SARS Language
SER 502 - Team 6
Spring 2020

https://github.com/Suraj7696/SER-502-2020-Project-Team-6

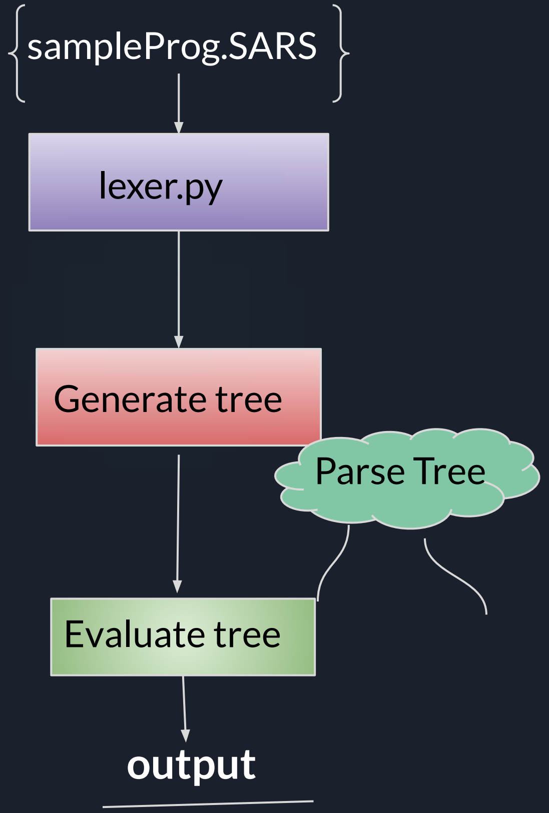
Sheran Dass
Akshay Kumar
Ria Mehta
Suraj Atmakumari

Contents:

INTRODUCTION - SARS

- SARS is an abbreviation of names of the team members - Sheran, Akshay, Ria, Suraj (and also very apt at this time)
- We are generating the tokens using python file lexer.py
- Parse tree generation is done using Prolog
- Language Design:
  - Every program has a ***begin{*** and an ***}end***.
  - Input program has a **.SARS** extension
  - lexer.py file generates tokens which are then processed using prolog to give the final output.

FLOW OF PROCESSING

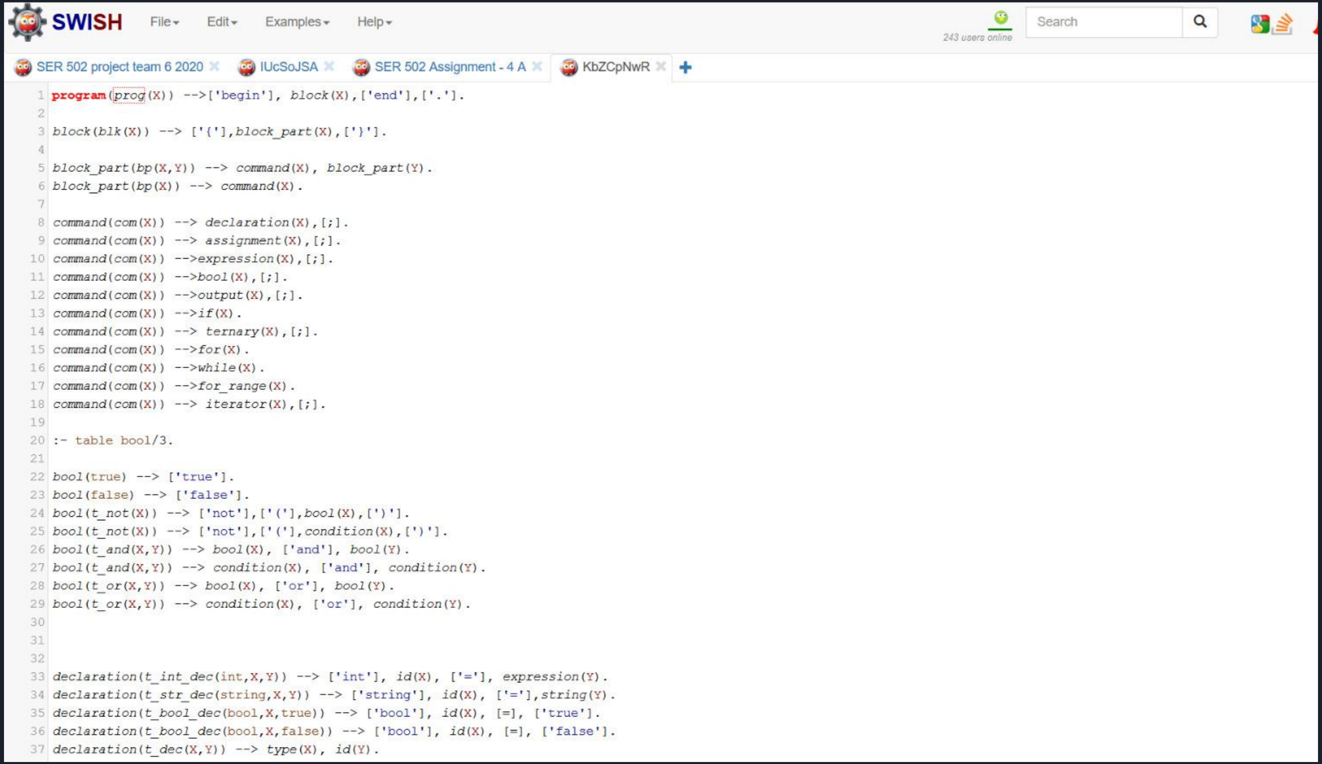


TOOLS

- Compiler
  - Prolog
- Runtime
  - Python3 for token generation
  - SWI-Prolog

GRAMMAR

We have used DCG to define the Grammar:



```prolog
program(prog(X)) -->['begin'], block(X),['end'],['.'].

block(blk(X)) --> ['{'],block_part(X),['}'].

block_part(bp(X,Y)) --> command(X), block_part(Y).
block_part(bp(X)) --> command(X).

command(com(X)) --> declaration(X),[;].
command(com(X)) --> assignment(X),[;].
command(com(X)) -->expression(X),[;].
command(com(X)) -->bool(X),[;].
command(com(X)) -->output(X),[;].
command(com(X)) -->if(X).
command(com(X)) --> ternary(X),[;].
command(com(X)) -->for(X).
command(com(X)) -->while(X).
command(com(X)) -->for_range(X).
command(com(X)) --> iterator(X),[;].

:- table bool/3.

bool(true) --> ['true'].
bool(false) --> ['false'].
bool(t_not(X)) --> ['not'],['('],bool(X),[')'].
bool(t_not(X)) --> ['not'],['('],condition(X),[')'].
bool(t_and(X,Y)) --> bool(X), ['and'], bool(Y).
bool(t_and(X,Y)) --> condition(X), ['and'], condition(Y).
bool(t_or(X,Y)) --> bool(X), ['or'], bool(Y).
bool(t_or(X,Y)) --> condition(X), ['or'], condition(Y).



declaration(t_int_dec(int,X,Y)) --> ['int'], id(X), ['='], expression(Y).
declaration(t_str_dec(string,X,Y)) --> ['string'], id(X), ['='],string(Y).
declaration(t_bool_dec(bool,X,true)) --> ['bool'], id(X), [=], ['true'].
declaration(t_bool_dec(bool,X,false)) --> ['bool'], id(X), [=], ['false'].
declaration(t_dec(X,Y)) --> type(X), id(Y).
```

FEATURES OF SARS

- ➢ Datatypes:
  - ○ Int - 1,2,3,4 ...
  - ○ Bool - true/false
  - ○ String - “HelloWorld”
- ➢ Arithmetic Operations:
  - ○ Addition +
  - ○ Subtraction -
  - ○ Multiplication *
  - ○ Division /
- ➢ Relational Operators:
  - ○ Equal to ==
  - ○ Not Equal to !=
  - ○ Greater than >
  - ○ Greater than or equal to >=
  - ○ Less than < |
  - ○ Less than or equal to <=
- ➢ Increment/ Decrement operators
  - ○ ++
  - ○ --

FEATURES OF SARS - CONTINUED
(TERNARY OPERATOR)

Represented in the grammar as:

```
<ternary> ::= <identifier> <condition_operators> <identifier> ? <block> : <block>

            | <identifier> <condition_operators> <number> ? <block> : <block>

            | <identifier> <condition_operators> <string> ? <block> : <block>
```

Can be used as:

```
X > Y?  print(“X is greater”): print(“X is not greater”);
```

FEATURES OF SARS - CONTINUED *(STATEMENTS)*

- General Statements
    - Print statement print(X)
    - Declaration int i;
    - Assignment int i=0; x=1;

- If Condition

```
if (condition){
        <block>
else{
        <block>
}
```

(else is optional)

FEATURES OF SARS - CONTINUED *(LOOPS)*

- For Loops
  - Simple For Loop

```
for(int i=0;i<10;i++)
{
     print(i);
}
```

  - For loop with range given

```
for x in range (0:10)
{
     print(x);
}
```

- While loop

```
while(x>y)
{
      x++;
      y--;
 }
```

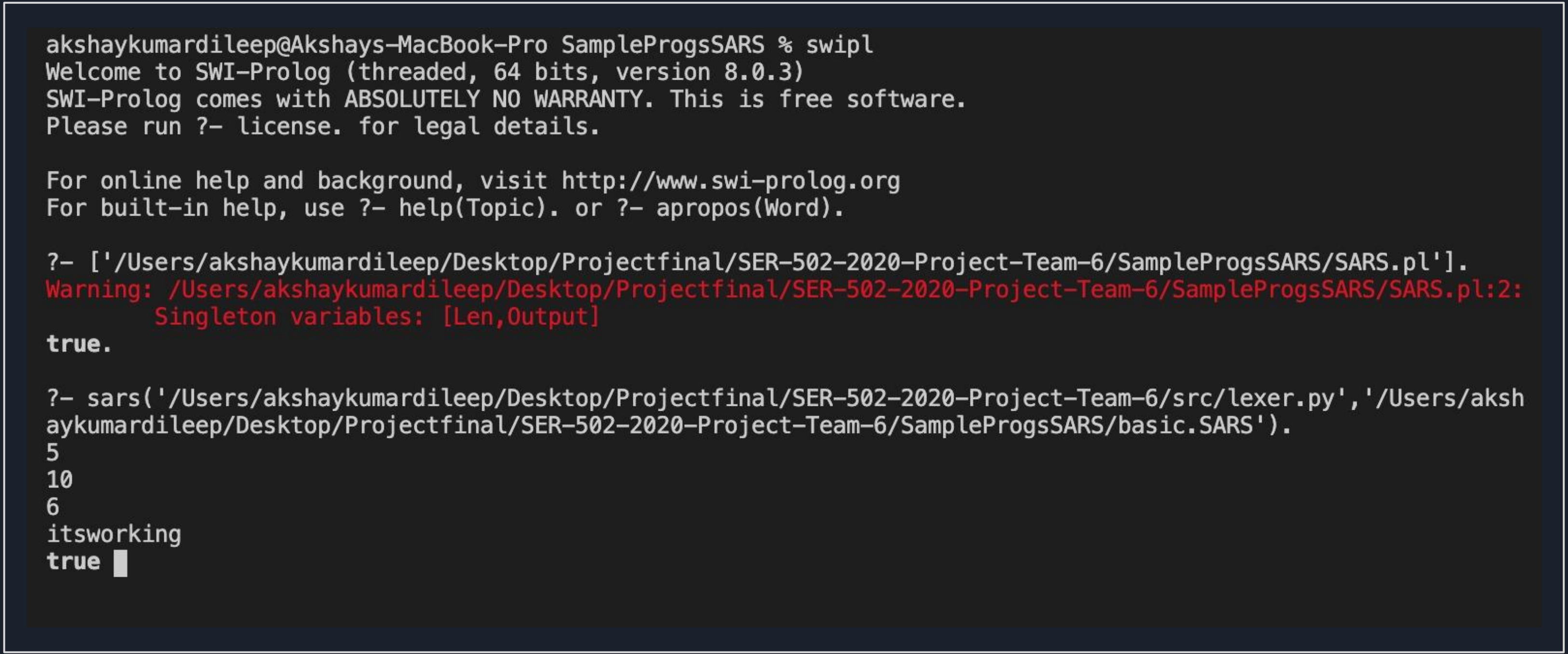
DEMONSTRATION OF THE LANGUAGE

- Steps:
  - Open swipl on terminal
  - Compile the SARS.pl file using the following command
    - [‘<path to SARS.pl file>’].
  - Run the Program:
    - sars(‘<path to the lexer.py file>’, ‘<path to the program file with SARS extension>’)

EXAMPLE PROGRAM

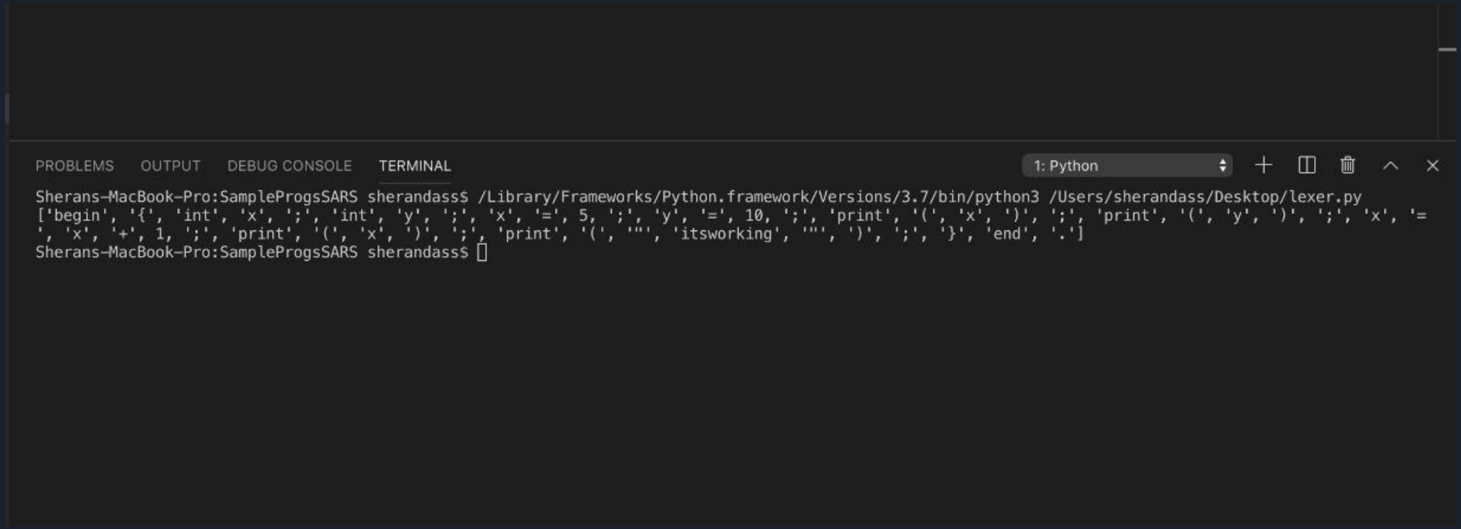
```
begin
{
int x;
int y;
x=5;
y=10;

print(x);
print(y);

x = x + 1;
print(x);

print ("its working");
}
end.
```

EXECUTION



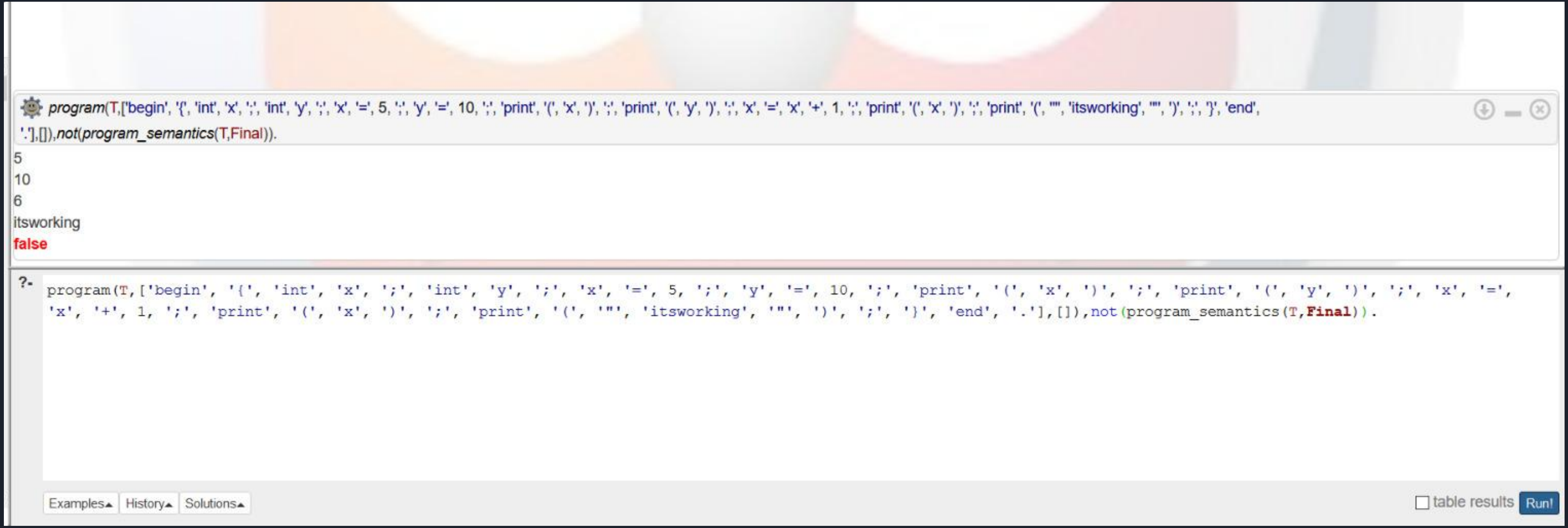
```
akshaykumardileep@Akshays-MacBook-Pro SampleProgsSARS % swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.0.3)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- ['/Users/akshaykumardileep/Desktop/Projectfinal/SER-502-2020-Project-Team-6/SampleProgsSARS/SARS.pl'].
Warning: /Users/akshaykumardileep/Desktop/Projectfinal/SER-502-2020-Project-Team-6/SampleProgsSARS/SARS.pl:2:
        Singleton variables: [Len,Output]
true.

?- sars('/Users/akshaykumardileep/Desktop/Projectfinal/SER-502-2020-Project-Team-6/src/lexer.py','/Users/aksh
aykumardileep/Desktop/Projectfinal/SER-502-2020-Project-Team-6/SampleProgsSARS/basic.SARS').
5
10
6
itsworking
true
```

LEXER.PY OUTPUT



```
Sherans-MacBook-Pro:SampleProgsSARS sherandass$ /Library/Frameworks/Python.framework/Versions/3.7/bin/python3 /Users/sherandass/Desktop/lexer.py
['begin', '{', 'int', 'x', ';', 'int', 'y', ';', 'x', '=', 5, ';', 'y', '=', 10, ';', 'print', '(', 'x', ')', ';', 'print', '(', 'y', ')', ';', 'x', '=
', 'x', '+', 1, ';', 'print', '(', 'x', ')', ';', 'print', '(', '"', 'itsworking', '"', ')', ';', '}', 'end', '.']
Sherans-MacBook-Pro:SampleProgsSARS sherandass$ 
```

GRAMMAR AND EVALUATION OUTPUT



```
program(T,['begin', '{', 'int', 'x', ';', 'int', 'y', ';', 'x', '=', 5, ';', 'y', '=', 10, ';', 'print', '(', 'x', ')', ';', 'print', '(', 'y', ')', ';', 'x', '=', 'x', '+', 1, ';', 'print', '(', 'x', ')', ';', 'print', '(', '"', 'itsworking', '"', ')', ';', '}', 'end', '.'],[]),not(program_semantics(T,Final)).
5
10
6
itsworking
false
```

```
?- program(T,['begin', '{', 'int', 'x', ';', 'int', 'y', ';', 'x', '=', 5, ';', 'y', '=', 10, ';', 'print', '(', 'x', ')', ';', 'print', '(', 'y', ')', ';', 'x', '=', 'x', '+', 1, ';', 'print', '(', 'x', ')', ';', 'print', '(', '"', 'itsworking', '"', ')', ';', '}', 'end', '.'],[]),not(program_semantics(T,Final)).
```