MY OWN TINY PROGRAMMING LANGUAGE

Tiny programming language is a programming language created by us using sly library in Python 3.6. SLY provides two separate classes Lexer and Parser. The Lexer class is used to break input text into a collection of tokens specified by a collection of regular expression rules. The Parser class is used to recognize language syntax that has been specified in the form of a context free grammar. The idea was to make a single line programming language.

Important Note:

- The variable name can start with a name and an underscore. No special characters are allowed except underscore. Arithmetic operations can be done on variables. EX: A=10, B=20 then A=A+B will result into A=30.
- The user can define comments using #. Example, #comments
- The keywords used in our language is IF, THEN, ELSE, FOR, FUN, TO, print, NEXT, ENDIF.
- Keywords and identifiers are case sensitive in our language. The variable can contain an integer. For string and floating-point work is still in progress.

Syntax of tiny programming language

- Print statement is defined as
  print “hello world” will give hello world as output.

- a=10 will declare a variable a with value a

- print a can be used to output the value of a or writing a will also retrieve the same result.

- IF-ELSE statement:
  IF condition THEN statement ELSE statement ENDIF

- FOR loop:
  FOR var_assign TO expr NEXT statement
  Ex: FOR I=0 TO 5 NEXT print a

- FUNCTIONS:
  FUN function_name()->FOR i=0 TO 5 NEXT print a

OUTPUT:

LEXICAL PHASE

```
---------------------------TINY PROGRAMMING LANGUAGE ----------------------------
 Lexical Phase

--> print "hello"
Token(type='PRINT', value='print', lineno=1, index=0)
Token(type='STRING', value='"hello"', lineno=1, index=6)

--> a=10
Token(type='NAME', value='a', lineno=1, index=0)
Token(type='=', value='=', lineno=1, index=1)
Token(type='NUMBER', value='10', lineno=1, index=2)

--> print a
Token(type='PRINT', value='print', lineno=1, index=0)
Token(type='NAME', value='a', lineno=1, index=6)

--> IF a==6 THEN a=10 ELSE a=0 ENDIF
Token(type='IF', value='IF', lineno=1, index=0)
Token(type='NAME', value='a', lineno=1, index=3)
Token(type='EQEQ', value='==', lineno=1, index=4)
Token(type='NUMBER', value='6', lineno=1, index=6)
Token(type='THEN', value='THEN', lineno=1, index=8)
Token(type='NAME', value='a', lineno=1, index=13)
Token(type='=', value='=', lineno=1, index=14)
Token(type='NUMBER', value='10', lineno=1, index=15)
Token(type='ELSE', value='ELSE', lineno=1, index=18)
Token(type='NAME', value='a', lineno=1, index=23)
Token(type='=', value='=', lineno=1, index=24)
Token(type='NUMBER', value='0', lineno=1, index=25)
Token(type='ENDIF', value='ENDIF', lineno=1, index=27)

--> FOR I=0 TO 5 NEXT print a
Token(type='FOR', value='FOR', lineno=1, index=0)
Token(type='NAME', value='I', lineno=1, index=4)
Token(type='=', value='=', lineno=1, index=5)
Token(type='NUMBER', value='0', lineno=1, index=6)
Token(type='TO', value='TO', lineno=1, index=8)
Token(type='NUMBER', value='5', lineno=1, index=11)
Token(type='NEXT', value='NEXT', lineno=1, index=13)
Token(type='PRINT', value='print', lineno=1, index=18)
Token(type='NAME', value='a', lineno=1, index=24)
```

PARSING PHASE

In this phase the syntax tree is generated and if the syntax is invalid the compiler will report an error message.

```
Parsing Phasing

--> print "hello"
Tree Generated is as follows :
('PRINT', '"hello"')

--> a=10
Tree Generated is as follows :
('var_assign', 'a', '10')

--> IF a==10 THEN a=10 ELSE a=0 ENDIF
Tree Generated is as follows :
('if_stmt', ('condition_eqeq', ('var', 'a'), ('num', '10')), ('branch', ('var_assign', 'a', '10'),
('var_assign', 'a', '0')))

--> FOR I=0 TO 5 NEXT print a
Tree Generated is as follows :
('for_loop', ('for_loop_setup', ('var_assign', 'I', '0'), ('num', '5')), ('var', 'a'))

--> FUN hello()->FOR i=0 TO 5 print a
yacc: Syntax error at line 1, token=PRINT
Tree Generated is as follows :
('var', 'a')

--> FUN hello()->FOR i=0 TO 5 NEXT print a
Tree Generated is as follows :
('fun_def', 'hello', ('for_loop', ('for_loop_setup', ('var_assign', 'i', '0'), ('num', '5')), ('var', 'a')))
```

TINY PROGRAMMING LANGUAGE:

```
--------------------------TINY PROGRAMMING LANGUAGE ---------------------------

--> print "HELLO WORLD"
HELLO WORLD

--> a=10

--> print a
10

--> IF a==6 THEN a=10 ELSE a=0 ENDIF

--> print a
10

--> FOR I=0 TO 5 NEXT print a
10
10
10
10
10
```

```
--> FUN hello()->FOR I=0 TO 5 NEXT print a

--> hello()
10
10
10
10
10
```

```
--> a=10

--> b=20

--> a=a+b

--> print a
30
```

STEPS TO EXECUTE:

1. Install Python 3.6 and sly library.
2. Using anaconda prompt: Command pip -install sly can be used to install sly library.
3. Run tiny.py to execute the program(executed on Spyder).
4. For checking lexer and parsing phase result file “lexing.py” and “parsing.py” can be executed.