CS335 - Course Project

Group 15 - Milestone 2

Source : Go

Implementation Language : Python

Target Language: x86

Repo Link

Structure of Directory

```
.
├── docs
│   └── Milestone1_group15_part.pdf
├── go.sh
├── README.md
├── src
│   ├── lexer.py
│   └── README.md
└── tests
    ├── test1.go
    ├── test2.go
    ├── test3.go
    ├── test4.go
    └── test5.go
```

Output of `git show lexer`

```
commit f4e86d4aa7ccda494000526be0845a40902ee5d7
Author: adip1343 <aadhip43@gmail.com>
Date:   Tue Feb 1 20:54:12 2022 +0530

    Added Lexer Code

diff --git a/README.md b/README.md
index 34d47ad..b0676d3 100644
--- a/README.md
+++ b/README.md
@@ -1,2 +1,18 @@
 # Compiler-Design-Group-15
 Designing Compiler of Go in Python
+
+# How to run
+
+Install PLY package by using
+```pip install ply```
+
```

```
+Give executing permission
+```chmod u+x go.sh```
+
+To run lexer on test case use
+```./go.sh #n```
+
+### references
+https://gobyexample.com/
+https://ply.readthedocs.io/en/latest/ply.html
+https://go.dev/ref/spec
diff --git a/go.sh b/go.sh
new file mode 100755
index 0000000..bbe04e5
--- /dev/null
+++ b/go.sh
@@ -0,0 +1,2 @@
+FILE="./tests/test$1.go"
+python3 src/lexer.py $FILE
\ No newline at end of file
diff --git a/src/README.md b/src/README.md
new file mode 100644
index 0000000..e69de29
diff --git a/src/lexer.py b/src/lexer.py
new file mode 100644
index 0000000..a496dad
--- /dev/null
+++ b/src/lexer.py
@@ -0,0 +1,218 @@
+import ply.lex as lex
+import sys
+from ply.lex import TOKEN
+
+reserved = {
+    'var'      : 'VAR',
+    'break'    : 'BREAK',
+    'continue' : 'CONTINUE',
+    'return'   : 'RETURN',
+    'nil'      : 'NIL',
+    'default'  : 'DEFAULT',
+    'func'     : 'FUNC',
+    'case'     : 'CASE',
+    'go'       : 'GO',
+    'struct'   : 'STRUCT',
+    'else'     : 'ELSE',
+    'package'  : 'PACKAGE',
+    'switch'   : 'SWITCH',
+    'const'    : 'CONST',
+    'if'       : 'IF',
+    'type'     : 'TYPE',
+    'for'      : 'FOR',
+    'import'   : 'IMPORT',
+    'struct'   : 'STRUCT',
+    'make' : 'MAKE'
+}
+
+dec_digits = r'[0-9]+'
+hex_digits = r'[0-9a-fA-F]+'
+number_lit = rf'((0[xX]{hex_digits})|{dec_digits})'
```

```
+float_mantissa = rf'(({dec_digits}(\.){dec_digits})|({dec_digits}(\.))|((\.)
{dec_digits}))'
+float_exp = rf'([Ee][+-]?{dec_digits})'
+float_lit = rf'{float_mantissa}{float_exp}?'
+
+class Lexer:
+
+    tokens = list(reserved.values()) + [
+        'PLUS_EQ',
+        'MINUS_EQ',
+        'STAR_EQ',
+        'DIVIDE_EQ',
+        'MODULO_EQ',
+        'AMP_EQ',
+        'OR_EQ',
+        'CARET_EQ',
+        'EQ',
+        'EQ_EQ',
+        'NOT',
+        'NOT_EQ',
+        'LT_EQ',
+        'GT_EQ',
+        'LT',
+        'GT',
+        'AMP_AMP',
+        'OR_OR',
+        'PLUS_PLUS',
+        'MINUS_MINUS',
+        'LSQUARE',
+        'RSQUARE',
+        'LROUND',
+        'RROUND',
+        'LCURLY',
+        'RCURLY',
+        'COMMA',
+        'DOT',
+        'SEMICOLON',
+        'COLON',
+        'SINGLE_QUOTES',
+        'DOUBLE_QUOTES',
+        'INT_LIT',
+        'FLOAT_LIT',
+        'STRING_LIT',
+        'BOOL_LIT',
+        'NEWLINE',
+        'IDENTIFIER',
+        'DATA_TYPE',
+        'PLUS',
+        'MINUS',
+        'STAR',
+        'DIVIDE',
+        'MODULO',
+        'AMP',
+        'OR',
+        'CARET',
+        'AND_NOT',
+        'LSHIFT',
+        'RSHIFT',
```

```
+        'ASSIGN'
+    ]
+
+    t_PLUS_PLUS = r'(\+\+)'
+    t_MINUS_MINUS = r'(--)'
+    t_PLUS_EQ = r'(\+=)'
+    t_MINUS_EQ = r'(-=)'
+    t_STAR_EQ = r'(\*=)'
+    t_DIVIDE_EQ = r'/='
+    t_MODULO_EQ = r'(%=)'
+    t_AMP_EQ = r'(&=)'
+    t_OR_EQ = r'(\|=)'
+    t_CARET_EQ = r'(\^=)'
+    t_AMP_AMP = r'(&&)'
+    t_OR_OR = r'(\|\|)'
+    t_LSHIFT = r'(<<)'
+    t_RSHIFT = r'(>>)'
+    t_EQ_EQ = r'(==)'
+    t_NOT_EQ = r'(!=)'
+    t_ASSIGN = r'(\:=)'
+    t_NOT = r'(!)'
+    t_LT_EQ = r'(<=)'
+    t_GT_EQ = r'(>=)'
+    t_LSQUARE = r'(\[)'
+    t_RSQUARE = r'(\])'
+    t_LROUND = r'(\()'
+    t_RROUND = r'(\))'
+    t_LCURLY = r'(\{)'
+    t_RCURLY = r'(\})'
+    t_COMMA = r'(\,)'
+    t_DOT = r'(\.)'
+    t_SEMICOLON = r'(\;)'
+    t_COLON = r'(\:)'
+    t_DOUBLE_QUOTES = r'(\")'
+    t_SINGLE_QUOTES = r'(\')'
+    t_PLUS = r'(\+)'
+    t_MINUS = r'(-)'
+    t_STAR = r'(\*)'
+    t_DIVIDE = r'(\/)'
+    t_MODULO = r'(\%)'
+    t_LT = r'(<)'
+    t_GT = r'(>)'
+    t_EQ = r'(=)'
+    t_AMP = r'(\&)'
+    t_OR = r'\|'
+    t_CARET = r'\^'
+    t_AND_NOT = r'&^'
+    t_ignore = " \t"
+
+    def __init__(self):
+        self.last_newline = -1
+        self.line_no = 1
+
+    def t_DATA_TYPE(self,t):
+        r'((uint8)|(uint16)|(uint32)|(uint64)|(int8)|(int16)|(int32)|(int64)|
(float32)|(float64)|(byte)|(rune)|(bool)|(int)|(uint)|(string))'
+        return t
+
```

```
+    def t_BOOL_LIT(self, t):
+        r'((true)|(false))'
+        t.value = 1 if t.value == "true" else "false"
+        return t
+
+    @TOKEN(float_lit)
+    def t_FLOAT_LIT(self, t):
+        t.value = float(eval(t.value))
+        return t
+
+    @TOKEN(number_lit)
+    def t_INT_LIT(self, t):
+        t.value = int(eval(t.value))
+        return t
+
+    def t_STRING_LIT(self, t):
+        r'\"[^\"]*\"'
+        cnt = t.value.count('\n')
+        if cnt != 0 :
+            print("[ERROR] String shouldn't span multiple lines.. Line:",
self.line_no,"Col:",t.lexpos - self.last_newline)
+            self.line_no += cnt
+            self.last_newline = t.lexpos + t.value.rfind('\n')
+            exit(0)
+        else :
+            return t
+
+    def t_SINGLE_LINE_COMMENT(self, t):
+        r'//[^\n]*\n'
+        self.line_no += 1
+        self.last_newline = t.lexpos + len(t.value) - 1
+        pass
+
+    def t_MULTI_LINE_COMMENT(self, t):
+        r'/\*(.|\n)*?\*/'
+        cnt = t.value.count('\n')
+        if cnt != 0 :
+            self.line_no += cnt
+            self.last_newline = t.lexpos + t.value.rfind('\n')
+
+    def t_newline(self, t):
+        r'\n'
+        self.line_no += 1
+        self.last_newline = t.lexpos
+
+    def t_IDENTIFIER(self, t):
+        r'[a-zA-Z_][a-zA-Z_0-9]*'
+        t.type = reserved.get(t.value,'IDENTIFIER')
+        return t
+
+    def t_error(self, t):
+        print(f"ERROR...{t.lexpos}")
+        t.lexer.skip(1)
+        pass
+
+    def build(self):
+        self.lexer = lex.lex(object=self)
+
```

```
+    def input(self, data):
+        self.lexer.input(data)
+        data = [["Token", "Lexeme", "Line#", "Column#"]]
+        while True :
+            tok = self.lexer.token()
+            if not tok :
+                break
+            data.append([tok.type, tok.value, self.line_no, tok.lexpos -
self.last_newline])
+        for row in data:
+            print("{: <15} {: <15} {: <8} {: >8}".format(*row))
+
+if __name__ == "__main__" :
+    file = open(sys.argv[1], 'r')
+    data = file.read()
+    lexer = Lexer()
+    lexer.build()
+    lexer.input(data)
\ No newline at end of file
diff --git a/tests/test1.go b/tests/test1.go
new file mode 100644
index 0000000..1e801cf
--- /dev/null
+++ b/tests/test1.go
@@ -0,0 +1,24 @@
+package main
+
+import "fmt"
+
+func main() {
+
+    if 7%2 == 0 {
+        fmt.Println("7 is even")
+    } else {
+        fmt.Println("7 is odd")
+    }
+
+    if 8%4 == 0 {
+        fmt.Println("8 is divisible by 4")
+    }
+
+    if num := 9; num < 0 {
+        fmt.Println(num, "is negative")
+    } else if num < 10 {
+        fmt.Println(num, "has 1 digit")
+    } else {
+        fmt.Println(num, "has multiple digits")
+    }
+}
\ No newline at end of file
diff --git a/tests/test2.go b/tests/test2.go
new file mode 100644
index 0000000..6211f4e
--- /dev/null
+++ b/tests/test2.go
@@ -0,0 +1,26 @@
+package main
+
```

```
+import "fmt"
+
+func fact(n int) int {
+    if n == 0 {
+        return 1
+    }
+    return n * fact(n-1)
+}
+
+func main() {
+    fmt.Println(fact(7))
+
+    var fib func(n int) int
+
+    fib = func(n int) int {
+        if n < 2 {
+            return n
+        }
+
+        return fib(n-1) + fib(n-2)
+    }
+
+    fmt.Println(fib(7))
+}
\ No newline at end of file
diff --git a/tests/test3.go b/tests/test3.go
new file mode 100644
index 0000000..97f18ad
--- /dev/null
+++ b/tests/test3.go
@@ -0,0 +1,37 @@
+package main
+
+import "fmt"
+
+type person struct {
+    name string
+    age  int
+}
+
+func newPerson(name string) *person {
+
+    p := person{name: name}
+    p.age = 42
+    return &p
+}
+
+func main() {
+
+    fmt.Println(person{"Bob", 20})
+
+    fmt.Println(person{name: "Alice", age: 30})
+
+    fmt.Println(person{name: "Fred"})
+
+    fmt.Println(&person{name: "Ann", age: 40})
+
+    fmt.Println(newPerson("Jon"))
```

```
+
+    s := person{name: "Sean", age: 50}
+    fmt.Println(s.name)
+
+    sp := &s
+    fmt.Println(sp.age)
+
+    sp.age = 51
+    fmt.Println(sp.age)
+}
\ No newline at end of file
diff --git a/tests/test4.go b/tests/test4.go
new file mode 100644
index 0000000..8aaff98
--- /dev/null
+++ b/tests/test4.go
@@ -0,0 +1,26 @@
+package main
+
+import "fmt"
+
+type rect struct {
+    width, height int
+}
+
+func (r *rect) area() int {
+    return r.width * r.height
+}
+
+func (r rect) perim() int {
+    return 2*r.width + 2*r.height
+}
+
+func main() {
+    r := rect{width: 10, height: 5}
+
+    fmt.Println("area: ", r.area())
+    fmt.Println("perim:", r.perim())
+
+    rp := &r
+    fmt.Println("area: ", rp.area())
+    fmt.Println("perim:", rp.perim())
+}
\ No newline at end of file
diff --git a/tests/test5.go b/tests/test5.go
new file mode 100644
index 0000000..60ccfd2
--- /dev/null
+++ b/tests/test5.go
@@ -0,0 +1,26 @@
+package main
+
+import "fmt"
+
+func main() {
+
+    var a [5]int
+    fmt.Println("emp:", a)
```

```
+
+    a[4] = 100
+    fmt.Println("set:", a)
+    fmt.Println("get:", a[4])
+
+    fmt.Println("len:", len(a))
+
+    b := [5]int{1, 2, 3, 4, 5}
+    fmt.Println("dcl:", b)
+
+    var twoD [2][3]int
+    for i := 0; i < 2; i++ {
+        for j := 0; j < 3; j++ {
+            twoD[i][j] = i + j
+        }
+    }
+    fmt.Println("2d: ", twoD)
+}
\ No newline at end of file
```