Github: https://github.com/pukoarmin/Formal-Langauges-and-Compiler-Design.git

Parser -> Documentation:

Language used: Python

=== RDConfing ===
grammar: Grammar => Contains the grammar used for parsing
state_of_parser: String => Contains the current state of the parser
position_of_current_symbol: Integer => Contains the current position of the symbol in input sequence
working_stack: List => Represents the working stack (contains the way the parse is built)
input_stack: List => Represents the input stack, part of the tree to be built
---
expand(): => For a nonterminal head of input stack it expands it according to the rule
advance(): => For a terminal head of input stack that equals the currenty symbol from input, i is incremented and alpha stack advances with the top of beta stack
momentary_insuccess(): => For a terminal head of input stack that is diferent from the current symbol from input, parser state is set to back state ('b')
back(): => For a terminal head of working stack, i is decremented and alpha stack advances
another_try(): => For a nonterminal head of working stack, either the program goes into error state ('e') or normal state ('q')
success(): => Sets the final state ('f') corresponding to success w ∈ L(G)

=== RDParser ===
+ grammar: contains the grammar used for parsing
---
parse(): ParserOutput => Parses the file and prints "sequence accepted" if everything is alright and error otherwise

=== ParserOutput ===
+ tree: list => The tree formed formed by the parser
+ grammar: Grammar => Contains the actual grammar
+ production_string: String => the production string

- get_tree(grammar, production_string): list => forms the tree from the input
+ plot_parse_tree(filename): generates the representation of the graph

Sample input:
g1.txt

S
S A B
a b epsilon
S -> epsilon | a B | b A
A -> a | a B

```
B -> b | b

seq.txt
a
b
a
b
a

g2.txt
program
BEGIN END term program cmpdstmt decllist declaration type type1 arraydecl stmt
stmtlist simplstmt assignstmt expression iostmt structstmt ifstmt forstmt condition
RELATION
IDENTIFIER Boolean Integer String List < type1 > = + - * / ( ) [ ] , read print
CONSTANT : { } if else for < <= == != >= > ; write
program -> decllist cmpdstmt
decllist -> declaration | declaration decllist
declaration -> type IDENTIFIER
type  -> type1 | arraydecl
type1 -> Boolean | Integer | String
arraydecl -> List < type1 >
cmpdstmt -> BEGIN stmtlist END
stmtlist -> stmt | stmt ; stmtlist
stmt -> simplstmt | structstmt
simplstmt -> assignstmt | iostmt
assignstmt -> IDENTIFIER = expression
expression -> expression + | - term | term
iostmt -> read ( IDENTIFIER ) | write ( IDENTIFIER | CONSTANT )
structstmt -> cmpdstmt | ifstmt | forstmt
ifstmt -> if condition : stmt [ else : stmt ]
forstmt -> for assignstmt condition assignstmt : stmt
condition -> expression RELATION expression
RELATION -> < | <= | == | != | >= | >

seq2.txt
BEGIN
Integer
n
Boolean
isPrime
=
True
read
(
n
)
if
n
```

```
<
2
:
isPrime
=
False
for
index
=
4
,
index
<
sqrt
(
n
/
2
)
+
1
,
index
++
:
if
n
%
i
==
0
:
isPrime
=
False
if
isPrime
==
True
:
print
(
″
N is prime
″
)
else
print
(
```

```
″
N is not prime
″
)
END
```