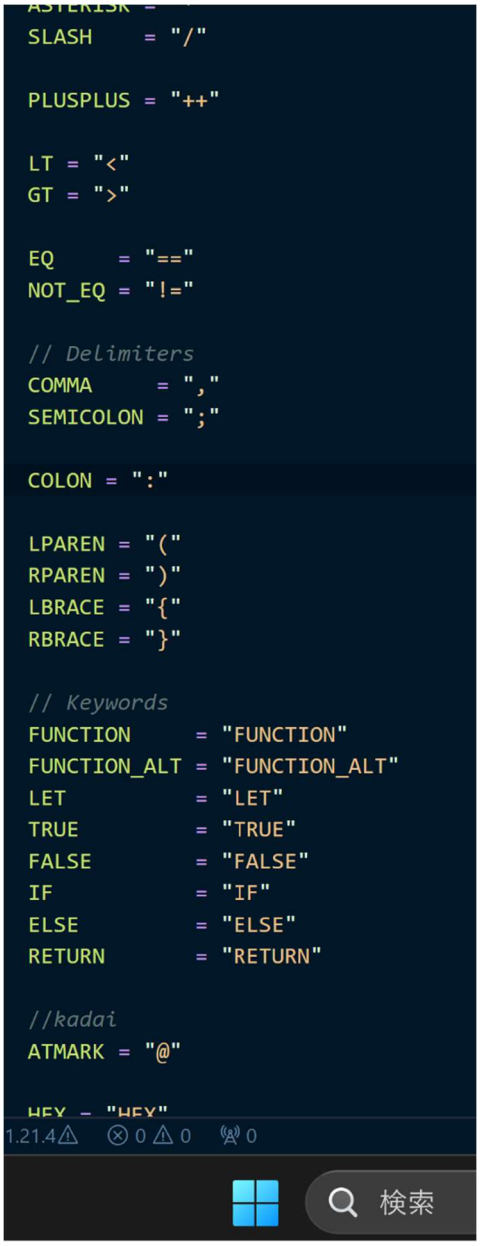
言語処理　課題２

情報科学科　２年
学習番号：２２１４０００３
佐倉仙汰郎

１．ソースコードの変更点と実行結果

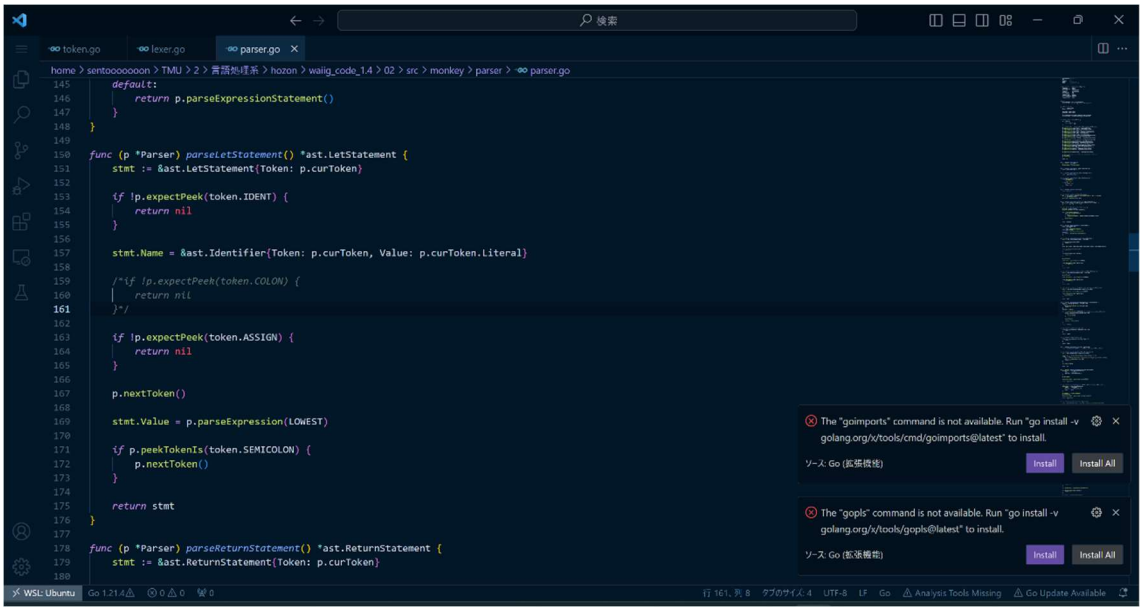
```go
func (p *Parser) parseLetStatement() *ast.LetStatement {
    stmt := &ast.LetStatement{Token: p.curToken}

    if !p.expectPeek(token.IDENT) {
        return nil
    }

    stmt.Name = &ast.Identifier{Token: p.curToken, Value: p.curToken.Literal}

    if !p.expectPeek(token.COLON) {
        return nil
    }

    if !p.expectPeek(token.ASSIGN) {
        return nil
    }

    p.nextToken()

    stmt.Value = p.parseExpression(LOWEST)

    if p.peekTokenIs(token.SEMICOLON) {
        p.nextToken()
    }

    return stmt
}
```

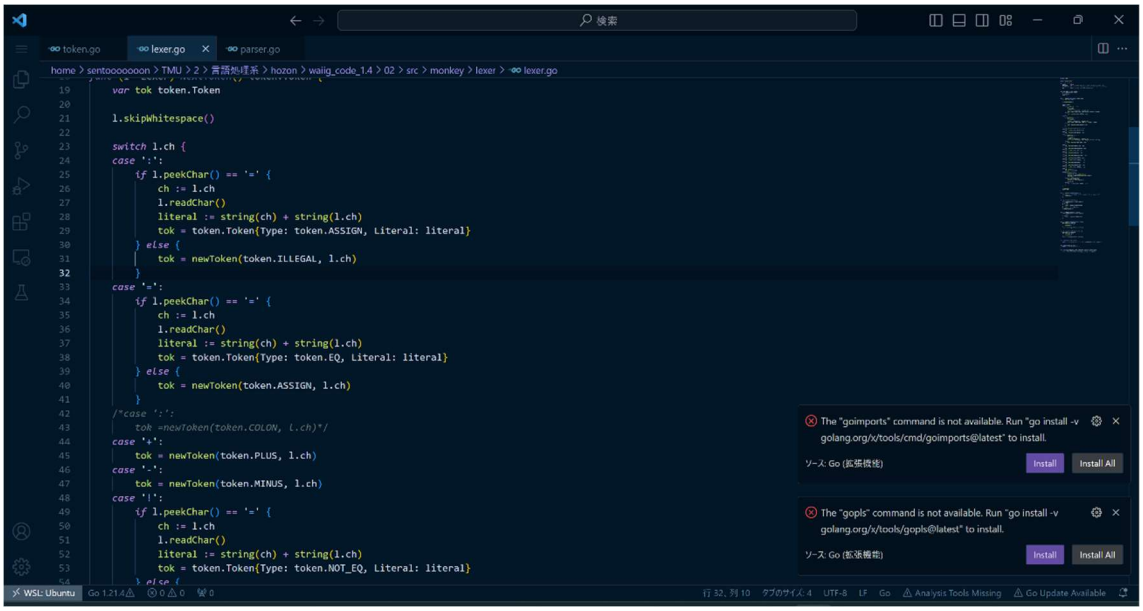
```
        tok = newToken(token.ASTERISK, l.ch)
    case '<':
        tok = newToken(token.LT, l.ch)
    case '>':
        tok = newToken(token.GT, l.ch)
    case ';':
        tok = newToken(token.SEMICOLON, l.ch)
    case ',':
        tok = newToken(token.COMMA, l.ch)
    case ':':
        tok = newToken(token.COLON, l.ch)
    case '{':
        tok = newToken(token.LBRACE, l.ch)
    case '}':
        tok = newToken(token.RBRACE, l.ch)
    case '(':
        tok = newToken(token.LPAREN, l.ch)
    case ')':
        tok = newToken(token.RPAREN, l.ch)
    case 0:
        tok.Literal = ""
        tok.Type = token.EOF
    default:
        if isLetter(l.ch) {
            tok.Literal = l.readIdentifier()
```

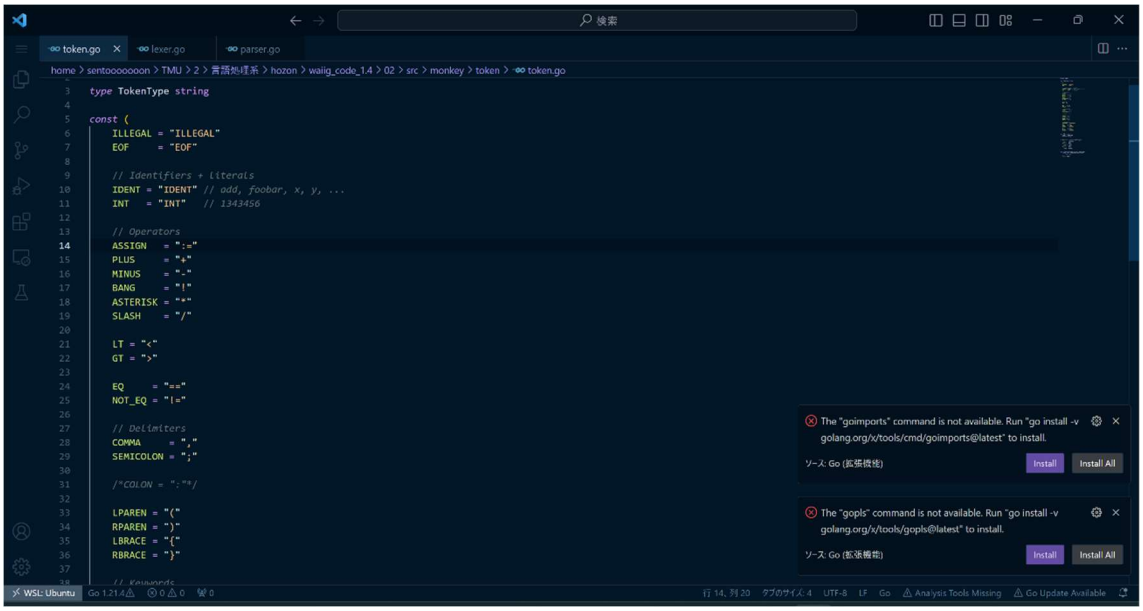

ASTERISK =
SLASH = "/"
PLUSPLUS = "++"
LT = "<"
GT = ">"
EQ = "=="
NOT_EQ = "!="
// Delimiters
COMMA = ","
SEMICOLON = ";"
COLON = ":"
LPAREN = "("
RPAREN = ")"
LBRACE = "{"
RBRACE = "}"
// Keywords
FUNCTION = "FUNCTION"
FUNCTION_ALT = "FUNCTION_ALT"
LET = "LET"
TRUE = "TRUE"
FALSE = "FALSE"
IF = "IF"
ELSE = "ELSE"
RETURN = "RETURN"
//kadai
ATMARK = "@"
HEX = "HEX"
1.21.4 ⊗ 0 ⚠ 0 0
検索

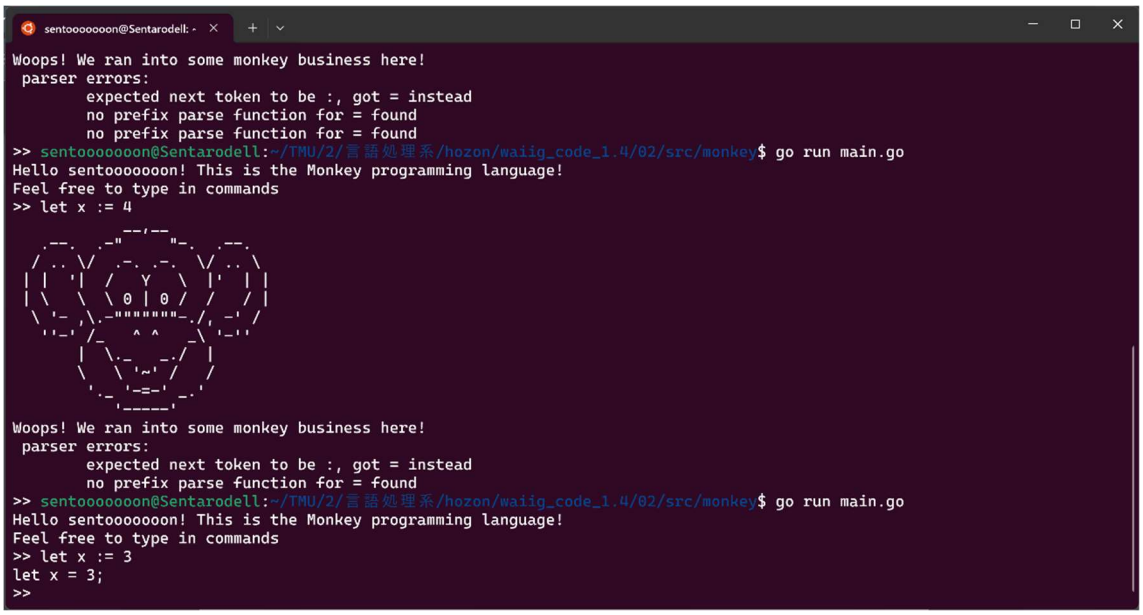
```
sentooooooon@Sentarodell:~/TMU/2/言語処理系/hozon/waiig_code_1.4/02/src/monkey$ code parser/parser.go
sentooooooon@Sentarodell:~/TMU/2/言語処理系/hozon/waiig_code_1.4/02/src/monkey$ ls
ast  lexer  main.go  parser  repl  token
sentooooooon@Sentarodell:~/TMU/2/言語処理系/hozon/waiig_code_1.4/02/src/monkey$ go run main.go
Hello sentooooooon! This is the Monkey programming language!
Feel free to type in commands
>> let x := 3
let x = 3;
>> sentooooooon@Sentarodell:~/TMU/2/言語処理系/hozon/waiig_code_1.4/02/src/monkey$ go run main.go
Hello sentooooooon! This is the Monkey programming language!
Feel free to type in commands
>> let x = 2
            __,__
   .--.  .-"     "-.  .--.
  / .. \/  .-. .-.  \/ .. \
 | |  '|  /   Y   \  |'  | |
 | \   \  \ 0 | 0 /  /   / |
  \ '- ,\.-"""""""-./, -' /
   ''-' /_   ^ ^   _\ '-''
       |  \._   _./  |
       \   \ '~' /   /
        '._ '-=-' _.'
           '-----'
Woops! We ran into some monkey business here!
 parser errors:
        expected next token to be :, got = instead
        no prefix parse function for = found
>> let x := 6
let x = 6;
>>
```

2．ソースコードの変更点と実行結果

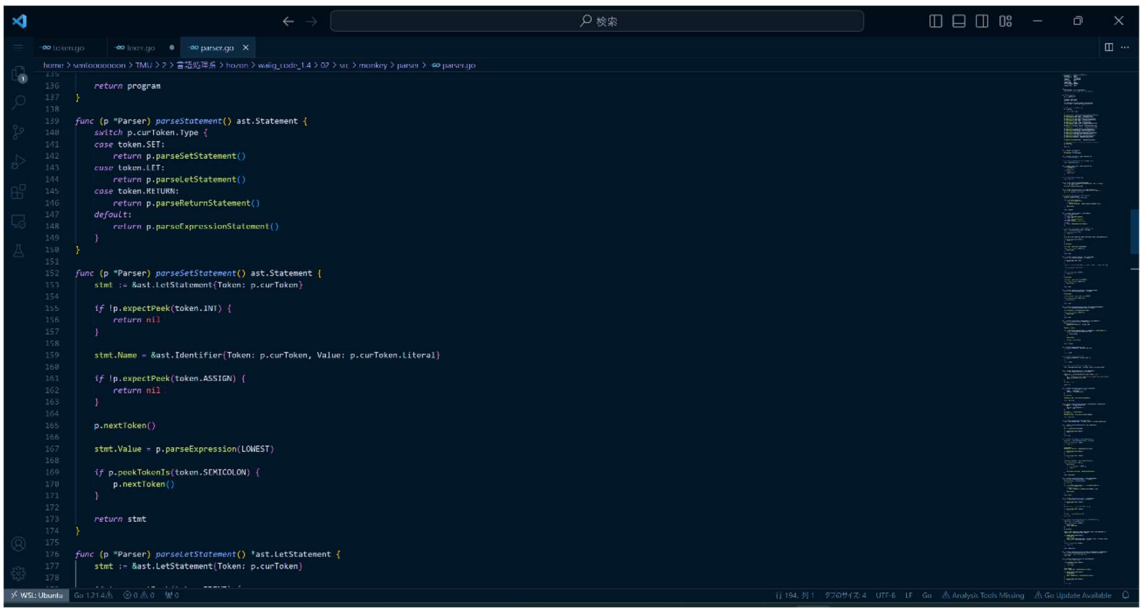


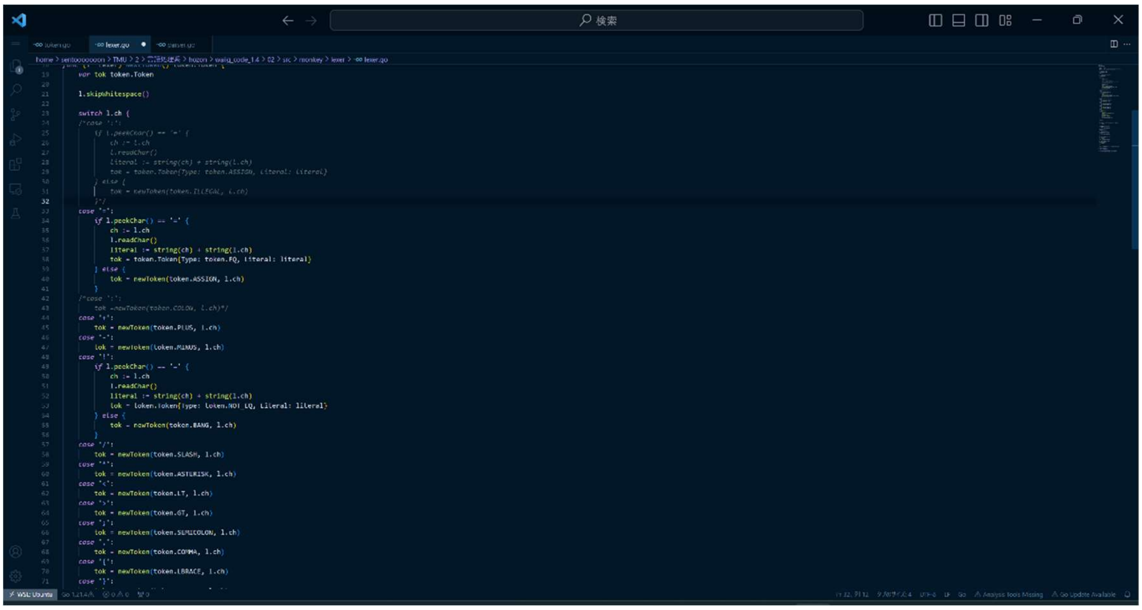

token.go
lexer.go
parser.go
home > sentooooooon > TMU > 2 > 言語処理系 > hozon > waiig_code_1.4 > 02 > src > monkey > lexer > lexer.go
var tok token.Token
l.skipWhitespace()
switch l.ch {
case ':':
if l.peekChar() == '=' {
ch := l.ch
l.readChar()
literal := string(ch) + string(l.ch)
tok = token.Token{Type: token.ASSIGN, Literal: literal}
} else {
tok = newToken(token.ILLEGAL, l.ch)
}
case '=':
if l.peekChar() == '=' {
ch := l.ch
l.readChar()
literal := string(ch) + string(l.ch)
tok = token.Token{Type: token.EQ, Literal: literal}
} else {
tok = newToken(token.ASSIGN, l.ch)
}
/*case ':':
tok =newToken(token.COLON, l.ch)*/
case '+':
tok = newToken(token.PLUS, l.ch)
case '-':
tok = newToken(token.MINUS, l.ch)
case '!':
if l.peekChar() == '=' {
ch := l.ch
l.readChar()
literal := string(ch) + string(l.ch)
tok = token.Token{Type: token.NOT_EQ, Literal: literal}
The "goimports" command is not available. Run "go install -v golang.org/x/tools/cmd/goimports@latest" to install.
ソース: Go (拡張機能)
Install
Install All
The "gopls" command is not available. Run "go install -v golang.org/x/tools/gopls@latest" to install.
ソース: Go (拡張機能)
Install
Install All
WSL: Ubuntu
Go 1.21.4
行 32, 列 10
タブのサイズ: 4
UTF-8
LF
Go
Analysis Tools Missing
Go Update Available


token.go
lexer.go
parser.go
home > sentooooooon > TMU > 2 > 言語処理系 > hozon > waiig_code_1.4 > 02 > src > monkey > token > token.go
type TokenType string
const (
ILLEGAL = "ILLEGAL"
EOF = "EOF"
// Identifiers + literals
IDENT = "IDENT" // add, foobar, x, y, ...
INT = "INT" // 1343456
// Operators
ASSIGN = ":="
PLUS = "+"
MINUS = "-"
BANG = "!"
ASTERISK = "*"
SLASH = "/"
LT = "<"
GT = ">"
EQ = "=="
NOT_EQ = "!="
// Delimiters
COMMA = ","
SEMICOLON = ";"
/*COLON = ":"*/
LPAREN = "("
RPAREN = ")"
LBRACE = "{"
RBRACE = "}"
The "goimports" command is not available. Run "go install -v golang.org/x/tools/cmd/goimports@latest" to install.
ソース: Go (拡張機能)
Install
Install All
The "gopls" command is not available. Run "go install -v golang.org/x/tools/gopls@latest" to install.
ソース: Go (拡張機能)
Install
Install All
WSL: Ubuntu
Go 1.21.4
行 14, 列 20
タブのサイズ: 4
UTF-8
LF
Go
Analysis Tools Missing
Go Update Available

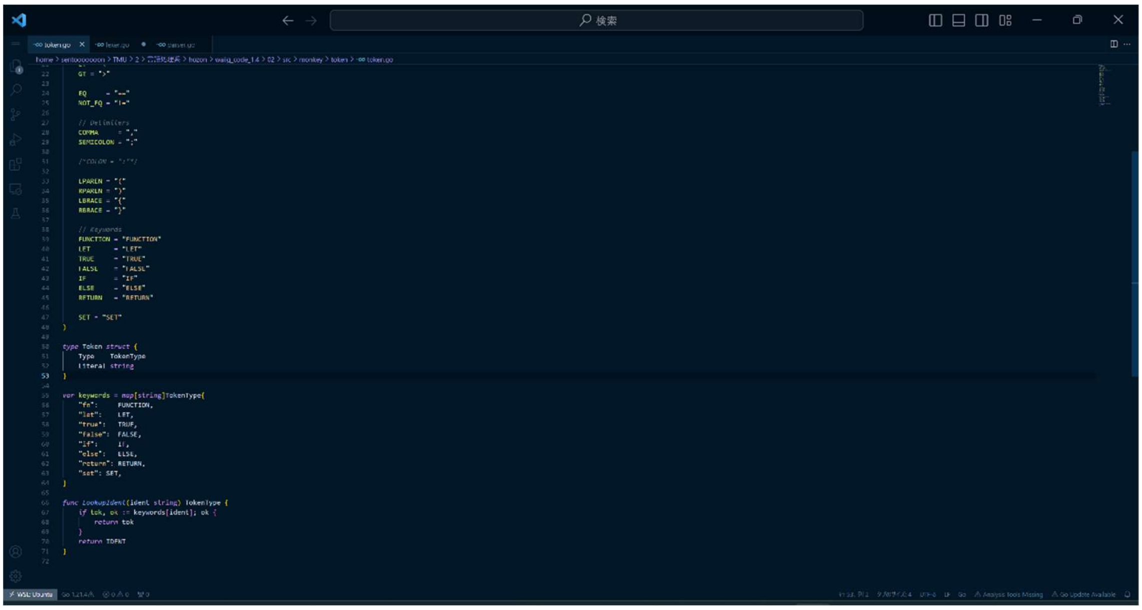


3．ソースコードの変更点と実行結果



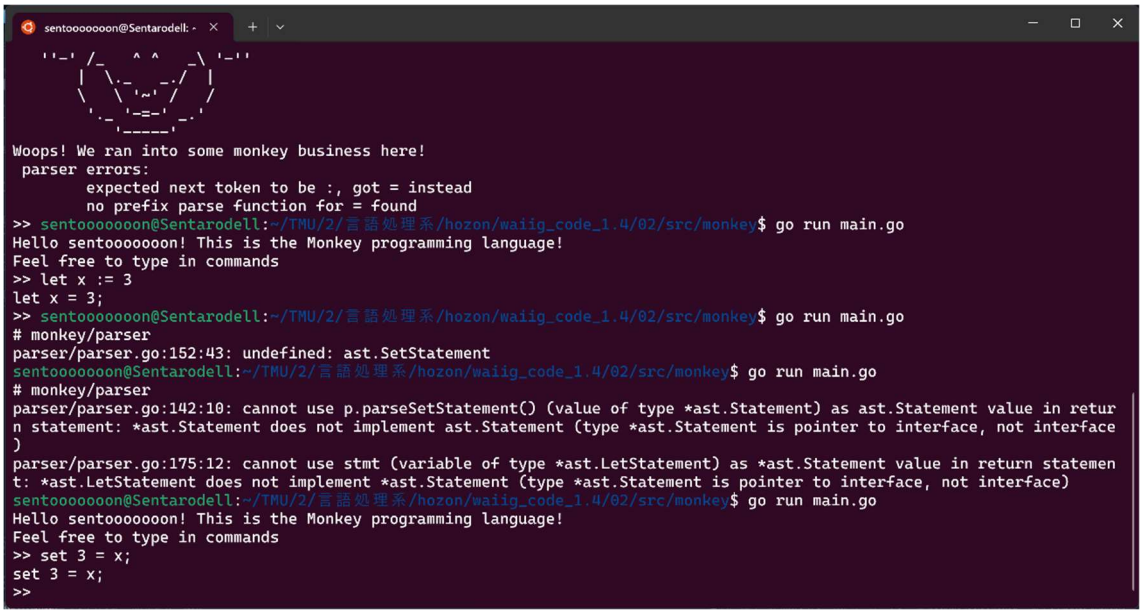


token.go
lexer.go
parser.go
var tok token.Token
l.skipWhitespace()
switch l.ch {
/*case ':':
if l.peekChar() == '=' {
ch := l.ch
l.readChar()
literal := string(ch) + string(l.ch)
tok = token.Token{Type: token.ASSIGN, Literal: literal}
} else {
tok = newToken(token.ILLEGAL, l.ch)
}*/
case '=':
if l.peekChar() == '=' {
ch := l.ch
l.readChar()
literal := string(ch) + string(l.ch)
tok = token.Token{Type: token.EQ, Literal: literal}
} else {
tok = newToken(token.ASSIGN, l.ch)
}
/*case ':':
tok =newToken(token.COLON, l.ch)*/
case '+':
tok = newToken(token.PLUS, l.ch)
case '-':
tok = newToken(token.MINUS, l.ch)
case '!':
if l.peekChar() == '=' {
ch := l.ch
l.readChar()
literal := string(ch) + string(l.ch)
tok = token.Token{Type: token.NOT_EQ, Literal: literal}
} else {
tok = newToken(token.BANG, l.ch)
}
case '/':
tok = newToken(token.SLASH, l.ch)
case '*':
tok = newToken(token.ASTERISK, l.ch)
case '<':
tok = newToken(token.LT, l.ch)
case '>':
tok = newToken(token.GT, l.ch)
case ';':
tok = newToken(token.SEMICOLON, l.ch)
case ',':
tok = newToken(token.COMMA, l.ch)
case '{':
tok = newToken(token.LBRACE, l.ch)
case '}':
WSL: Ubuntu

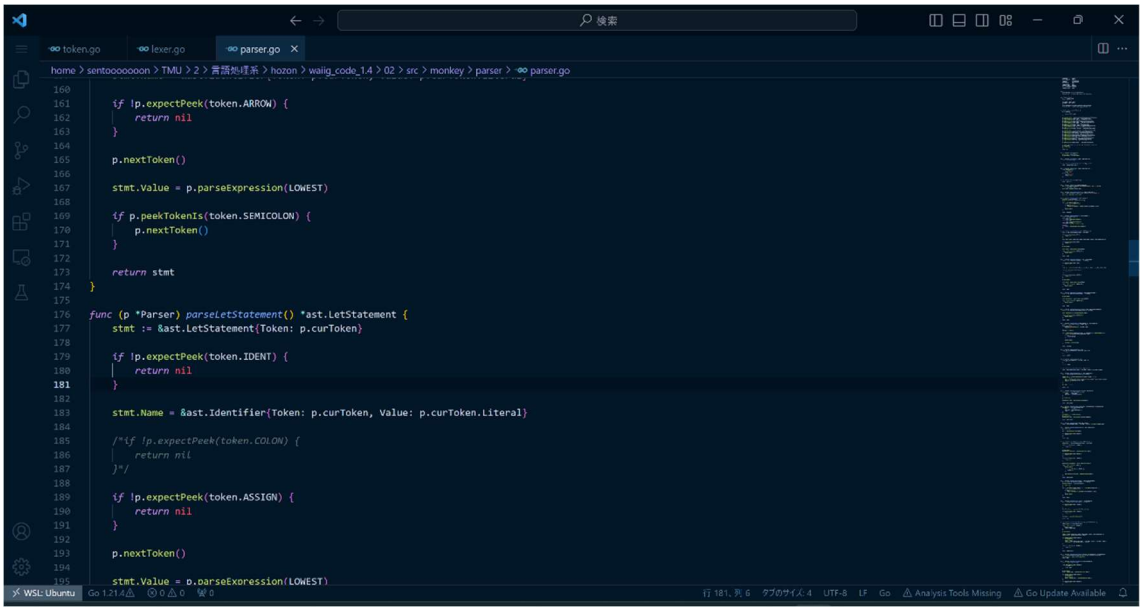


token.go
lexer.go
parser.go
GT = ">"
EQ = "=="
NOT_EQ = "!="
// Delimiters
COMMA = ","
SEMICOLON = ";"
/*COLON = ":"*/
LPAREN = "("
RPAREN = ")"
LBRACE = "{"
RBRACE = "}"
// Keywords
FUNCTION = "FUNCTION"
LET = "LET"
TRUE = "TRUE"
FALSE = "FALSE"
IF = "IF"
ELSE = "ELSE"
RETURN = "RETURN"
SET = "SET"
)
type Token struct {
Type TokenType
Literal string
}
var keywords = map[string]TokenType{
"fn": FUNCTION,
"let": LET,
"true": TRUE,
"false": FALSE,
"if": IF,
"else": ELSE,
"return": RETURN,
"set": SET,
}
func LookupIdent(ident string) TokenType {
if tok, ok := keywords[ident]; ok {
return tok
}
return IDENT
}
WSL: Ubuntu

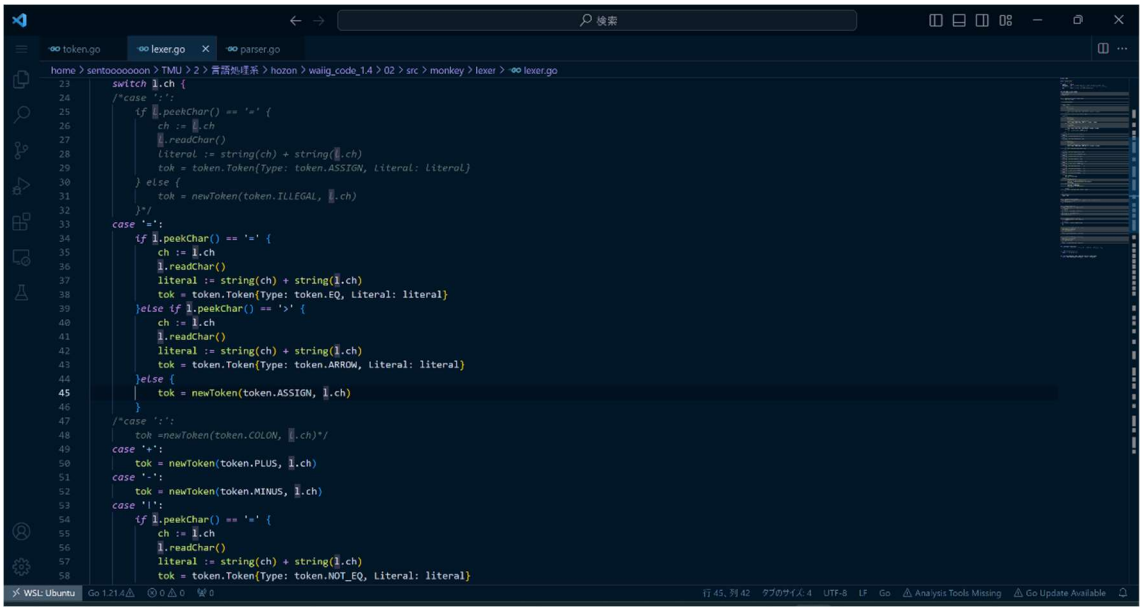
4 .

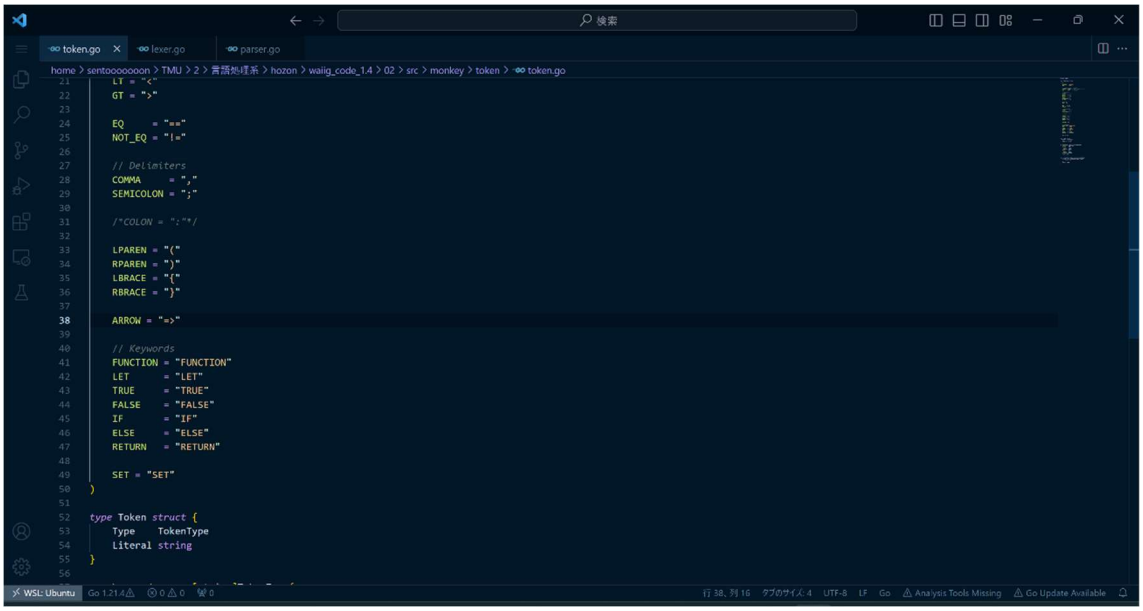


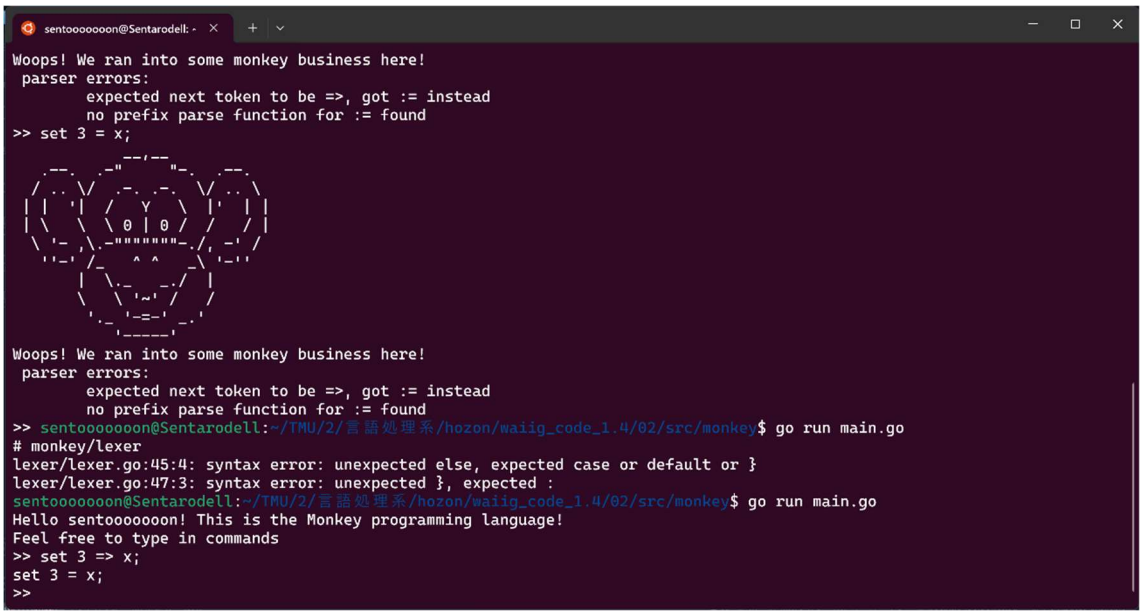
```
sentooooooon@Sentarodell: ~
   ''-' /_   ^ ^   _\ '-''
       |  \._   _./  |
       \   \ '~' /   /
        '-. '-=-' .-'
          '-----'
Woops! We ran into some monkey business here!
 parser errors:
        expected next token to be :, got = instead
        no prefix parse function for = found
>> sentooooooon@Sentarodell:~/TMU/2/言語処理系/hozon/waiig_code_1.4/02/src/monkey$ go run main.go
Hello sentooooooon! This is the Monkey programming language!
Feel free to type in commands
>> let x := 3
let x = 3;
>> sentooooooon@Sentarodell:~/TMU/2/言語処理系/hozon/waiig_code_1.4/02/src/monkey$ go run main.go
# monkey/parser
parser/parser.go:152:43: undefined: ast.SetStatement
sentooooooon@Sentarodell:~/TMU/2/言語処理系/hozon/waiig_code_1.4/02/src/monkey$ go run main.go
# monkey/parser
parser/parser.go:142:10: cannot use p.parseSetStatement() (value of type *ast.Statement) as ast.Statement value in retur
n statement: *ast.Statement does not implement ast.Statement (type *ast.Statement is pointer to interface, not interface
)
parser/parser.go:175:12: cannot use stmt (variable of type *ast.LetStatement) as *ast.Statement value in return statemen
t: *ast.LetStatement does not implement *ast.Statement (type *ast.Statement is pointer to interface, not interface)
sentooooooon@Sentarodell:~/TMU/2/言語処理系/hozon/waiig_code_1.4/02/src/monkey$ go run main.go
Hello sentooooooon! This is the Monkey programming language!
Feel free to type in commands
>> set 3 = x;
set 3 = x;
>>
```



```
token.go  lexer.go  parser.go
home > sentooooooon > TMU > 2 > 言語処理系 > hozon > waiig_code_1.4 > 02 > src > monkey > parser > parser.go
160
161     if !p.expectPeek(token.ARROW) {
162         return nil
163     }
164
165     p.nextToken()
166
167     stmt.Value = p.parseExpression(LOWEST)
168
169     if p.peekTokenIs(token.SEMICOLON) {
170         p.nextToken()
171     }
172
173     return stmt
174 }
175
176 func (p *Parser) parseLetStatement() *ast.LetStatement {
177     stmt := &ast.LetStatement{Token: p.curToken}
178
179     if !p.expectPeek(token.IDENT) {
180         return nil
181     }
182
183     stmt.Name = &ast.Identifier{Token: p.curToken, Value: p.curToken.Literal}
184
185     /*if !p.expectPeek(token.COLON) {
186         return nil
187     }*/
188
189     if !p.expectPeek(token.ASSIGN) {
190         return nil
191     }
192
193     p.nextToken()
194
195     stmt.Value = p.parseExpression(LOWEST)
WSL: Ubuntu  Go 1.21.4  行 181, 列 6  タブのサイズ: 4  UTF-8  LF  Go  Analysis Tools Missing  Go Update Available
```


検索
token.go
lexer.go
parser.go
home > sentooooooon > TMU > 2 > 言語処理系 > hozon > waiig_code_1.4 > 02 > src > monkey > lexer > lexer.go
switch l.ch {
/*case ':':
if l.peekChar() == '=' {
ch := l.ch
l.readChar()
literal := string(ch) + string(l.ch)
tok = token.Token{Type: token.ASSIGN, Literal: literal}
} else {
tok = newToken(token.ILLEGAL, l.ch)
}*/
case '=':
if l.peekChar() == '=' {
ch := l.ch
l.readChar()
literal := string(ch) + string(l.ch)
tok = token.Token{Type: token.EQ, Literal: literal}
}else if l.peekChar() == '>' {
ch := l.ch
l.readChar()
literal := string(ch) + string(l.ch)
tok = token.Token{Type: token.ARROW, Literal: literal}
}else {
tok = newToken(token.ASSIGN, l.ch)
}
/*case ':':
tok =newToken(token.COLON, l.ch)*/
case '+':
tok = newToken(token.PLUS, l.ch)
case '-':
tok = newToken(token.MINUS, l.ch)
case '!':
if l.peekChar() == '=' {
ch := l.ch
l.readChar()
literal := string(ch) + string(l.ch)
tok = token.Token{Type: token.NOT_EQ, Literal: literal}
WSL: Ubuntu
Go 1.21.4
行 45, 列 42
タブのサイズ: 4
UTF-8
LF
Go
Analysis Tools Missing
Go Update Available


検索
token.go
lexer.go
parser.go
home > sentooooooon > TMU > 2 > 言語処理系 > hozon > waiig_code_1.4 > 02 > src > monkey > token > token.go
LT = "<"
GT = ">"
EQ = "=="
NOT_EQ = "!="
// Delimiters
COMMA = ","
SEMICOLON = ";"
/*COLON = ":"*/
LPAREN = "("
RPAREN = ")"
LBRACE = "{"
RBRACE = "}"
ARROW = "=>"
// Keywords
FUNCTION = "FUNCTION"
LET = "LET"
TRUE = "TRUE"
FALSE = "FALSE"
IF = "IF"
ELSE = "ELSE"
RETURN = "RETURN"
SET = "SET"
type Token struct {
Type TokenType
Literal string
WSL: Ubuntu
Go 1.21.4
行 38, 列 16
タブのサイズ: 4
UTF-8
LF
Go
Analysis Tools Missing
Go Update Available


sentooooooon@Sentarodell: ~
Woops! We ran into some monkey business here!
parser errors:
expected next token to be =>, got := instead
no prefix parse function for := found
>> set 3 = x;
Woops! We ran into some monkey business here!
parser errors:
expected next token to be =>, got := instead
no prefix parse function for := found
>> sentooooooon@Sentarodell:~/TMU/2/言語処理系/hozon/waiig_code_1.4/02/src/monkey$ go run main.go
monkey/lexer
lexer/lexer.go:45:4: syntax error: unexpected else, expected case or default or }
lexer/lexer.go:47:3: syntax error: unexpected }, expected :
sentooooooon@Sentarodell:~/TMU/2/言語処理系/hozon/waiig_code_1.4/02/src/monkey$ go run main.go
Hello sentooooooon! This is the Monkey programming language!
Feel free to type in commands
>> set 3 => x;
set 3 = x;
>>