-based evaluation with lexical scoping
   - Pattern matching for complex expressions
   - Built-in function and operator handling
4. **Object System**
   - Modular OOP implementation
   - Class hierarchy management
   - Method resolution and dispatch
5. **External Integration**
   - Python subprocess communication
   - File I/O operations (Command line support)
   - Sysetm interaction capabilities

3.2 Execution Flow

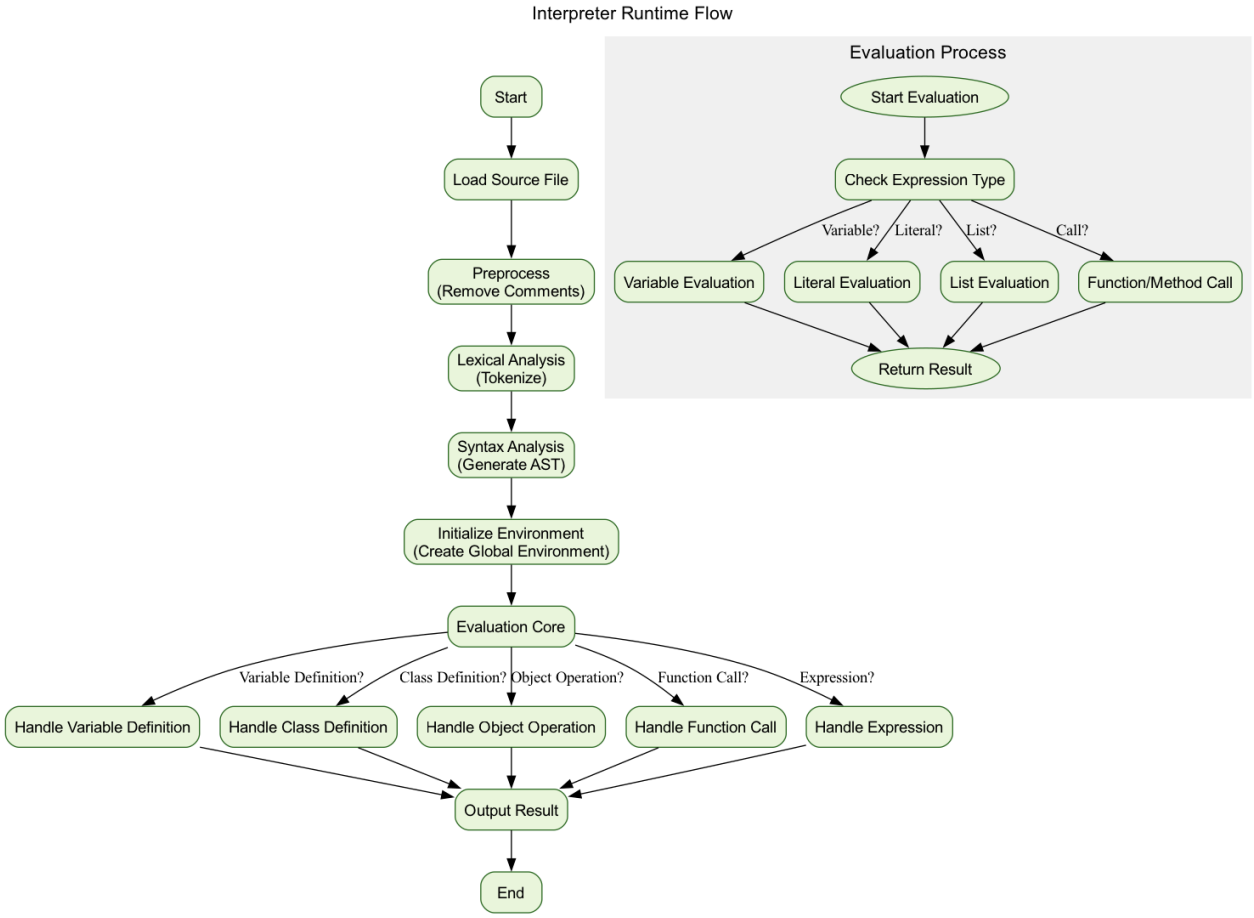

Figure 3.2.1

3.3 Techincal Implementation Details

3.3.1 Environment Chain Implementation

```
(struct env (parent bindings))

(define (env-lookup e var)
  (hash-ref (env-bindings e) var
            (lambda ()
              (if (env-parent e)
                  (env-lookup (env-parent e) var)
                  (error (~a var " 者、未聞其名也。"))))))
```

Code Snippet 3.3.1.1

Key Features:

- **Lexical Scoping**: Variable resolution follows proper lexical scope rules
- **Environment Chaining**: Parent environments are searched recursively
- **Hash Table Storage**: Efficient O(1) variable lookup within each scope
- **Error Propagation**: Meaningful error messages in Classical Chinese

3.3.2 Advanced Stringification System

```
(define (stringify val)
  (cond
    [(number? val)
     (call-python (format "python3 \"~a\" --to-chinese ~a" py-script-path val))]
    [(list? val)
     (define elements (map stringify val))
     (format "【~a】" (string-join elements "、"))]
    [(string-literal? val) (string-literal-value val)]
    [(boolean? val) (if val "陽" "陰")]
    [(instance? val)
     (format "【~a之實體】" (class-name (instance-class-ref val)))]
    [(class? val)
     (format "【~a之類】" (class-name val))]
    [else (format "~a" val)]))
```

Code Snippet 3.3.2.1

Features:

- **Type-Aware Formatting**: Different representation strategies for each data type
- **Chinese Numeral Integration**: Automatic conversion for numeric values
- **Nested Structure Support**: Recursive handling of complex data structures
- **Cultural Representation**: Boolean values as 陽/陰 (Yang/Yin)

3.3.3 Comment Processing

```
(define (remove-comments code)
  (string-join
   (map (lambda (line)
          (let ([semicolon-pos (string-index-of line ";")])
            (if semicolon-pos
                (substring line 0 semicolon-pos)
                line)))
        (string-split code "\n"))
   "\n"))
```

Code Snippet 3.3.3.1

Functionality:

- **Line-by-line Processing**: Handles comments on individual lines
- **Preserves Structure**: Maintains original code formatting
- **Efficient Implementation**: Single-pass comment removal

---

4 Installation and Setup

4.1 System Requirements

Component	Minimum Version	Recommended Version
Operating System	macOS 10.14+ / Windows 10+ / Ubuntu 18.04+	Latest stable release
Racket	8.0	8.11+
Python	3.6	3.10+
Memory	512 MB RAM	2 GB RAM
Storage	100 MB	500 MB

4.2 Installation Steps

4.2.1 Step 1: Install Dependencies

For macOS:

```
# Install Racket using Homebrew
brew install racket

# Install Python (if not already installed)
brew install python3
```

For Ubuntu/Debian:

```
# Install Racket
sudo apt update
sudo apt install racket

# Install Python
sudo apt install python3 python3-pip
```

For Windows:

1. Download Racket from https://racket-lang.org/
2. Download Python from https://python.org/
3. Ensure both are added to your system PATH

4.2.2 Step 2: Project Setup

```
# Clone or download the project
git clone <repository-url>
cd KanbunSE

# Verify installation
racket --version
python3 --version

# Test the interpreter
racket main.rkt example.kse
```

4.3 Project Structure

```
KanbunSE/
├── main.rkt                  # Primary interpreter engine
├── oo-module.rkt            # Object-oriented programming module
├── chinese_number.py        # Chinese numeral conversion utility
├── example.kse              # Basic feature demonstration
├── oo.kse                   # Advanced OOP examples
├── diagrams/                # Architecture diagrams
│   ├── structure.png
│   └── flow.png
└── README.md               # Project documentation
```

---

5 Data Types and Literals

5.1 Primitive Data Types

5.1.1 Numeric Types

KanbunSE supports both integer and floating-point numbers with comprehensive Chinese numeral representation:

Arabic	Traditional Chinese	Formal Chinese	Usage Context
0	零	零	General use
1	一	壹	Financial/Formal
10	十	拾	Standard counting
100	百	佰	Large numbers
1000	千	仟	Very large numbers
10000	萬	萬	Traditional units

Table 5.1.1.1

Examples:

三百一十四點一五　; 314.15
五千零五十　; 5050
負二十三點七　; -23.7

5.1.2 String Literals

String are enclosed in Chinese quotation marks 「」:

「你好世界」　; "Hello World"
「這是一個字符串」　; "This is a string"
「包含數字*123*的字符串」 ; "String containing numbers 123"

5.1.3 Boolean Values

Value	Chinese	Meaning
`true`	陽	Yang (positive, true)
`false`	陰	Yin (negative, false)

Table 5.1.3.1

5.1.4 List Types

KanbunSE supports two distinct list types:

Immutable Lists (Native):

- Used for parameter passing and expression grouping
- Greated implicitly in many contexts
- Read-only after creation

Mutable Lists:

(為列以 (一 二 三 四 五)) ; Create mutable list [1, 2, 3, 4, 5]

5.1.4.1 Object Types

- **Classes**
  Template definitions for object ceratino
- **Instances**
  Concrete object with state and behavior
- **Methods**
  Functions bound to class instances

---

6 Basic Syntax Reference

6.1 Variable Definition and Assignment

KanbunSE provides two syntactically equivalent methods for variable definition:

6.1.1 Method 1: Classical Declaration Style

(變量名 者 值 也)

Translation: "*Variable name* is *value.*"

Examples:

(年齡 者 二十五 也) ; age = 25
(姓名 者 「張三」 也) ; name = "Zhang San"
(是否學生 者 陽 也) ; isStudent = true

6.1.2 Method 2: Naming Style

(名 值 以 變量名)

Translation: "Name *value* as *variable name.*"

Examples:

(名 二十五 以 年齡) ; age = 25
(名 「張三」 以 姓名) ; name = "Zhang San"
(名 陽 以 是否學生) ; isStudent = true

6.2 Arithmetic Operations

All arithmetic operations follows the pattern (operator operand$_1$ 以 operand$_2$):

Operation	Syntax	Example	Result
Addition	(加 a 以 b)	(加 三 以 四)	七
Subtraction	(減 a 以 b)	(減 十 以 三)	七
Multiplication	(乘 a 以 b)	(乘 五 以 六)	三十
Division	(除 a 以 b)	(除 十 以 三)	三點三...
Modulo	(除 a 以 b 所餘)	(除 十 以 三 所餘)	一
Integer Division	(整除 a 以 b)	(整除 十 以 三)	三

Table 6.2.1

Complex Expression Example:

(加 (乘 三 以 四) 以 (除 十 以 二)) ; $(3 \times 4) + (10 \div 2) = 17$

6.3 Comparison and Logical Operations

6.3.1 Comparison Operations

Operation	Syntax	Example
Equal	(a 等於 b)	(五 等於 五) → 陽
Greater Than	(a 大於 b)	(十 大於 五) → 陽
Less Than	(a 小於 b)	(三 小於 八) → 陽

Table 6.3.1.1

6.3.2 Logical Operations

Operation	Syntax	Example
AND	(a 且 b)	(陽 且 陰) → 陰
OR	(a 或 b)	(陽 或 陰) → 陽
NOT	(非 a)	(非 陽) → 陰

Table 6.3.2.1

6.4 Control Flow Structures

6.4.1 Conditional Statements

If-Then-Else:

(若 condition 則 then-branch 若非 else-branch)

If-Then:

(若 condition 則 then-branch)

Example:

(若 (年齡 大於 十八) 則
(書 「已成年」)
若非
(書 「未成年」))

6.4.2 Pattern Matching

(卦 test-value 而 (
(case1 則 action1)
(case2 則 action2)
(case3 則 action3)
))

Example:

(卦 成績等級 而 (
(「A」 則 (書 「優秀」))
(「B」 則 (書 「良好」))
(「C」 則 (書 「及格」))
))

6.5 Loop Constructs

6.5.1 Fixed Iteration Loop

(為 body times 遍)

Example:

(為 (書 「重複執行」) 五 遍) ; Execute 5 times

6.5.2 Indexed Loop

(大衍 variable 自 start 至 end 為 body)

Example:

```
(大衍 甲 自 一 至 十 為 (
      (書 「第」 未善)
      (書 甲 未善)
      (書 「次」)
))
```

6.6 Function Definition and Invocation

6.6.1 Lambad Functions

(夫 (param1 param2 ...) body)

Examples:

```
; Simple addition function
(夫 (a b) (加 a 以 b))

; Function with multiple statements
(夫 (x) (
(名 (乘 x 以 x) 以 平方)
(書 平方)
平方
))
```

6.6.2 Function Invocation

(施 function 於 arguments)

Examples:

```
; Single argument
(施 平方函數 於 五)

; Multiple arguments
(施 加法函數 於 (三 四))

; Complex argument
(施 計算函數 於 ((加 一 以 二) (乘 三 以 四)))
```

Note: When using complex expressions as **ONE** argument, since (a b c d) will be treated as arguments a, b, c and d, user should add an extra parenthesis, i.e., ((a b c d)).

---

7 Object-Oriented Programming

7.1 Class Definition

KanbunSE supports both inheritance-based and standalone class definitions:

7.1.1 Standalone Class

(立 ClassName 具 (attribute1 attribute2 ...) 能 (
(method1 (param1 param2) method-body1)
(method2 (param1) method-body2)
...
))

7.1.2 Inherited Class

(立 ChildClass 承 ParentClass 具 (new-attributes ...) 能 (
(new-methods ...)
(overridden-methods ...)
))

7.2 Comprehensive OOP Example

```
; Base Shape class
(立 形狀 具 (名稱 顏色) 能 (
(描述 () (
 (書 「此乃一」 未善)
 (書 (此 之 顏色) 未善)
 (書 (此 之 名稱))
))
(面積 () (書 「面積計算需子類實現」))
))

; Rectangle class inheriting from Shape
(立 矩形 承 形狀 具 (長 寬) 能 (
(面積 () (乘 (此 之 長) 以 (此 之 寬)))
(周長 () (乘 二 以 (加 (此 之 長) 以 (此 之 寬))))
(是否正方形 () ((此 之 長) 等於 (此 之 寬)))
))

; Create and use instances
(名 (造 矩形 之實體) 以 我的矩形)
(令 我的矩形 之 名稱 為 「測試矩形」)
(令 我的矩形 之 顏色 為 「藍色」)
(令 我的矩形 之 長 為 五)
(令 我的矩形 之 寬 為 三)

; Method invocation
(施 (我的矩形 之 描述) 於 ())
(施 (我的矩形 之 面積) 於 ())
```

7.3 Object System Features

7.3.1 Instance Creation

(造 ClassName 之實體)

7.3.2 Attriubte Access and Modification

; Get attribute
(instance 之 attributeName)

; Set attribute
(令 instance 之 attributeName 為 value)

7.3.3 Method Invocation

(施 (instance 之 methodName) 於 arguments)

7.3.4 Self-Reference in Methods

The keyword 此(this) provides access to the current nistance within method contexts:

(立 計數器 具 (數值) 能 (
(增加 (amount) (
(令 (此 之 數值) 為 (加 (此 之 數值) 以 amount))
(此 之 數值)
))
))

---

8 Advanced Features

8.1 List Operations and Manipulation

8.1.1 Creating Mutable Lists

(為列以 (element1 element2 element3 ...))

Example:

(名 (為列以 (一 二 三 四 五)) 以 數字列表)

8.1.2 List Operations

Operation	Syntax	Description
Length	(list 之 長)	Get list length
Append	(充 list 以 value)	Add element to list
Access	(list 之 index 者)	Get element at index (1-based)

Table 8.1.2.1

Example:

; Get length
(書 (數字列表 之 長))　　; Output: 五

; Add element
(充 數字列表 以 六)　　; Append 6 to list

; Access element
(書 (數字列表 之 三 者))　　; Output: 三 (3rd element)

8.1.3 Nested Lists

KanbunSE fully supports ensted list structures:

(名 (為列以 ((一 二) (三 四) (五 六))) 以 嵌套列表)

; Access nested elements
(書 ((嵌套列表 之 一 者) 之 二 者)) ; Access first sublist, second element

8.2 Pattern Matching System

The pattern matching system provides a powerful alternative ot traditional if-else chains:

(卦 test-expression 而 (
(pattern1 則 action1)
(pattern2 則 action2)
(default-pattern 則 default-action)
))

8.3 Chinese Numeral Integration

The Chinese numeral system is deeply integrated into **KanbunSE** through the external Python utility:

8.3.1 Supported Numeral Types

1. **Traditional Numbers**
   一、二、三、四、五、六、七、八、九、十
2. **Large Unit Numbers**
   百、千、万、億、兆、京
3. **Decimal Numbers**
   點 (decimal point), 分、厘、毫、絲 (decimal places)
4. **Formal Numbers**
   壹、貳、叁、肆、伍 (used in financial contexts)

8.3.2 Automatic Conversion Examples

```
; Input: 314.159
; Output: 三百一十四點一五九

; Input: 50050
; Output: 五万零五十

; Input: 0.001
; Output: 一毫
```

---

9 Example Programs

Program	Objectives
KanbunSE/example.kse	This example demonstrates fundamental language features.
KanbunSE/oo.kse	This example includes a comprehensive demonstration of OOP capabilities.

Table 9.1

10 Error Handling

10.1 Error Categories and Messages

KanbunSE provides comprehensive error handling with messages in Classical Chinese:

10.1.1 Variable Resolution Errors

錯誤: "變數名 者、未聞其名也。"
Translation: "Variable name is unknown/unheard of."
Context: Attempting to access an undefined variable

10.1.2 Type Errors

錯誤: "此物無屬性可取"
Translation: "This object has no attributes to access."
Context: Trying to access attributes on non-object types

10.1.3 Syntax Errors

錯誤: "括未合、終無所見。"
Translation: "Brackets are unmatched, nothing seen at the end."
Context: Unbalanced parentheses in S-expressions

10.1.4 Method Resolution Errors

錯誤: "方法 methodName 不存在於此類及其父類"
Translation: "Method methodName does not exist in this class or its parent classes."
Context: Calling undefined methods on objects

10.1.5 Index Out of Bounds

錯誤: "索驥圖外、其轍亂矣。"

Translation: "Seeking outside the map, the tracks are in disorder."

Context: List index access beyond bounds

---

11 Performance Notes

Operation	Time Complexity	Space Complexity	Notes
Variable Lookup	O(n) worst case	O(1)	n = environment chain depth
List Access	O(1)	O(1)	Direct index access
Method Resolution	O(m)	O(1)	m = inheritance chain depth
Function Call	O(1)	O(k)	k = number of parameters

---

12 Conclusion

KanbunSE represents a unique fusion of classical Chinese literary tradition with modern programming language design. Through its comprehensive feature set, including object-oriented programming, functional programming constructs, and advanced data manipulation capabilities, **KanbunSE** demonstrates that programming languages can serve as bridges between cultures and epochs.

13 Appendix

13.1 Python Integration

The Python script (`KanbunSE/chinese_number.py`) is designed to be a command line tool that converts between Chinese numerals and Arabic numerals in both direction. Also, for clearity, it reports a Racket-friendly output for the Racket language to determine whether the conversion is done successfully, and push forward the process in the control flow of the *atom* section.

Key Option	Usage
`--from-chinese`	Convert Chinese numerals to floats.
`--to-chinese`	Convert float numbers to floats.
`--formal`	For formal Chinese numerals.
`--no-units`	Omit units in decimal output.

Table 14.1

「程式之美，在於文化與技術之交融」

"The beauty of programming lies in the fusion of culture and technology"