ΑΡΧΕΣ ΓΛΩΣΣΩΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΚΑΙ ΜΕΤΑΦΡΑΣΤΩΝ

ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ

ΟΜΑΔΑ:

Δέρβου Αικατερίνη, 1054185, 3ο έτος, st1054185@ceid.upatras.gr
Κουκουβέλα Ασπασία, 1059617, 3ο έτος, st1059617@ceid.upatras.gr
Μαχιά Αριάδνη, 1059556, 3ο έτος, st1059556@ceid.upatras.gr
Φουσέκη Αθηνά, 1059623, 3ο έτος, st1059623@ceid.upatras.gr

1. BNF Ορισμός Γλώσσας

<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<number> ::= <digit> | <number><digit>

<float> ::= <number>.<number>

<alpha> ::= a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z |
A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

<alphanum> ::= <alpha>|<number>

<underscore> ::= _ | _ <underscore> | ε

<ID> ::= <underscore><alpha> | <ID><alpha>_ | <ID><underscore><alphanum>

<PHRASE> ::= <ID> | <PHRASE>' '<ID>

<STRING> ::= " <PHRASE> "

<import_modules> ::= import <ID> \n | import <ID> as <ID> \n | import <ID> from <ID> \n | import <ID> from <ID> as <ID> \n

<class_def> ::= class <ID> : <class_body>

<class_body> ::= \t <class_constr> <class_body> | \t <def_func> <class_body> | ε

<class_constr> ::= def __init__(self, <parameters>): <self_assign>

<self_assign> ::= \t\t self.<ID> = <ID> <self_assign> | ε

<obj_create> ::= <ID> = <ID>(<parameters>) \n

<assgn_stmt> ::= <ID> = <ID> \n | <ID> = <number> \n | <ID> = <float> \n

<def_func> ::= def <ID>(<parameters>): \n <tail>

<parameters> ::= <pars> <ID> | ε

<pars> ::= <pars> <ID>, | ε

<print_stmt> ::= print(<STRING>) \n

<if_stmt> ::= if <expression> : \n <tail>

<tail> ::= <tail> <tb> <commands_in> | <tb> <commands_in>

<tb> ::= \t | <tb> \t

<commands_in> ::= <print_stmt> | <assgn_stmt> | <def_func> | <if_stmt> | <for_loop>

<expression> ::= <bool_st> | <cmprsn>

<bool_st> ::= <ID> <equal> <boolean>

<equal> ::= not | ==

<boolean> ::= True | False

<cmprsn> ::= <ID> <compare> <ID> | <ID> <compare> <number> | <ID> <compare> <float>

<compare> ::= == | != | >= | <= | > | <

<for_loop> ::= for <ID> in <ID> : \n <tail> | for <ID> in <range_tool> : \n <tail>

<range_tool> ::= range (<number> , <number>)

2.Αρχείο FLEX (lexerino3.l)

```
%option case-sensitive          //Θέτουμε case-sensitivity όσον αφορά τη γλώσσα μας
%option yylineno
%option noyywrap
%option reject

%{

#include <stdio.h>             //Τομέας των ορισμών
#include <string.h>
#include <unistd.h>
#include <stdlib.h>
#include "parserino3.tab.h"    //include για διασύνδεση flex και bison αρχείου

%}


number                         ([0-9])+                  //χρήσει regex ορίζουμε τα συστατικά
float                          {number}\.{number}  // της γλώσσας μας
alpha                          [a-zA-Z]
alphanum                       {alpha}|{number}

ID                             \_*{alpha}{alphanum}*(_{alphanum}+)*_?
STRING                         \"{ID}(" "{ID})*\"
ASCII                          [ !"$%&'()*+,-./:;<=>?@\[\\\]^_`{|}~]
TEXT                           ({ID}*" "+{ASCII}*)*
%%
//Πεδίο των translation rules της μορφής: pattern {action}

// keyoword tokens
"import"                       {printf("import \t"); return T_IMPORT;}
{number}                       {printf("integer\t"); yylval.intval =atoi( yytext);return T_INT;}
                                    //μετατρέπουμε το string που έχουμε κάνει capture
                                    //από το κείμενο/κώδικα που αναλύουμε σε integer,
                                    //ώστε να το χρησιμοποιήσουμε για
                                    //την υλοποίηση πράξεων
```

```
"class"                         {printf("class\t"); return T_CLASS;}
" "                             {printf("whitespace\t");}  //κάνουμε διάφορες εκτυπώσεις για
                                                           //να ελέγξουμε την ορθότητα της
                                                           //λειτουργίας του αναλυτή μας
"list"                          {return T_LIST;}
"continue"                      {return T_CONTINUE;}
"break"                         {return T_BREAK;}
"if"                            {printf("if\t"); return T_IF;}
"else"                          {return T_ELSE;}
"for"                           {printf("for\t"); return T_FOR;}
"default"                       {return T_DEFAULT;}

"len"                           {return T_LEN;}
"new"                           {return T_NEW;}
"main"                          {return T_MAIN;}
"sizeof"                        {return T_SIZEOP;}

"from"                          {printf("from \t");return T_FROM;}
"as"                            {printf("as\t");return T_AS;}
\n                              {printf("\n"); return T_LINE;}
\t                              {return T_TAB;}
"def"                           {printf("def\t"); return T_DEF;}
"or"                            {printf("or\t");  return T_OR;    }
"and"                           {printf("and\t");          return T_AND;   }
"not"|"=="                      {printf("not|==\t"); return T_EQ;    }
"True"|"False"                  {printf("boolean variable \t");return T_BOOL;}
"return"                        {printf("return \t"); return T_RETURN;}
"print"                         {printf("print\t"); return T_PRINT;}
"__init__"                      {printf("INIT\t"); return T_INIT;}
"self"                          {printf("SELF\t"); return T_SELF;}
"lambda"                        {return T_LAMBDA;}
"elif"                          {return T_ELIF;}
"dict"                          {return T_DICT;}
"range"                         {printf("range\t");return T_RANGE;}
"in"                            {printf("in\t"); return T_IN;}

{float}                         {printf("float\t"); yylval.flval =atof( yytext); return T_FLOAT;}
{STRING}                        {printf("string\t");return T_STRING;}
{ID}                            {printf("identifier \t"); yylval.strval=strdup(yytext); return
                                 T_ID;}                    //ομοίως λειτουργούμε για floats και
                                                           //τους Identifiers
```

// άλλα tokens, σύμβολα που χρησιμοποιούμε στη γλώσσα

```
"||"                            {return T_OROP;}
"&&"                            {return T_ANDOP;}
"=="|"!="                       {return T_EQUOP;}
">="|"<="|">"|"<"               {return T_RELOP;}
"+"                             {return T_ADD;}
"-"                             {return T_SUB;}
"*"                             {return T_MUL;}
"/"                             {return T_DIV;}
"!"                             {return T_NOTOP;}
```

```
"("                             {printf("left paren\t");return T_LPAREN;}
")"                             {printf("right paren\t");return T_RPAREN;}
";"                             {return T_SEMI;}
"."                             {printf("DOT \t");return T_DOT;}
","                             {printf("COMMA \t");return T_COMMA;}
"="                              {printf("ASSIGN \t");return T_ASSIGN;}
"::"                            {return T_METH;}
":"                             {printf("COLON\t"); return T_COLON;}
"["                             {return T_LBRACK;}
"]"                             {return T_RBRACK;}
"&"                             {return T_REFER;}
"{"                             {return T_LBRACE;}
"}"                             {return T_RBRACE;}
"#".*                           {printf("COMMENT HEEERE\n");}


.                               { printf("\nUnrecognised character at line \n");return
                                  T_ERROR;}
                                //Με την τελεία, "." , από τη θεωρία των regex καλύπτουμε
                                //όλο το φάσμα των χαρακτήρων, εκτός από το "\n". Συνεπώς,
                                //αποτελεί τον τελευταίο κανόνα έτσι ώστε:
                                //              1) Για να έχουμε τρόπο να αντιστοιχίσουμε
                                //              όσους χαρακτήρες δεν έχουμε δηλώσει πιο
                                //              πάνω
                                //              2) Για να μην επικαλύπτει κάποιον από τους
                                //              υπόλοιπους κανόνες
```

3. Αρχείο Bison (parserino3.y)

```
%{
#include <stdio.h>              //απαραίτητες βιβλιοθήκες
#include <math.h>
#include <string.h>
#include <stdlib.h>


void yyerror(char *);
extern FILE *yyin;              //μεταβλητές για είσοδο και έξοδο αρχείου
extern FILE *yyout;
int yylex();
extern int yylineno;
int yylex(void);
char *var;                      //μεταβλητές που χρησιμοποιούμε σε πράξεις με tokens
char *var1;
char *var2;
int y=0;
%}
```

```
%union                  //union που ορίσαμε τα types των διαφόρων tokens
{
        char *strval;
        int intval;
        float flval;
};

%token <strval> T_ELSE T_WHILE T_RETURN T_IF T_ANY T_SELF T_AS T_LAMBDA T_ELIF T_DICT
T_OROP T_ANDOP T_EQUOP  T_RELOP
%token <strval> T_DEF T_INIT T_CLASS T_COLON T_FOR T_RANGE T_IN T_BOOL T_PRINT T_NOTOP
T_DOT T_METH T_ID T_STRING
%token <strval> T_FROM T_CONTINUE T_BREAK T_DEFAULT T_LEN T_NEW  T_MAIN T_SIZEOP
T_ERROR
%token <strval> T_IMPORT  T_LINE T_TAB T_EQ T_OR T_AND T_NOT  T_START T_LIST T_COMMNT
%token <strval> T_LPAREN T_RPAREN T_SEMI T_COMMA T_LBRACK T_RBRACK T_REFER
T_LBRACE T_RBRACE
%token <flval> T_FLOAT
%token <intval> T_INT

%left <strval> T_ADD T_SUB
%left <strval> T_MUL T_DIV
%right <strval> T_ASSIGN

%%
//-----R U L E S --------

prog
        :stmt
        |prog stmt
        ;

stmt
        :imp_stmt
        |assgn_stmt
        |def_func
        |print_stmt
        |if_stmt
        |for_loop
        |class_def
        |obj_create
        |function_call
        |T_LINE
        ;

imp_stmt    //με αυτό τον κανόνα φροντίζουμε για το parsing όλων των τύπων import
        :T_IMPORT T_ID T_LINE                   {var = $2;
                                                printf("IMPORTING MODULE: %s \n", var);}

        |T_IMPORT T_ID T_AS T_ID T_LINE     {var = $2; var1 = $4;
                                              printf("IMPORTING MODULE: %s AS: %s \n", var, var1);}
        |T_IMPORT T_ID T_FROM T_ID T_LINE           {var = $2; var1 = $4; printf("IMPORTING
                                                    MODULE: %s FROM: %s\n", var, var1);}
        |T_IMPORT T_ID T_FROM T_ID T_AS T_ID T_LINE                 {var = $2; var1 = $4; var2 = $6;
                                printf("IMPORTING MODULE: %s FROM: %s AS: %s\n", var, var1, var2);}
                ;
```

```
class_def:          //κανόνας για τον ορισμό κλάσης
                    T_CLASS T_ID T_COLON T_LINE class_body
                            {var = $2; printf("CREATED CLASS: %s\n", var);}
                    ;

class_body:         //κανόνας για τη μορφή του εσωτερικού μίας κλάσης
                    T_TAB class_constr class_body
                    |T_TAB def_func class_body
                    |/*empty */
                    ;

class_constr:       //κανόνας για τον constructor της κλάσης
                    T_DEF T_INIT T_LPAREN T_SELF T_COMMA parameters T_RPAREN T_COLON
                    T_LINE self_assign
                            {printf("THIS IS A CLASS CONSTRUCTOR \n");}
                    ;

self_assign:        //αρχικοποίηση των γνωρισμάτων της κλάσης
                    T_TAB T_TAB T_SELF T_DOT T_ID T_ASSIGN T_ID T_LINE self_assign
                    | /* empty */
                    ;

obj_create:         //δημιουργία αντικειμένου
                    T_ID T_ASSIGN T_ID T_LPAREN parameters T_RPAREN T_LINE
                            {printf("OBJECT CREATION\n");}
                    ;


assgn_stmt:         //ανάθεση τιμής σε μεταβλητή
                    T_ID T_ASSIGN T_ID T_LINE
                            {printf("ASSIGNING A VARIABLE TO A VARIABLE\n");}
                    |T_ID T_ASSIGN T_INT T_LINE
                            {printf("ASSIGNING AN INTEGER TO A VARIABLE\n");}
                    |T_ID T_ASSIGN T_FLOAT T_LINE
                            {printf("ASSIGNING A FLOAT TO A VARIABLE\n");}
                    |T_ID T_ASSIGN operation T_LINE
                            {printf("THIS IS AN ASSING STATEMENT WITH AN OPERATION\n");}
                    ;


operation :         // υλοποίηση των τελεστών +, -, *, /
                    T_INT T_ADD T_INT
                            {printf("This is an ADDITION: %d + %d = %d \n",$1, $3,($1 + $3));}
                    |T_INT T_SUB T_INT
                            {printf("This is an SUBTRACTION: %d \n", ($1 - $3));}
                    |T_INT T_MUL T_INT
                            {printf("This is an MULTIPLICATION: %d \n",($1 * $3));}
                    |T_INT T_DIV T_INT
                            {printf("This is an DIVISION: %d \n", ($1 / $3));}
                    |T_FLOAT T_ADD T_FLOAT
                            {printf("This is an ADDITION: %.3f + %.3f = %.3f \n",$1, $3, ($1 + $3));}
                    |T_FLOAT T_SUB T_FLOAT
                            {printf("This is an SUBTRACTION: %.3f \n", ($1 - $3));}
                    |T_FLOAT T_MUL T_FLOAT
```

```
        {printf("This is an MULTIPLICATION: %.3f \n",($1 * $3));}
    |T_FLOAT T_DIV T_FLOAT
        {printf("This is an DIVISION: %.3f \n",($1 / $3));}
    |T_FLOAT T_ADD T_INT
        {printf("This is an ADDITION: %.3f + %d = %.3f \n",$1, $3, ($1 + $3));}
    |T_FLOAT T_SUB T_INT
        {printf("This is an SUBTRACTION: %.3f \n", ($1 - $3));}
    |T_FLOAT T_MUL T_INT
        {printf("This is an MULTIPLICATION: %.3f \n",($1 * $3));}
    |T_FLOAT T_DIV T_INT
        {printf("This is an DIVISION: %.3f \n",($1 / $3));}
    |T_INT  T_ADD T_FLOAT
        {printf("This is an ADDITION: %.d + %.3f = %.3f \n",$1, $3, ($1 + $3));}
    |T_INT  T_SUB T_FLOAT
        {printf("This is an SUBTRACTION: %.3f \n", ($1 - $3));}
    |T_INT  T_MUL T_FLOAT
        {printf("This is an MULTIPLICATION: %.3f \n",($1 * $3));}
    |T_INT  T_DIV T_FLOAT
        {printf("This is an DIVISION: %.3f \n",($1 / $3));}

    ;
    //χρησιμοποιούμε τα $1 και $3 για να πάρουμε τις τιμή του 1ου token του
    //κανόνα και του 3ου token αντίστοιχα. Μέσω αυτών κάνουμε τις
    //απαραίτητες πράξεις για την υλοποίηση των τελεστών.



def_func:       //ορισμός συνάρτησης
                T_DEF T_ID T_LPAREN parameters T_RPAREN T_COLON T_LINE tail
                    {printf("FUNCTION DEFINITION HERE\n");}
                ;

parameters:     //κανόνας για τα arguments μιας συνάρτησης
                pars T_ID
                | /* empty */
                ;

pars:           pars T_ID T_COMMA
                | /* empty */
                ;

function_call:  //κλήση συνάρτησης
                T_ID T_LPAREN parameters T_RPAREN
                    {printf("FUNCTION CALL HERE\n");}


print_stmt:     //κανόνας για την print
                T_PRINT T_LPAREN T_STRING T_RPAREN T_LINE
                    {printf("THIS IS A PRINT LOL\n");}
                ;
```

```
if_stmt:
                T_IF expression T_COLON T_LINE tail          {printf("THIS IS AN IF
                                                                    STATEMENT\n");}
                |T_IF expression T_COLON T_LINE tail T_ELIF expression T_COLON T_LINE
                tail                {printf("THIS IS AN IF-ELIF STATEMENT\n");}
                |T_IF expression T_COLON T_LINE tail T_ELIF expression T_COLON T_LINE
                tail T_ELSE T_COLON T_LINE tail
                        {printf("THIS IS AN IF-ELIF-ELSE STATEMENT\n");}
                |T_IF expression T_COLON T_LINE tail T_ELSE T_COLON T_LINE tail
                        {printf("THIS IS AN IF-ELSE STATEMENT\n");}
                ;


tail:           //το μπλοκ μια if ή μιας συνάρτησης ή μιας for
                tail tb commands_in
                |tb commands_in
                ;


tb:             //κανόνας για την αναγνώριση των tabs σε εμφωλευμένες if, for κ.λπ.
                T_TAB
                |tb T_TAB
                ;

commands_in:    //κανόνας για τις εντολές που μπορούν να υπάρξουν μέσα σε ένα μπλοκ
                print_stmt
                |assgn_stmt
                |def_func
                |if_stmt
                |for_loop
                ;

expression:     //οι πιθανές συνθήκες σε μια if
                bool_st
                |cmprsn
                ;

bool_st:        //μορφή μιας συνθήκης με boolean μεταβλητή
                T_ID T_EQ T_BOOL
                ;

cmprsn:         //συνθήκες σύγκρισης
                T_ID compare T_ID
                |T_ID compare T_INT
                |T_ID compare T_FLOAT
                ;
compare:
                T_EQUOP
                |T_RELOP
                ;

for_loop:       //ορισμός δύο μορφών for loop
                T_FOR T_ID T_IN T_ID T_COLON T_LINE        tail
                        {printf("THIS IS A PYTHONIAN FOR LOOP\n");}
                |T_FOR T_ID T_IN range_tool T_COLON T_LINE        tail
```

```
                        {printf("THIS IS A FOR LOOP\n");}
                    ;

range_tool:         //μορφή της range() που χρησιμοποιούμε σε for loops
                    T_RANGE T_LPAREN T_INT T_COMMA T_INT T_RPAREN
                    ;


%%

void yyerror(char *s) {
      fprintf(stderr,"AT LINE %d : %s\n",yylineno,s);

}
int main ( int argc, char **argv )
{
  ++argv; --argc;
  if ( argc > 0 )
       yyin = fopen( argv[0], "r" );
  else
       yyin = stdin;
  yyout = fopen ( "output", "w" );
  yyparse ();




  return 0;
}
```

4. Παραδείγματα Εκτέλεσης της Εφαρμογής

Δημιουργήσαμε τα αρχεία flex και bison σε virtual box σε λειτουργικό lubuntu. Οι εντολές που χρησιμοποιήσαμε για το compiling των αρχείων κώδικα είναι:

```
bison -d parserino3.y
flex lexerino3.l
gcc -c parserino3.tab.c lex.yy.c
gcc parserino3.tab.o lex.yy.o -o parserino3
```

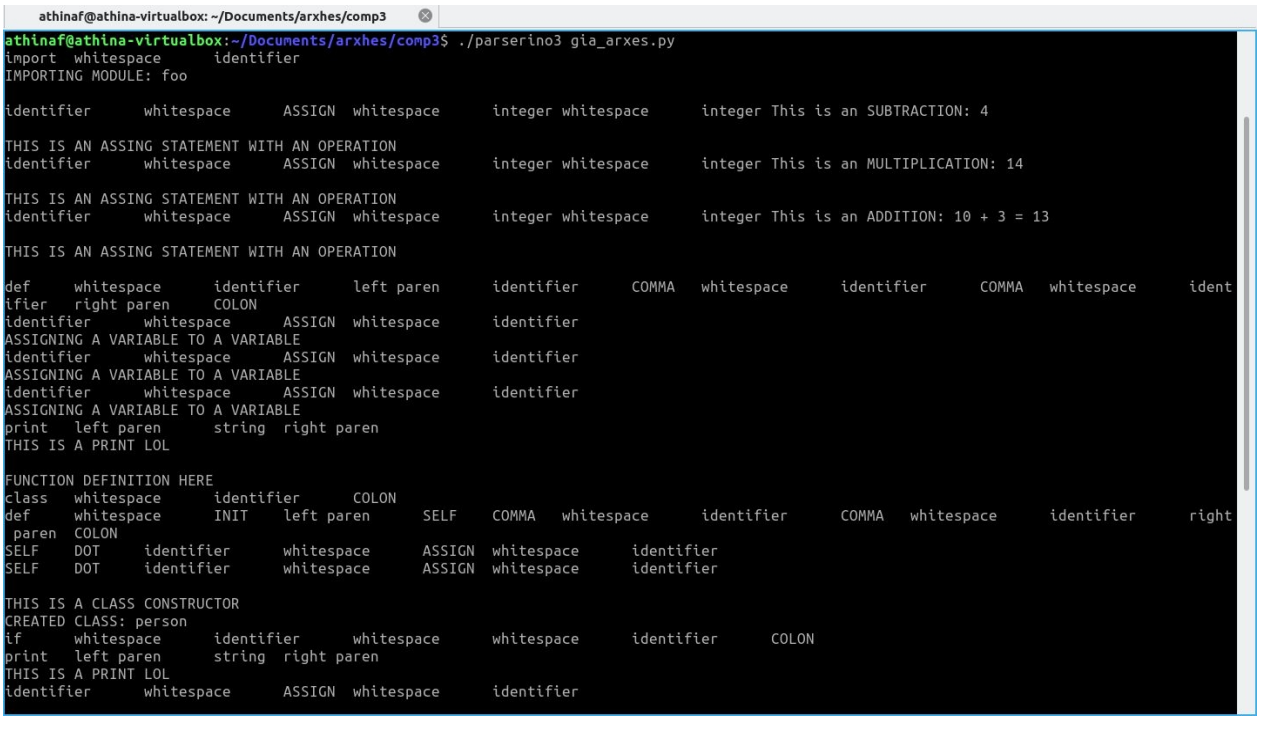
Δίνουμε ως είσοδο στον parser ένα αρχείο με κώδικα στη γλώσσα python και παίρνουμε ως έξοδο τη συντακτική του ανάλυση και εάν υπάρχει κάποιο λάθος στον κώδικα, εμφανίζεται μήνυμα λάθους.

Ο κώδικας python είναι ο εξής και περιέχει όλα τα ζητούμενα στοιχεία:

```
import foo                          | //ξεκινάμε με μια απλή import της foo,
                                    | // μιας dummie βιβλιοθήκης
```

```
x = 5 -1                                  | //έχουμε διάφορες αναθέσεις τιμών σε
b = 7 *2                                  | //μεταβλητές για να δείξουμε την
a = 10 +3                                 | // υλοποίηση των τελεστών
                                          |
def my_func(x, b, a):                     | //Έχουμε τον ορισμό μια συνάρτησης
        x = x                             | //με απλές αναθέσεις τιμών
        b = b                             |
        a = a                             |
        print("I am in a function")       | //και μια print
                                          |
class person:                             | //Δημιουργούμε την κλάση person
        def __init__(self, name, age):    | //τον constructor με δύο γνωρίσματα
                self.name = name          | //αρχικοποιούμε τα γνωρίσματα
                self.age = age            |
                                          |
if x < b:                                 | //Ακολουθεί μια if, μια nested  if και
        print("x is smaller than b")       | // ένα for loop μέσα στη nested if
        x = b                             |
        if x < a:                         |
                print("x is smaller than a")|
                for i in range(1,10):     |
                        print("Hello World")|
                                          |
                                          |
my_func(a,b,x)                            | //κάνουμε κλήση της συνάρτησης
                                          |
p1 = MyClass()                            | //δημιουργούμε το αντικείμενο p1 της
                                          | //κλάσης person
#THIS IS THE RIGHT FILE                   | //comment
                                          |
                                          |
```

Εκτελούμε το πρόγραμμά μας με την εντολή: `./parserino3 gia_arxes.py`
Παίρνουμε την ακόλουθη έξοδο:



```
ASSIGNING A VARIABLE TO A VARIABLE
if      whitespace      identifier      whitespace      whitespace      identifier      COLON
print   left paren      string  right paren
THIS IS A PRINT LOL
for     whitespace      identifier      whitespace      in      whitespace      range   left paren      integer COMMA   integer right paren C
OLON
print   left paren      string  right paren
THIS IS A PRINT LOL

THIS IS A FOR LOOP
THIS IS AN IF STATEMENT
THIS IS AN IF STATEMENT

identifier      left paren      identifier      COMMA   identifier      COMMA   identifier      right paren     FUNCTION CALL HERE


identifier      whitespace      ASSIGN  whitespace      identifier      left paren      right paren
OBJECT CREATION

COMMENT HEEERE

athinaf@athina-virtualbox:~/Documents/arxhes/comp3$
```

Στην έξοδο έχουμε εμφάνιση κάθε token όπως αυτό διαβάζεται από το αρχείο python. Εφόσον ο κώδικάς μας είναι συντακτικά σωστός, βλέπουμε ότι ο parser δεν μας εμφανίζει κάποιο error.

Στην περίπτωση που ο κώδικας python έχει κάποιο λάθος, π.χ λείπει μία παρένθεση από τον ορισμό της συνάρτησης θα έχουμε:

```python
def my_func x, b, a):
    x = x
    b = b
    a = a
    print("I am in a function")
```

και έξοδο:

```
athinaf@athina-virtualbox:~/Documents/arxhes/comp3$ ./parserino3 gia_arxes.py
import  whitespace      identifier
IMPORTING MODULE: foo

identifier      whitespace      ASSIGN  whitespace      integer whitespace      integer This is an SUBTRACTION: 4

THIS IS AN ASSING STATEMENT WITH AN OPERATION
identifier      whitespace      ASSIGN  whitespace      integer whitespace      integer This is an MULTIPLICATION: 14

THIS IS AN ASSING STATEMENT WITH AN OPERATION
identifier      whitespace      ASSIGN  whitespace      integer whitespace      integer This is an ADDITION: 10 + 3 = 13

THIS IS AN ASSING STATEMENT WITH AN OPERATION

AT LINE 7 : syntax error
def     whitespace      identifier      whitespace      identifier      athinaf@athina-virtualbox:~/Documents/arxhes/comp3$
```

Μας δείχνει ότι στη γραμμή 7 του προγράμματός μας εντοπίστηκε κάποιο συντακτικό λάθος.

5.Σχόλια

- Στον φάκελο .zip περιέχονται επιπλέον τα αρχεία flex-bison, ο κώδικας python και τα παραγόμενα αρχεία από το compiling των flex-bison.