Tokenization Module Documentation

Lohit P Talavar

May 9, 2025

Contents

1 Overview

The `tokenization.py` module converts source code into a stream of tokens for the STCL language. It handles:

- Whitespace and comment skipping
- Identifier/keyword recognition
- Number/string literal parsing
- Error handling with line/column tracking

2 Token Class

2.1 Structure

```
class Token:
    def __init__(self, value, type, line, column):
        self.value = value  # Raw token value
        self.type = type    # Token category
        self.line = line      # Source line number
        self.column = column# Starting column position
```

2.2 Token Types

Type	Examples
KEYWORD	var, int, float, char
IDENTIFIER	count, temperature
INT	42, -15, 0
FLOAT	3.14, -0.5e-3
STRING	”hello”, ’A’
OPERATOR	+, -, *, /
PUNCTUATOR	(,), {, }, :, ;
COMMENT	//..., /*...*/
ERROR	Invalid tokens

3 Main Tokenization Flow

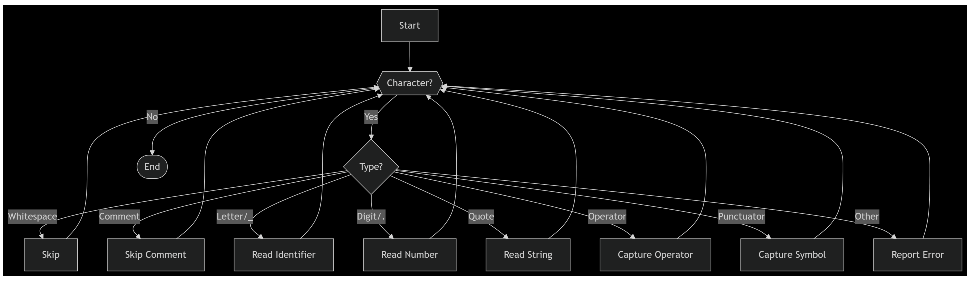

Figure 1: Tokenization process flowchart

4 Key Functions

4.1 tokenization(code: str) → list[Token]

Main entry point that processes the input string.

4.1.1 Workflow

1. Initialize position tracking (line/column)
2. Process characters until EOF
3. Dispatch to helper functions based on current char
4. Collect tokens with source positions

4.2 Helper Functions

- `advance()`: Move to next character
- `skip_whitespaces()`: Skip spaces/tabs/newlines
- `skip_comment()`: Handle // and /* */ comments
- `read_identifier()`: Capture [a-zA-Z0-9_]+
- `read_number()`: Parse int/float literals
- `read_string()`: Process quoted strings

5 Error Handling

5.1 Detected Errors

- Unterminated strings/comments
- Invalid number formats
- Unexpected characters
- Malformed operators

5.2 Error Reporting

Errors raise `SyntaxError` with format:

```
SyntaxError: <message> at line <line>:<column>
```

6 Examples

6.1 Input Code

```
var count: int(0, 100); // Simple counter
var msg: string("Hello", lower+upper);
```

6.2 Generated Tokens

```
Token('var', 'KEYWORD', line=1, column=1)
Token('count', 'IDENTIFIER', line=1, column=5)
Token(':', 'COLON', line=1, column=10)
Token('int', 'KEYWORD', line=1, column=12)
Token('(', 'LPAREN', line=1, column=15)
Token('0', 'INT', line=1, column=16)
Token(',', 'COMMA', line=1, column=17)
Token('100', 'INT', line=1, column=19)
Token(')', 'RPAREN', line=1, column=22)
Token(';', 'SEMICOLON', line=1, column=23)
Token('// Simple counter', 'COMMENT', line=1, column=25)
...
```

7 Edge Case Handling

7.1 Valid Cases

- Numbers: `.5`, `1e-5`, `123_456`
- Strings: `"Embedded "quote"`, `'multi'`
- Comments: `/* Nested /* comments */ */`

7.2 Invalid Cases

- `var 123bad: int;` (Invalid identifier)
- `var price: float(1.2.3);` (Invalid float)
- `var str: "unclosed string;` (Missing quote)