

# Μέλη της Ομάδας:

Αργύρης Μάλλιος Α.Μ.:1051937 Έτος:5<sup>ο</sup> email: amallios@ceid.upatras.gr

Παναγιώτης Μπαρμπούνης Α.Μ: 1054382 Έτος:4<sup>ο</sup> email: up1054382@upnet.gr

Κωνσταντίνος Γκαγιάννης Α.Μ.: 1054426 Έτος:4<sup>ο</sup> email: up1054426@upnet.gr

Αλέξανδρος Νατσολλάρη Α.Μ.:1057769 Έτος:4<sup>ο</sup> email: up1057769@upnet.gr

Τα **ΤΕΛΙΚΑ** αρχεία περιγραφής της γλώσσας, τα οποία δίνονται ως **είσοδος** στα **Flex και Bison** βρίσκονται στις **τελευταίες σελίδες του pdf**. Παρακάτω παραθέτουμε screenshot εκτέλεσης του parser αλλά όπως θα δείτε και από το αρχείο input.txt έχουμε παραθέσει κάθε δυνατό συνδυασμό εισόδου για να τεστάρουμε τον κώδικα μας. Αν θεωρείτε πως δεν χρειάζεται μπορείτε να αφαιρέσετε το πρώτο κομμάτι μέχρι εκεί που ξεκάνει το <<class ceid:>>. Επίσης για μεγαλύτερη ευκολία έχουμε βάλει μια printf σχεδόν σε κάθε κομμάτι που χρειάζεται δόκιμη έτσι ώστε όταν εντοπίζεται ένα στοιχείο στο αρχείο input.txt να βλέπουμε και τα αντίστοιχα αποτελέσματα. Για παράδειγμα αν εντοπίσουμε for loop έχουμε βάλει να τυπώνεται και το αντίστοιχο μήνυμα κατά την εκτέλεση. Στην λύση μας έχουμε υλοποιήσει το πρώτο ερώτημα καθώς και τα dictionaries από το τρίτο ερώτημα. Γνωρίζουμε ότι κατά το compile εμφανίζονται warnings για shift/reduce και reduce/reduce τα οποία όμως δε μπορούμε να διορθώσουμε καθώς έχουμε χρησιμοποιήσει ίδια tokens σε πολλαπλούς κανόνες με μικρές αλλαγές ωστόσο που έχουν νόημα για την δημιουργία νέου κανόνα διότι θέλαμε σε κάποια σημεία να φτιάξουμε αναδρομικότητα στους κανόνες έτσι ώστε να μπορούν να χρησιμοποιούνται αρκετές φορές στο input μας. Δηλαδή ένα παράδειγμα θα ήταν οι κανόνες :

```
simple_statements: statements {printf("STATEMENT\n");}
```

```
|simple_statements statements {printf("SIMPLESTATEMENT\n");}
```

```
|pass_stm
```

```
|del_stmt
```

```
|return_stmt
```

```
|yield_stmt
```

```
|raise_stmt
```

```
|break_stmt
```

```
|continue_stmt
```

```
|import_stmt
```

|global\_stmt

|nonlocal\_stmt

;

Και όπως θα δείτε και στο αρχείο projbison.y το **|continue\_stmt** είναι μόνο μια λέξη. Το οποίο καθιστά τον κανόνα **continue\_stmt: CONTINUE** κατά μια έννοια άχρηστο, αφού θα μπορούσαμε να βάλουμε το continue απευθείας στο simple\_statements. Ωστόσο αυτό κατά την κρίση μας δεν είναι και πολύ πρακτικό στο να φαίνεται η υλοποίηση του parser όσο το δυνατόν πιο κατανοητή σε κάποιο τρίτο που την διαβάζει και δε ξέρει ότι με την λέξη continue θέλαμε να υλοποιήσουμε το statement continue. Τέλος κάποια από τα shift/reduce και reduce/reduce προέρχονται από κανόνες του τύπου:

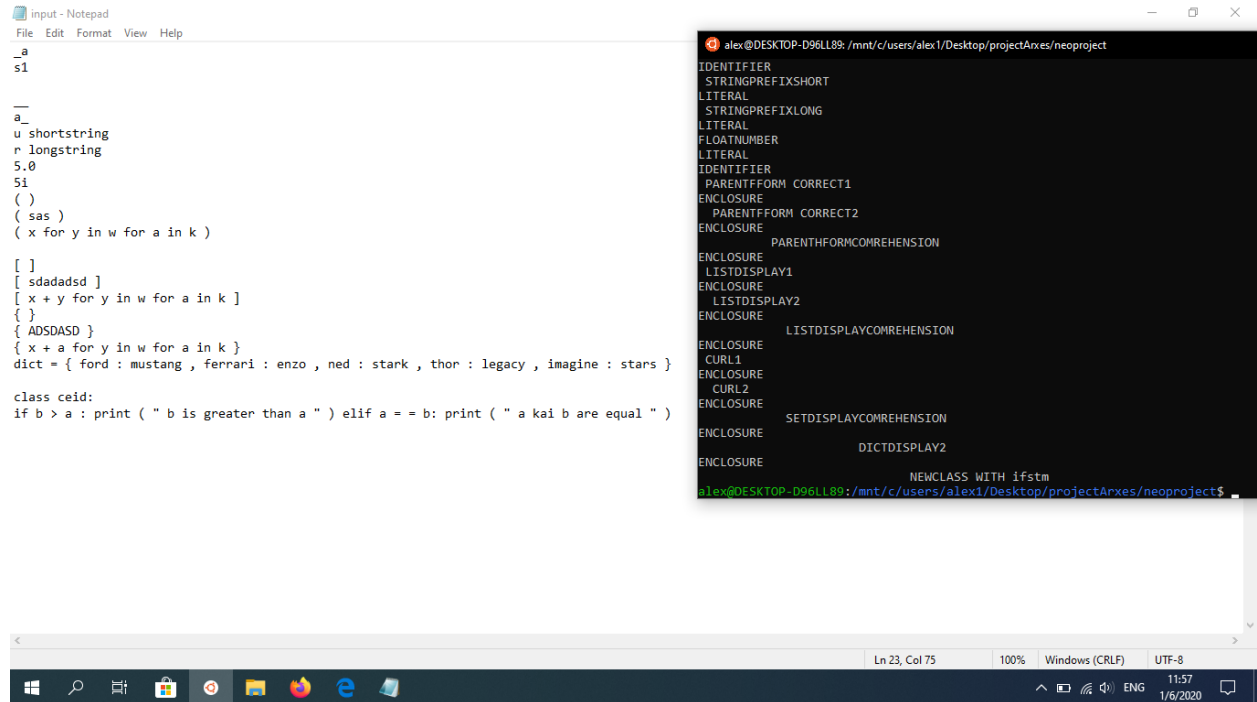
```
simple_statements: statements {printf("STATEMENT\n");}
```

```
|simple_statements statements {printf("SIMPLESTATEMENT\n");}
```

Ωστόσο η λογική μας πίσω από κανόνες τέτοιου τύπου ήταν η αναδρομικότητα του κανόνα.

## SCREENSHOT ΕΚΤΕΛΕΣΗΣ ΚΩΔΙΚΑ:

### IF STATEMENT:



The screenshot shows a Windows desktop with two windows. The 'input - Notepad' window contains the following Python code:

```
_a
s1

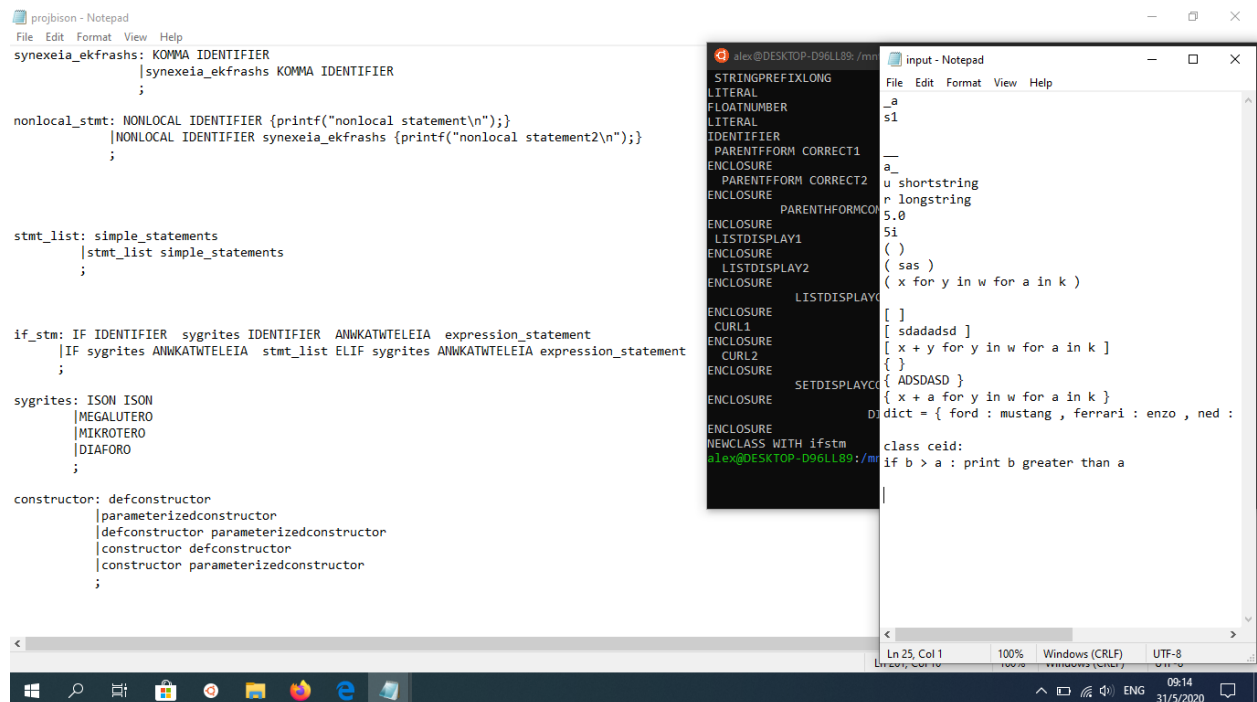
__
a_
u shortstring
r longstring
5.0
5i
( )
( sas )
( x for y in w for a in k )

[ ]
[ sdadadsd ]
[ x + y for y in w for a in k ]
{ }
{ ADSDASD }
{ x + a for y in w for a in k }
dict = { ford : mustang , ferrari : enzo , ned : stark , thor : legacy , imagine : stars }

class ceid:
if b > a : print ( " b is greater than a " ) elif a == b : print ( " a kai b are equal " )
```

The terminal window shows the execution output, which is a list of tokens and their corresponding Python grammar rules:

```
IDENTIFIER
STRINGPREFIXSHORT
LITERAL
STRINGPREFIXLONG
LITERAL
FLOATNUMBER
LITERAL
IDENTIFIER
PARENTFFORM CORRECT1
ENCLOSURE
PARENTFFORM CORRECT2
ENCLOSURE
PARENTFFORMCOMREHENSION
ENCLOSURE
LISTDISPLAY1
ENCLOSURE
LISTDISPLAY2
ENCLOSURE
LISTDISPLAYCOMREHENSION
ENCLOSURE
CURL1
ENCLOSURE
CURL2
ENCLOSURE
SETDISPLAYCOMREHENSION
ENCLOSURE
DICTDISPLAY2
ENCLOSURE
NEWCLASS WITH ifstm
alex@DESKTOP-D96LL89:/mnt/c/users/alex1/Desktop/projectArxes/neoproject$
```



The screenshot shows a Windows desktop with two windows. The 'projbison - Notepad' window contains the following Python code:

```
synexeia_ekfrashs: KOMMA IDENTIFIER
|synexeia_ekfrashs KOMMA IDENTIFIER
;

nonlocal_stmt: NONLOCAL IDENTIFIER {printf("nonlocal statement\n");}
|NONLOCAL IDENTIFIER synexeia_ekfrashs {printf("nonlocal statement2\n");}
;

stmt_list: simple_statements
|stmt_list simple_statements
;

if_stm: IF IDENTIFIER sygrites IDENTIFIER ANMKATWTELEIA expression_statement
|IF sygrites ANMKATWTELEIA stmt_list ELIF sygrites ANMKATWTELEIA expression_statement
;

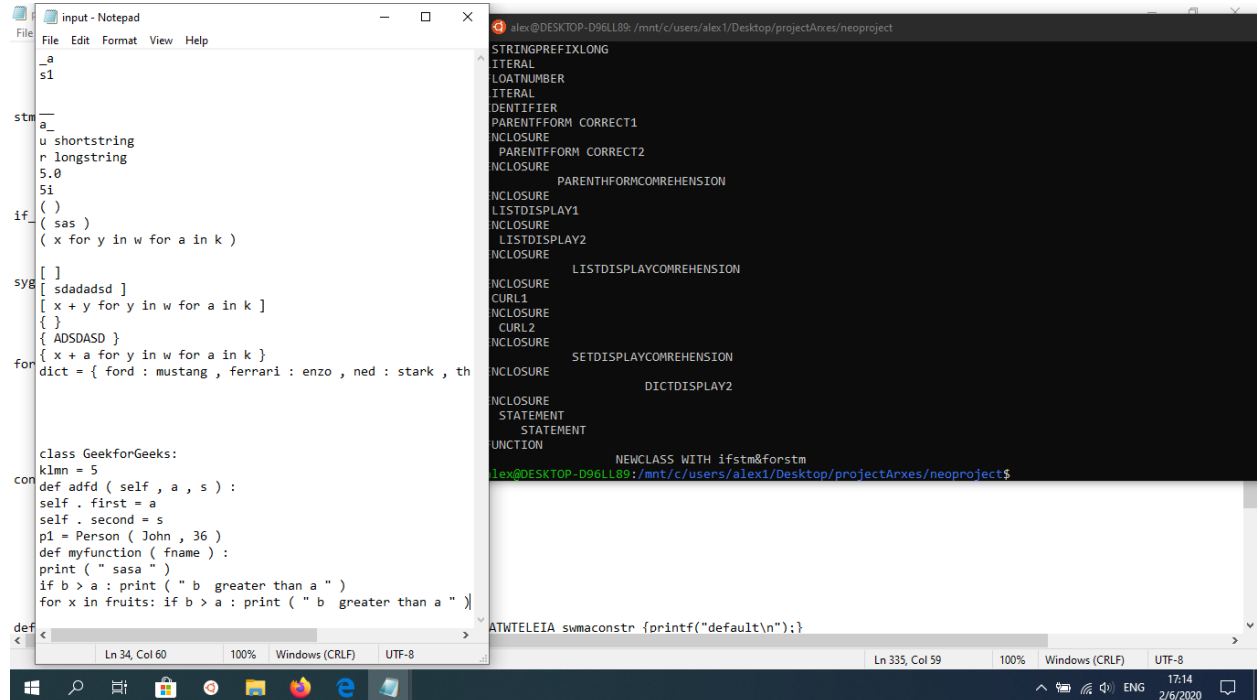
sygrites: ISON ISON
|MEGALUTERO
|MIKROTERO
|DIAFORO
;

constructor: defconstructor
|parameterizedconstructor
|defconstructor parameterizedconstructor
|constructor defconstructor
|constructor parameterizedconstructor
;
```

The terminal window shows the execution output, which is a list of tokens and their corresponding Python grammar rules:

```
STRINGPREFIXLONG
LITERAL
FLOATNUMBER
LITERAL
IDENTIFIER
PARENTFFORM CORRECT1
ENCLOSURE
PARENTFFORM CORRECT2
ENCLOSURE
PARENTFFORMCOMREHENSION
ENCLOSURE
LISTDISPLAY1
ENCLOSURE
LISTDISPLAY2
ENCLOSURE
LISTDISPLAYCOMREHENSION
ENCLOSURE
CURL1
ENCLOSURE
CURL2
ENCLOSURE
SETDISPLAYCOMREHENSION
ENCLOSURE
NEWCLASS WITH ifstm
alex@DESKTOP-D96LL89:/mnt/c/users/alex1/Desktop/projectArxes/neoproject$
```

## FOR STATEMENT(WITH IF STATEMENT INSIDE):



The screenshot shows a Windows desktop with two windows. The 'input - Notepad' window contains Python code. The terminal window shows the output of the C preprocessor for the same code, with various macros expanded.

```
File Edit Format View Help
_a
s1

stm
a_
u shortstring
r longstring
5.0
5i
( )
if ( sas )
( x for y in w for a in k )

[ ]
[ sdadadsd ]
[ x + y for y in w for a in k ]
{ }
{ ADSDASD }
{ x + a for y in w for a in k }
for dict = { ford : mustang , ferrari : enzo , ned : stark , th

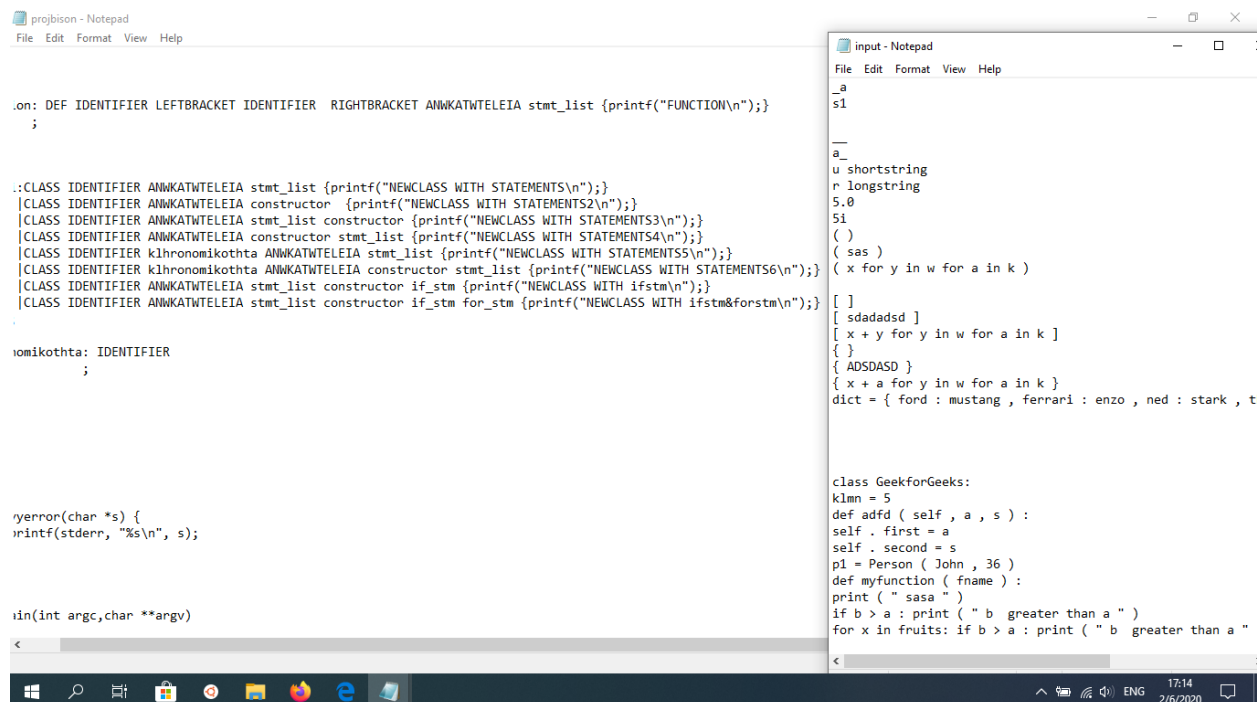
class GeekforGeeks:
    klmn = 5
def adfd ( self , a , s ) :
    self . first = a
    self . second = s
    p1 = Person ( John , 36 )
    def myfunction ( fname ) :
        print ( " sasa " )
        if b > a : print ( " b greater than a " )
        for x in fruits: if b > a : print ( " b greater than a " )
def

Ln 34, Col 60 100% Windows (CRLF) UTF-8
```

```
alex@DESKTOP-D96LL89: /mnt/c/users/alex1/Desktop/projectArxes/neoproject$
STRINGPREFIXLONG
ITERAL
LOATNUMBER
ITERAL
IDENTIFIER
PARENTFFORM CORRECT1
NCLOSURE
PARENTFFORM CORRECT2
NCLOSURE
PARENTFFORMCOMREHENSION
NCLOSURE
LISTDISPLAY1
NCLOSURE
LISTDISPLAY2
NCLOSURE
LISTDISPLAYCOMREHENSION
NCLOSURE
CURL1
NCLOSURE
CURL2
NCLOSURE
SETDISPLAYCOMREHENSION
NCLOSURE
DICTDISPLAY2
NCLOSURE
STATEMENT
STATEMENT
FUNCTION
NEWCLASS WITH ifstm&forstm
alex@DESKTOP-D96LL89: /mnt/c/users/alex1/Desktop/projectArxes/neoproject$

ATWTELEIA swmaconstr {printf("default\n");}
```

Ln 335, Col 59 100% Windows (CRLF) UTF-8



The screenshot shows a Windows desktop with two Notepad windows. The 'projbison - Notepad' window contains C preprocessor output for the Python code. The 'input - Notepad' window contains the original Python code.

```
File Edit Format View Help

.on: DEF IDENTIFIER LEFTBRACKET IDENTIFIER RIGHTBRACKET ANWKATWTELEIA stmt_list {printf("FUNCTION\n");}
;

.:CLASS IDENTIFIER ANWKATWTELEIA stmt_list {printf("NEWCLASS WITH STATEMENTS\n");}
|CLASS IDENTIFIER ANWKATWTELEIA constructor {printf("NEWCLASS WITH STATEMENTS2\n");}
|CLASS IDENTIFIER ANWKATWTELEIA stmt_list constructor {printf("NEWCLASS WITH STATEMENTS3\n");}
|CLASS IDENTIFIER ANWKATWTELEIA constructor stmt_list {printf("NEWCLASS WITH STATEMENTS4\n");}
|CLASS IDENTIFIER k1hronomikothta ANWKATWTELEIA stmt_list {printf("NEWCLASS WITH STATEMENTS5\n");}
|CLASS IDENTIFIER k1hronomikothta ANWKATWTELEIA constructor stmt_list {printf("NEWCLASS WITH STATEMENTS6\n");}
|CLASS IDENTIFIER ANWKATWTELEIA stmt_list constructor if_stm {printf("NEWCLASS WITH ifstm\n");}
|CLASS IDENTIFIER ANWKATWTELEIA stmt_list constructor if_stm for_stm {printf("NEWCLASS WITH ifstm&forstm\n");}

iomikothta: IDENTIFIER
;

yerror(char *s) {
    printf(stderr, "%s\n", s);

in(int argc, char **argv)
```

```
File Edit Format View Help
_a
s1

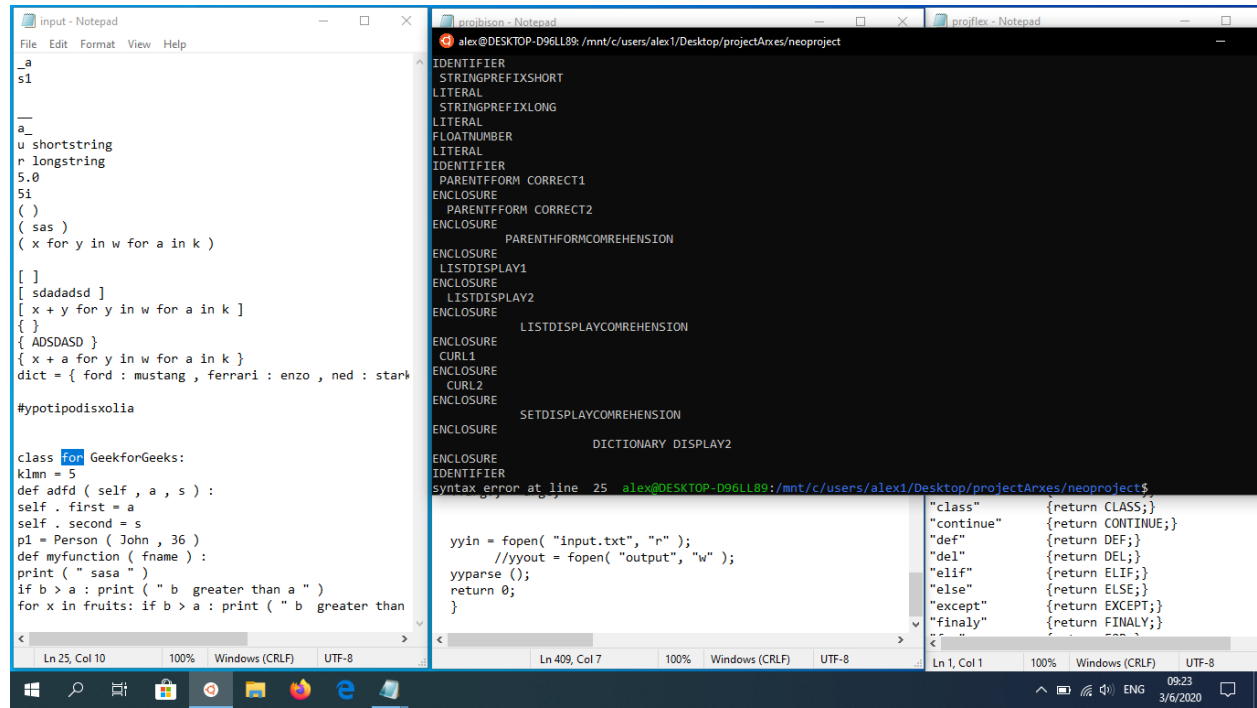
a_
u shortstring
r longstring
5.0
5i
( )
( sas )
( x for y in w for a in k )

[ ]
[ sdadadsd ]
[ x + y for y in w for a in k ]
{ }
{ ADSDASD }
{ x + a for y in w for a in k }
dict = { ford : mustang , ferrari : enzo , ned : stark , t

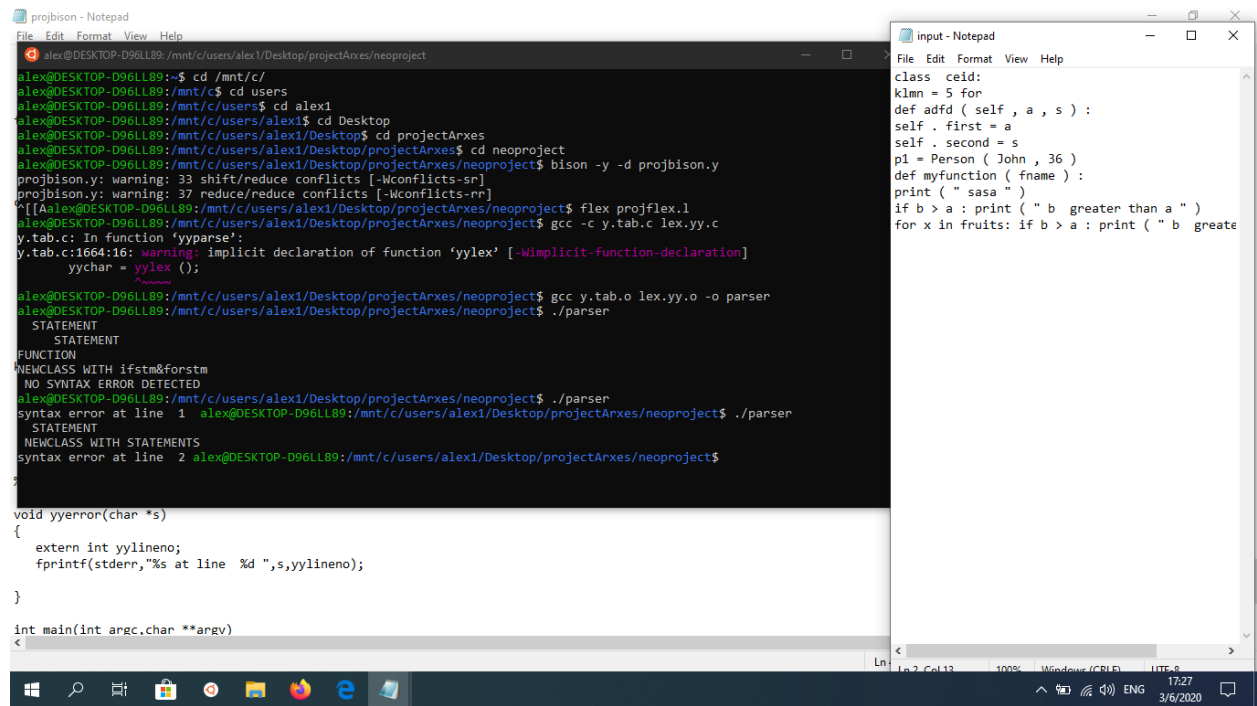
class GeekforGeeks:
    klmn = 5
def adfd ( self , a , s ) :
    self . first = a
    self . second = s
    p1 = Person ( John , 36 )
    def myfunction ( fname ) :
        print ( " sasa " )
        if b > a : print ( " b greater than a " )
        for x in fruits: if b > a : print ( " b greater than a "
```

## SYNTAX ERROR HANDLING:

Στο επόμενο screenshot φαίνεται τι γίνεται αν προσθέσουμε ένα for εκεί που δεν πρέπει και πως ο parser το χειρίζεται.

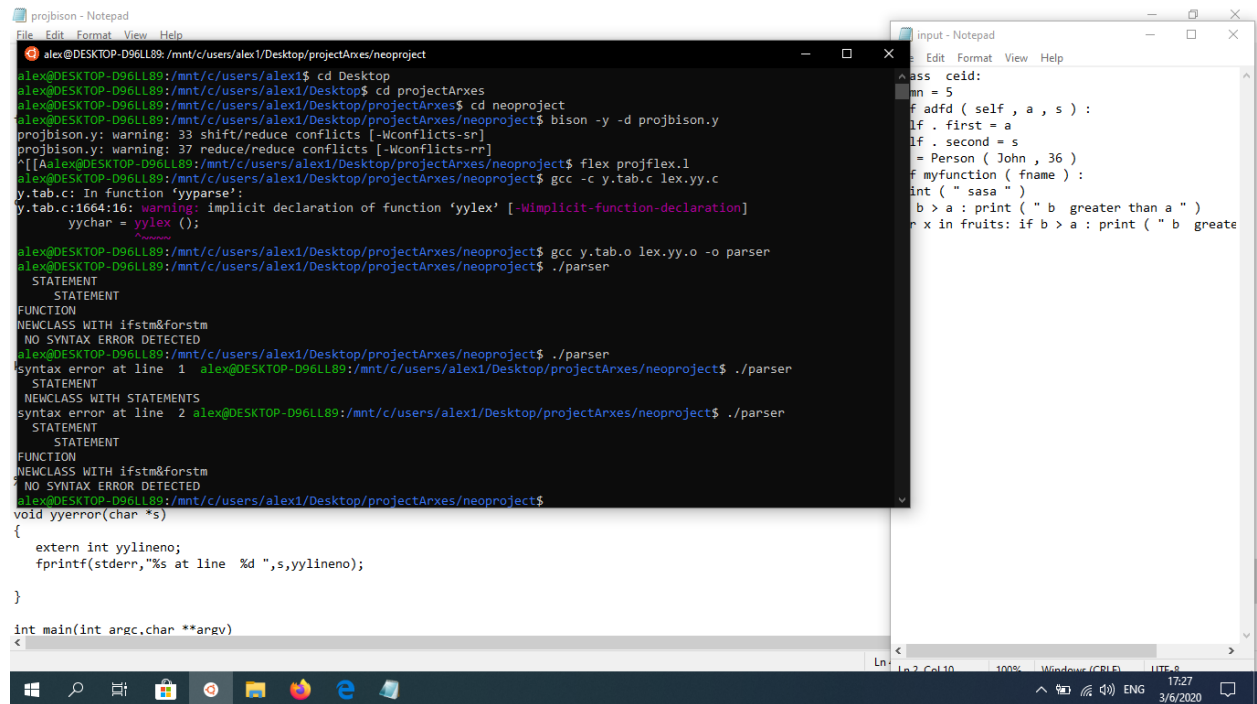


The screenshot shows three Notepad windows. The left window, titled 'input - Notepad', contains C code for a class 'GeekForGeeks' with methods 'adfd' and 'myfunction'. The middle window, titled 'projbison - Notepad', shows Bison grammar rules for identifiers, literals, and expressions, including a 'for' loop rule. The right window, titled 'projflex - Notepad', shows a C++ implementation of the parser, including a 'yyparse' function and a 'yylex' function that returns tokens like 'CLASS', 'CONTINUE', etc.



The screenshot shows a terminal window and a Notepad window. The terminal window, titled 'projbison - Notepad', shows the compilation and execution of the parser. It includes commands like 'cd /mnt/c/', 'cd users', 'cd alex1', 'cd Desktop', 'cd projectArxes', 'cd neoproject', 'bison -y -d projbison.y', 'flex projflex.l', 'gcc -c y.tab.c lex.yy.c', 'gcc y.tab.o lex.yy.o -o parser', and './parser'. The output shows warnings from bison and flex, and a syntax error at line 2. The Notepad window, titled 'input - Notepad', shows the C code for the 'GeekForGeeks' class, which is the same code as in the first screenshot.

Εδώ βλέπουμε τι γίνεται αν δεν υπάρχει κάποιο syntax error :



The screenshot shows two Notepad windows. The left window, titled 'projbison - Notepad', contains C code for a parser. It includes headers, a union for 'yylval', a 'yyparse' function, and a 'main' function. The code is being compiled and run in a terminal. The right window, titled 'input - Notepad', contains Python code defining a class 'ceid' with methods 'adfd' and 'myfunction'.

```
alex@DESKTOP-D96LL89:/mnt/c/users/alex1/Desktop/projectArxes/neoproject$ cd Desktop
alex@DESKTOP-D96LL89:/mnt/c/users/alex1/Desktop/projectArxes$ cd projectArxes
alex@DESKTOP-D96LL89:/mnt/c/users/alex1/Desktop/projectArxes$ cd neoproject
alex@DESKTOP-D96LL89:/mnt/c/users/alex1/Desktop/projectArxes/neoproject$ bison -y -d projbison.y
projbison.y: warning: 33 shift/reduce conflicts [-Wconflicts-sr]
projbison.y: warning: 37 reduce/reduce conflicts [-Wconflicts-rr]
alex@DESKTOP-D96LL89:/mnt/c/users/alex1/Desktop/projectArxes/neoproject$ flex projflex.l
alex@DESKTOP-D96LL89:/mnt/c/users/alex1/Desktop/projectArxes/neoproject$ gcc -c y.tab.c lex.yy.c
y.tab.c: In function 'yyparse':
y.tab.c:1664:16: warning: implicit declaration of function 'yylex' [-Wimplicit-function-declaration]
    yychar = yylex ();
                   ^
alex@DESKTOP-D96LL89:/mnt/c/users/alex1/Desktop/projectArxes/neoproject$ gcc y.tab.o lex.yy.o -o parser
alex@DESKTOP-D96LL89:/mnt/c/users/alex1/Desktop/projectArxes/neoproject$ ./parser
STATEMENT
STATEMENT
FUNCTION
NEWCLASS WITH ifstm&forstm
NO SYNTAX ERROR DETECTED
alex@DESKTOP-D96LL89:/mnt/c/users/alex1/Desktop/projectArxes/neoproject$ ./parser
syntax error at line 1 alex@DESKTOP-D96LL89:/mnt/c/users/alex1/Desktop/projectArxes/neoproject$ ./parser
STATEMENT
NEWCLASS WITH STATEMENTS
syntax error at line 2 alex@DESKTOP-D96LL89:/mnt/c/users/alex1/Desktop/projectArxes/neoproject$ ./parser
STATEMENT
STATEMENT
FUNCTION
NEWCLASS WITH ifstm&forstm
NO SYNTAX ERROR DETECTED
alex@DESKTOP-D96LL89:/mnt/c/users/alex1/Desktop/projectArxes/neoproject$
void yyerror(char *s)
{
    extern int yylineno;
    fprintf(stderr,"%s at line %d ",s,yylineno);
}

int main(int argc,char **argv)
{
    yyparse();
    return 0;
}
```

```
class ceid:
    def __init__(self, a, s):
        self.first = a
        self.second = s
    def Person(self, fname):
        print("sasa")
    def myfunction(self, fname):
        if b > a: print("b greater than a")
        for x in fruits: if b > a: print("b greater than a")
```

## **BNF ΥΠΟΣΥΝΟΛΟΥ ΤΗΣ PYTHON:**

atom ::= identifier

| literal

| enclosure

enclosure ::= parenth\_form

| list\_display

| generator\_expression

| dict\_display

| string\_conversion

| yield\_atom

literal ::= stringliteral

| integer

| longinteger

| floatnumber

| imagnumber

stringliteral ::= stringliteralpiece

| stringliteral stringliteralpiece

parenth\_form ::= "(" [expression\_list] ")"

list\_display ::= "[" [expression\_list | list\_comprehension] "]"

list\_comprehension ::= expression list\_for

generator\_expression ::= "(" expression genexpr\_for ")"

genexpr\_for ::= "for" target\_list "in" or\_test [genexpr\_iter]

dict\_display ::= "{" [key\_datum\_list] "}"

yield\_atom ::= "(" yield\_expression ")"

yield\_expression ::= "yield" [expression\_list]

primary ::= atom

| attributeref

| subscription

| slicing

| call

attributeref ::= primary "." identifier

subscription ::= primary "[" expression\_list "]"

slicing ::= simple\_slicing | extended\_slicing

simple\_slicing ::= primary "[" short\_slice "]"

extended\_slicing ::= primary "[" slice\_list "]"

slice\_list ::= slice\_item ("," slice\_item)\* [","]

slice\_item ::= expression | proper\_slice | ellipsis

proper\_slice ::= short\_slice | long\_slice



short\_slice ::= [lower\_bound] ":" [upper\_bound]

long\_slice ::= short\_slice ":" [stride]

lower\_bound ::= expression

upper\_bound ::= expression

stride ::= expression

ellipsis ::= "..."

call ::= primary "(" [argument\_list [",",  
| expression genexpr\_for] ")"

argument\_list ::= positional\_arguments [",", keyword\_arguments [",", "\*" expression] [",", "\*\*\*  
expression]

| keyword\_arguments [",", "\*" expression] [",", "\*\*\*" expression]

| "\*" expression [",", "\*\*\*" expression]

| "\*\*\*" expression

positional\_arguments ::= expression ("," expression)\*

keyword\_arguments ::= keyword\_item ("," keyword\_item)\*

keyword\_item ::= identifier "=" expression

power ::= primary ["\*\*" u\_expr]

u\_expr ::= power

| "-" u\_expr

| "+" u\_expr

| "~" u\_expr

m\_expr ::= u\_expr

| m\_expr "\*" u\_expr

| m\_expr "/" u\_expr

| m\_expr "/" u\_expr

| m\_expr "%" u\_expr

a\_expr ::= m\_expr

| a\_expr "+" m\_expr

| a\_expr "-" m\_expr

shift\_expr ::= a\_expr

| shift\_expr ( "<<" | ">>" ) a\_expr

and\_expr ::= shift\_expr | and\_expr "&" shift\_expr

xor\_expr ::= and\_expr

| xor\_expr "^" and\_expr

or\_expr ::= xor\_expr

| or\_expr "||" xor\_expr

comparison ::=

or\_expr ( comp\_operator or\_expr )\*

comp\_operator ::= "<" | ">" | "==" | ">=" | "<=" | "<>" | "!="

| "is" ["not"] | ["not"] "in"

expression ::= conditional\_expression | lambda\_form

old\_expression ::= or\_test | old\_lambda\_form

conditional\_expression ::= or\_test ["if" or\_test "else" expression]

or\_test ::= and\_test

| or\_test "or" and\_test

and\_test ::= not\_test

| and\_test "and" not\_test

not\_test ::= comparison

| "not" not\_test

$\text{lambda\_form} ::= \text{"lambda" } [\text{parameter\_list} ] \text{" : " expression}$

$\text{old\_lambda\_form} ::= \text{"lambda" } [\text{parameter\_list} ] \text{" : " old\_expression}$

$\text{expression\_list} ::= \text{expression ( " , " expression )}^* [ \text{" , " } ]$

$\text{simple\_stmt} ::= \text{expression\_stmt}$

| `assert_stmt`

| `assignment_stmt`

| `augmented_assignment_stmt`

| `pass_stmt`

| `del_stmt`

| `print_stmt`

| `return_stmt`

| `yield_stmt`

| `raise_stmt`

| `break_stmt`

| `continue_stmt`

| `import_stmt`

| `global_stmt`

| `exec_stmt`

$\text{expression\_stmt} ::= \text{expression\_list}$

assert\_stmt ::= "assert" expression ["," expression]

assignment\_stmt ::= (target\_list "=") (expression\_list | yield\_expression)

target\_list ::= target ("," target)\* [","]

target ::= identifier

| "(" target\_list ")"

| "[" target\_list "]"

| attributeref

| subscription

| slicing

augmented\_assignment\_stmt ::= target augop(expression\_list | yield\_expression)

augop ::= "+=" | "-=" | "\*=" | "/=" | "//=" | "%=" | "\*\*=" | ">>=" | "<<=" | "&=" | "^=" | "|="

pass\_stmt ::= "pass"

del\_stmt ::= "del" target\_list

print\_stmt ::= "print" ( [expression ("," expression)\* [","]] | ">>" expression [("," expression)+ [","]] )

return\_stmt ::= "return" [expression\_list]

yield\_stmt ::= yield\_expression

raise\_stmt ::= "raise" [expression ["," expression["," expression]]]

break\_stmt ::= "break"

continue\_stmt ::= "continue"

import\_stmt ::= "import" module ["as" name]( "," module ["as" name] )\*  
              | "from" relative\_module "import" identifier["as" name]( "," identifier ["as" name] )\*  
              | "from" relative\_module "import" "(" identifier ["as" name]( "," identifier ["as" name] )\* ["," ] ")"  
              | "from" module "import" "\*\*"

module ::= (identifier ".")\* identifier

relative\_module ::= "."\* module | "."+

name ::= identifier

global\_stmt ::= "global" identifier ("," identifier)\*

exec\_stmt ::= "exec" or\_expr["in" expression ["," expression]]

compound\_stmt ::= if\_stmt

              | while\_stmt

- | for\_stmt
- | try\_stmt
- | with\_stmt
- | funcdef
- | classdef

suite ::= stmt\_list NEWLINE

- | NEWLINE INDENT statement+ DEDENT

statement ::= stmt\_list NEWLINE

- | compound\_stmt

stmt\_list ::= simple\_stmt ( ";" simple\_stmt ) \* [ ";" ]

if\_stmt ::= "if" expression ":" suite ( "elif" expression ":" suite ) \* [ "else" ":" suite ]

while\_stmt ::= "while" expression ":" suite [ "else" ":" suite ]

for\_stmt ::= "for" target\_list "in" expression\_list ":" suite [ "else" ":" suite ]

try\_stmt ::= try1\_stmt

- | try2\_stmt

try1\_stmt ::= "try" ":" suite ( "except" [ expression [ "," target ] ] ":" suite ) + [ "else" ":" suite ] [ "finally" ":" suite ]

try2\_stmt ::= "try" ":" suite "finally" ":" suite

with\_stmt ::= "with" expression ["as" target] ":" suite

funcdef ::= [decorators] "def" funcname "(" [parameter\_list] ")" ":" suite

decorators ::= decorator+

decorator ::= "@" dotted\_name "(" [argument\_list [","]] ")" NEWLINE

dotted\_name ::=

    identifier ( "." identifier ) \*

parameter\_list ::=

    ( defparameter ", " ) \*

    ( "\*" identifier [ , "\*" identifier ]

      | "\*" identifier

      | defparameter [ ", " ] )

defparameter ::= parameter ["=" expression]

sublist ::= parameter ( ", " parameter ) \* [ ", " ]

parameter ::= identifier | "(" sublist ")"



funcname ::= identifier

classdef ::= "class" classname [inheritance] ":" suite

inheritance ::= "(" [expression\_list] ")"

classname ::= identifier

file\_input ::= (NEWLINE | statement)\*

interactive\_input ::= [stmt\_list] NEWLINE  
                          | compound\_stmt NEWLINE

eval\_input ::= expression\_list NEWLINE\*

input\_input ::= expression\_list NEWLINE

**BNF ΤΗΣ ΓΡΑΜΜΑΤΙΚΗΣ ΠΟΥ ΦΤΙΑΞΑΜΕ ΣΤΟ BISON:**

programm:

| atom programm

| classi programm

| function programm

;

atom::= IDENTIFIER

| literal

| enclosure

;

literal::= stringliteral

| INTEGER

| FLOATNUMBER

| IMAGINUMBER

;

stringliteral::= stringprefixShort

|stringprefixLong

;

stringprefixShort::= r SHORTSTRING

|u SHORTSTRING

|R SHORTSTRING

|U SHORTSTRING

|f SHORTSTRING

|F SHORTSTRING

|fr SHORTSTRING

|Fr SHORTSTRING

|fR SHORTSTRING

|FR SHORTSTRING

|rf SHORTSTRING

|rF SHORTSTRING

|RF SHORTSTRING

|RF SHORTSTRING

;

stringprefixLong::= r LONGSTRING

|u LONGSTRING

|R LONGSTRING

|U LONGSTRING

|f LONGSTRING

|F LONGSTRING

|fr LONGSTRING

|Fr LONGSTRING

|fR LONGSTRING

|FR LONGSTRING

|rf LONGSTRING

|rF LONGSTRING

|Rf LONGSTRING

|RF LONGSTRING

;

enclosure ::= parenth\_form

|list\_display

|set\_display

|dict\_display

//|string\_conversion

//|yield\_atom

;

parenth\_form ::= LEFTBRACKET RIGHTBRACKET

|LEFTBRACKET IDENTIFIER RIGHTBRACKET

|LEFTBRACKET comprehension RIGHTBRACKET

;

list\_display ::= LSQUAREBRACKET RSQUAREBRACKET

|LSQUAREBRACKET IDENTIFIER RSQUAREBRACKET

|LSQUAREBRACKET comprehension RSQUAREBRACKET

;

set\_display ::= CURLLEFT CURLRIGHT

|CURLLEFT IDENTIFIER CURLRIGHT

|CURLLEFT comprehension CURLRIGHT

;

dict\_display::=IDENTIFIER ISON CURLLEFT CURLRIGHT

|IDENTIFIER ISON CURLLEFT dictindex CURLRIGHT

;

dictindex::=IDENTIFIER ANWKATWTELEIA IDENTIFIER

|dictindex KOMMA IDENTIFIER ANWKATWTELEIA IDENTIFIER

;

comprehension::= IDENTIFIER FOR IDENTIFIER IN IDENTIFIER FOR IDENTIFIER IN IDENTIFIER

| IDENTIFIER PLUS IDENTIFIER FOR IDENTIFIER IN IDENTIFIER FOR IDENTIFIER IN IDENTIFIER

| IDENTIFIER MINUS IDENTIFIER FOR IDENTIFIER IN IDENTIFIER FOR IDENTIFIER IN IDENTIFIER

| IDENTIFIER MUL IDENTIFIER FOR IDENTIFIER IN IDENTIFIER FOR IDENTIFIER IN IDENTIFIER

| IDENTIFIER DIVISION IDENTIFIER FOR IDENTIFIER IN IDENTIFIER FOR IDENTIFIER IN IDENTIFIER

;

simple\_statements::= statements

|simple\_statements statements

|pass\_stm

|del\_stmt

|return\_stmt

|yield\_stmt

|raise\_stmt

|break\_stmt

|continue\_stmt

|import\_stmt

|global\_stmt

|nonlocal\_stmt

;

statements::= expression\_statement

|assignment\_statement

;

expression\_statement::= PRINT LEFTBRACKET DOUBLEQUOTE protash DOUBLEQUOTE RIGHTBRACKET

|PRINT FLOATNUMBER

;

protash::= IDENTIFIER

|protash IDENTIFIER

;

assignment\_statement::= IDENTIFIER ISON FLOATNUMBER

|IDENTIFIER ISON IMAGINUMBER

|IDENTIFIER ISON INTEGER

|IDENTIFIER ISON IDENTIFIER

|augmented\_assignment\_stmt

;

augmented\_assignment\_stmt ::= IDENTIFIER PLUS ISON IDENTIFIER

| IDENTIFIER MINUS ISON IDENTIFIER

;

pass\_stmt ::= PASS

;

del\_stmt ::= DEL IDENTIFIER

| DEL LEFTBRACKET IDENTIFIER RIGHTBRACKET

| DEL LSQUAREBRACKET IDENTIFIER RSQUAREBRACKET

;

return\_stmt ::= RETURN IDENTIFIER

;

yield\_stmt ::= YIELD IDENTIFIER

| YIELD FROM IDENTIFIER

;

raise\_stmt ::= RAISE IDENTIFIER FROM IDENTIFIER

| RAISE IDENTIFIER LEFTBRACKET DOUBLEQUOTE IDENTIFIER DOUBLEQUOTE RIGHTBRACKET FROM IDENTIFIER

;

break\_stmt ::= BREAK

;

continue\_stmt ::= CONTINUE

;

import\_stmt ::= IMPORT IDENTIFIER

| IMPORT module

| FROM module IMPORT IDENTIFIER

| IMPORT module AS IDENTIFIER

;

module ::= IDENTIFIER

| IDENTIFIER TELEIA IDENTIFIER

| module TELEIA IDENTIFIER

;

global\_stmt ::= GLOBAL IDENTIFIER

| GLOBAL IDENTIFIER synexeia\_ekfrashs

;

synexeia\_ekfrashs ::= KOMMA IDENTIFIER

| synexeia\_ekfrashs KOMMA IDENTIFIER

;

nonlocal\_stmt ::= NONLOCAL IDENTIFIER

| NONLOCAL IDENTIFIER synexeia\_ekfrashs

;



stmt\_list ::= simple\_statements

| stmt\_list simple\_statements

;

if\_stm ::= IF IDENTIFIER sygrites IDENTIFIER ANWKATWTELEIA expression\_statement

| IF IDENTIFIER sygrites IDENTIFIER ANWKATWTELEIA expression\_statement ELIF IDENTIFIER sygrites IDENTIFIER  
ANWKATWTELEIA expression\_statement

;

sygrites ::= ISON ISON

| MEGALUTERO

| MIKROTERO

| DIAFORO

;

for\_stm ::= FOR IDENTIFIER IN IDENTIFIER ANWKATWTELEIA expression\_statement

| FOR IDENTIFIER IN IDENTIFIER ANWKATWTELEIA if\_stm

;

constructor ::= defconstructor

| parameterizedconstructor

| defconstructor parameterizedconstructor

```
| constructor def constructor
| constructor parameterized constructor
;
```

```
def constructor ::= DEF INIT LEFTBRACKET IDENTIFIER RIGHTBRACKET ANWKATWTELEIA swmaconstr
;
```

```
parameterized constructor ::= DEF IDENTIFIER LEFTBRACKET IDENTIFIER RIGHTBRACKET swmaconstr
    | DEF IDENTIFIER LEFTBRACKET IDENTIFIER synexeia_ekfrashs RIGHTBRACKET ANWKATWTELEIA
    recursive_swmaconstr
    | DEF IDENTIFIER LEFTBRACKET IDENTIFIER synexeia_ekfrashs RIGHTBRACKET ANWKATWTELEIA
    recursive_swmaconstr object
    | DEF IDENTIFIER LEFTBRACKET IDENTIFIER synexeia_ekfrashs RIGHTBRACKET ANWKATWTELEIA
    recursive_swmaconstr function
    | DEF IDENTIFIER LEFTBRACKET IDENTIFIER synexeia_ekfrashs RIGHTBRACKET ANWKATWTELEIA
    recursive_swmaconstr object function
;
```

```
swmaconstr ::= module ISON DOUBLEQUOTE IDENTIFIER DOUBLEQUOTE
    | module ISON IDENTIFIER
;
```

```
recursive_swmaconstr ::= swmaconstr
                        | recursive_swmaconstr swmaconstr
                        ;
```

```
object ::= IDENTIFIER ISON IDENTIFIER LEFTBRACKET IDENTIFIER RIGHTBRACKET
          | IDENTIFIER ISON IDENTIFIER LEFTBRACKET IDENTIFIER synexeia_ekfrashs RIGHTBRACKET
          ;
```

```
function ::= DEF IDENTIFIER LEFTBRACKET IDENTIFIER RIGHTBRACKET ANWKATWTELEIA stmt_list
            ;
```

```
classi ::= CLASS IDENTIFIER ANWKATWTELEIA stmt_list
           | CLASS IDENTIFIER ANWKATWTELEIA constructor
           | CLASS IDENTIFIER ANWKATWTELEIA stmt_list constructor
```

|CLASS IDENTIFIER ANWKATWTELEIA constructor stmt\_list

|CLASS IDENTIFIER klhronomikothta ANWKATWTELEIA stmt\_list

|CLASS IDENTIFIER klhronomikothta ANWKATWTELEIA constructor stmt\_list

|CLASS IDENTIFIER ANWKATWTELEIA stmt\_list constructor if\_stm

|CLASS IDENTIFIER ANWKATWTELEIA stmt\_list constructor if\_stm for\_stm

;

klhronomikothta::= IDENTIFIER

;

**Τελικά αρχεία περιγραφής της γλώσσας, τα οποία δίνονται ως είσοδος στα Flex και Bison:**

- **Bison:**

```
%{  
#include <stdio.h>  
#include <string.h>  
#include <stdio.h>  
void yyerror(char *);  
extern FILE *yyin;  
extern FILE *yyout;  
  
char tmplist_expression[100];  
char list_expression[100];  
  
%}  
  
%locations  
  
%union  
{  
    char name;  
}  
  
%token FALSE  
%token TRUE  
%token AND  
%token ASSERT  
%token ASYNC  
%token AWAIT  
%token BREAK  
%token CLASS  
%token CONTINUE  
%token DEF  
%token DEL  
%token ELIF  
%token ELSE  
%token EXCEPT  
%token FINALY  
%token FOR  
%token FROM  
%token GLOBAL  
%token IF
```

%token IMPORT  
%token IN  
%token IS  
%token LAMBDA  
%token NONLOCAL  
%token NOT  
%token NONE  
%token OR  
%token PASS  
%token RAISE  
%token RETURN  
%token TRY  
%token WHILE  
%token WITH  
%token YIELD  
%token LBRACKET  
%token RBRACKET  
%token LBRACE  
%token RBRACE  
%token SEMICOLON  
%token LOWERCASE  
%token <ystr1> NUM  
%token LETTER  
%token KATWPAVLA  
%token QUOTE  
%token DOUBLEQUOTE  
%token DIGIT  
%token r  
%token u  
%token R  
%token U  
%token f  
%token F  
%token fr  
%token Fr  
%token fR  
%token FR  
%token rf  
%token Rf  
%token rF  
%token RF  
%token SHORTSTRING  
%token LONGSTRING  
%token IDENTIFIER  
%token INTEGER  
%token FLOATNUMBER  
%token IMAGINUMBER  
%token LEFTBRACKET  
%token RIGHTBRACKET  
%token LSQUAREBRACKET  
%token RSQUAREBRACKET

%token CURLLEFT  
%token CURLRIGHT  
%token PLUS  
%token MUL  
%token DIVISION  
%token MINUS  
%token ISON  
%token KOMMA  
%token ANWKATWTELEIA  
%token PRINT  
%token SINISON  
%token PLINISON  
%token TELEIA  
%token AS  
%token INIT  
%token MEGALUTERO  
%token MIKROTERO  
%token DIAFORO

%%

programm:

| atom programm  
| classi programm  
| function programm  
;

atom: IDENTIFIER {printf("IDENTIFIER\n");}  
| literal {printf("LITERAL\n");}  
| enclosure {printf("ENCLOSURE\n");}  
;

literal: stringliteral

| INTEGER  
| FLOATNUMBER {printf("FLOATNUMBER\n");}  
| IMAGINUMBER {printf("IMAGINERYNUMBER\n");}  
;

stringliteral: stringprefixShort {printf("STRINGPREFIXSHORT\n");}  
| stringprefixLong {printf("STRINGPREFIXLONG\n");}

;

stringprefixShort: r SHORTSTRING

|u SHORTSTRING  
|R SHORTSTRING  
|U SHORTSTRING  
|f SHORTSTRING  
|F SHORTSTRING  
|fr SHORTSTRING  
|Fr SHORTSTRING  
|fR SHORTSTRING  
|FR SHORTSTRING  
|rf SHORTSTRING  
|rF SHORTSTRING  
|Rf SHORTSTRING  
|RF SHORTSTRING

;

stringprefixLong: r LONGSTRING

|u LONGSTRING  
|R LONGSTRING  
|U LONGSTRING  
|f LONGSTRING  
|F LONGSTRING  
|fr LONGSTRING  
|Fr LONGSTRING  
|fR LONGSTRING  
|FR LONGSTRING  
|rf LONGSTRING  
|rF LONGSTRING  
|Rf LONGSTRING  
|RF LONGSTRING

;

enclosure: parenth\_form

|list\_display  
|set\_display  
|dict\_display  
//|string\_conversion  
//|yield\_atom

;

parenth\_form: LEFTBRACKET RIGHTBRACKET {printf("PARENTFFORM CORRECT1\n");}

|LEFTBRACKET IDENTIFIER RIGHTBRACKET {printf("PARENTFFORM CORRECT2\n");}

|LEFTBRACKET comprehension RIGHTBRACKET {printf("PARENTHFORMCOMREHENSION\n");}

;

list\_display: LSQUAREBRACKET RSQUAREBRACKET {printf("LISTDISPLAY1\n");}



```

|LSQUAREBRACKET IDENTIFIER RSQUAREBRACKET {printf("LISTDISPLAY2\n");}
|LSQUAREBRACKET comprehension RSQUAREBRACKET {printf("LISTDISPLAYCOMREHENSION\n");}
;

set_display: CURLLEFT CURLRIGHT {printf("CURL1\n");}
|CURLLEFT IDENTIFIER CURLRIGHT {printf("CURL2\n");}
|CURLLEFT comprehension CURLRIGHT {printf("SETDISPLAYCOMREHENSION\n");}
;

dict_display:IDENTIFIER ISON CURLLEFT CURLRIGHT {printf("DICTIONARY DISPLAY\n");}
|IDENTIFIER ISON CURLLEFT dictindex CURLRIGHT {printf("DICTIONARY DISPLAY2\n");}
;

dictindex:IDENTIFIER ANWKATWTELEIA IDENTIFIER
|dictindex KOMMA IDENTIFIER ANWKATWTELEIA IDENTIFIER
;

comprehension: IDENTIFIER FOR IDENTIFIER IN IDENTIFIER FOR IDENTIFIER IN IDENTIFIER
|IDENTIFIER PLUS IDENTIFIER FOR IDENTIFIER IN IDENTIFIER FOR IDENTIFIER IN IDENTIFIER
|IDENTIFIER MINUS IDENTIFIER FOR IDENTIFIER IN IDENTIFIER FOR IDENTIFIER IN IDENTIFIER
|IDENTIFIER MUL IDENTIFIER FOR IDENTIFIER IN IDENTIFIER FOR IDENTIFIER IN IDENTIFIER
|IDENTIFIER DIVISION IDENTIFIER FOR IDENTIFIER IN IDENTIFIER FOR IDENTIFIER IN IDENTIFIER
;

simple_statements: statements {printf("STATEMENT\n");}
|simple_statements statements {printf("SIMPLESTATMENT\n");}
|pass_stm
|del_stmt
|return_stmt
|yield_stmt
|raise_stmt
|break_stmt
|continue_stmt
|import_stmt
|global_stmt
|nonlocal_stmt
;

statements: expression_statement
|assignment_statment
;

expression_statement: PRINT LEFTBRACKET DOUBLEQUOTE protash DOUBLEQUOTE RIGHTBRACKET
|PRINT FLOATNUMBER
;

```

```
protash: IDENTIFIER
    |protash IDENTIFIER
    ;
```

```
assignment_statment: IDENTIFIER ISON FLOATNUMBER
    |IDENTIFIER ISON IMAGINUMBER
    |IDENTIFIER ISON INTEGER
    |IDENTIFIER ISON IDENTIFIER
    |augmented_assignment_stmt {printf("AUGUMENTEDSTATMENT\n");}
    ;
```

```
augmented_assignment_stmt: IDENTIFIER PLUS ISON IDENTIFIER
    |IDENTIFIER MINUS ISON IDENTIFIER
    ;
```

```
pass_stm: PASS
    ;
```

```
del_stmt: DEL IDENTIFIER
    |DEL LEFTBRACKET IDENTIFIER RIGHTBRACKET
    |DEL LSQUAREBRACKET IDENTIFIER RSQUAREBRACKET
    ;
```

```
return_stmt: RETURN IDENTIFIER {printf("return statement\n");}
    ;
```

```
yield_stmt: YIELD IDENTIFIER {printf("yield statement\n");}
    |YIELD FROM IDENTIFIER
    ;
```

```
raise_stmt: RAISE IDENTIFIER FROM IDENTIFIER {printf("raisestm\n");}
    |RAISE IDENTIFIER LEFTBRACKET DOUBLEQUOTE IDENTIFIER DOUBLEQUOTE RIGHTBRACKET FROM IDENTIFIER
    {printf("raisestm2\n");}
    ;
```

```
break_stmt: BREAK
    ;
```

```
continue_stmt: CONTINUE
    ;
```

```
import_stmt: IMPORT IDENTIFIER {printf("IMPORT IDENTIFIER\n");}
    |IMPORT module {printf("IMPORT MODULE\n");}
    |FROM module IMPORT IDENTIFIER {printf("FROM IMPORT MODULE\n");}
```

```

|IMPORT module AS IDENTIFIER {printf(" IMPORT MODULE AS\n");}
;

module: IDENTIFIER
|IDENTIFIER TELEIA IDENTIFIER
|module TELEIA IDENTIFIER
;

global_stmt: GLOBAL IDENTIFIER {printf("global statement\n");}
|GLOBAL IDENTIFIER synexeia_ekfrashs {printf("global statement2\n");}
;

synexeia_ekfrashs: KOMMA IDENTIFIER
|synexeia_ekfrashs KOMMA IDENTIFIER
;

nonlocal_stmt: NONLOCAL IDENTIFIER {printf("nonlocal statement\n");}
|NONLOCAL IDENTIFIER synexeia_ekfrashs {printf("nonlocal statement2\n");}
;

stmt_list: simple_statements
|stmt_list simple_statements
;

if_stm: IF IDENTIFIER sygrites IDENTIFIER ANWKATWTELEIA expression_statement
|IF IDENTIFIER sygrites IDENTIFIER ANWKATWTELEIA expression_statement ELIF IDENTIFIER sygrites IDENTIFIER
ANWKATWTELEIA expression_statement
;

sygrites: ISON ISON
|MEGALUTERO
|MIKROTERO
|DIAFORO
;

for_stm: FOR IDENTIFIER IN IDENTIFIER ANWKATWTELEIA expression_statement
|FOR IDENTIFIER IN IDENTIFIER ANWKATWTELEIA if_stm
;

constructor: defconstructor
|parameterizedconstructor
|defconstructor parameterizedconstructor
|constructor defconstructor

```

```
|constructor parameterizedconstructor  
;
```

```
defconstructor: DEF INIT LEFTBRACKET IDENTIFIER RIGHTBRACKET ANWKATWTELEIA swmaconstr  
{printf("default\n");}  
;
```

```
parameterizedconstructor: DEF IDENTIFIER LEFTBRACKET IDENTIFIER RIGHTBRACKET swmaconstr  
    | DEF IDENTIFIER LEFTBRACKET IDENTIFIER synexeia_ekfrashs RIGHTBRACKET ANWKATWTELEIA  
recursive_swmaconstr {printf("parameterizedconstructor2\n");}  
    | DEF IDENTIFIER LEFTBRACKET IDENTIFIER synexeia_ekfrashs RIGHTBRACKET ANWKATWTELEIA  
recursive_swmaconstr object {printf("parameterizedconstructor with object creation\n");}  
    | DEF IDENTIFIER LEFTBRACKET IDENTIFIER synexeia_ekfrashs RIGHTBRACKET ANWKATWTELEIA  
recursive_swmaconstr function  
    | DEF IDENTIFIER LEFTBRACKET IDENTIFIER synexeia_ekfrashs RIGHTBRACKET ANWKATWTELEIA  
recursive_swmaconstr object function  
;
```

```
swmaconstr: module ISON DOUBLEQUOTE IDENTIFIER DOUBLEQUOTE  
    | module ISON IDENTIFIER  
;
```

```
recursive_swmaconstr: swmaconstr  
    | recursive_swmaconstr swmaconstr  
;
```

```
object: IDENTIFIER ISON IDENTIFIER LEFTBRACKET IDENTIFIER RIGHTBRACKET  
    | IDENTIFIER ISON IDENTIFIER LEFTBRACKET IDENTIFIER synexeia_ekfrashs RIGHTBRACKET  
;
```

```
function: DEF IDENTIFIER LEFTBRACKET IDENTIFIER RIGHTBRACKET ANWKATWTELEIA stmt_list
{printf("FUNCTION\n");}
;
```

```
classi:CLASS IDENTIFIER ANWKATWTELEIA stmt_list {printf("NEWCLASS WITH STATEMENTS\n");}
|CLASS IDENTIFIER ANWKATWTELEIA constructor {printf("NEWCLASS WITH STATEMENTS2\n");}
|CLASS IDENTIFIER ANWKATWTELEIA stmt_list constructor {printf("NEWCLASS WITH STATEMENTS3\n");}
|CLASS IDENTIFIER ANWKATWTELEIA constructor stmt_list {printf("NEWCLASS WITH STATEMENTS4\n");}
|CLASS IDENTIFIER klhronomikothta ANWKATWTELEIA stmt_list {printf("NEWCLASS WITH STATEMENTS5\n");}
|CLASS IDENTIFIER klhronomikothta ANWKATWTELEIA constructor stmt_list {printf("NEWCLASS WITH
STATEMENTS6\n");}
|CLASS IDENTIFIER ANWKATWTELEIA stmt_list constructor if_stm {printf("NEWCLASS WITH ifstm\n");}
|CLASS IDENTIFIER ANWKATWTELEIA stmt_list constructor if_stm for_stm {printf("NEWCLASS WITH
ifstm&forstm\n NO SYNTAX ERROR DETECTED\n");}
;
```

```
klhronomikothta: IDENTIFIER
;
```

```
%%
```

```
void yyerror(char *s)
{
    extern int yylineno;
    fprintf(stderr,"%s at line %d ",s,yylineno);
}

```

```
int main(int argc,char **argv)
{

```

```
    ++argv; --argc;
```

```
    yyin = fopen( "input.txt", "r" );
        //yyout = fopen( "output", "w" );
    yyparse ();
    return 0;
}
```

- **Flex:**  
%{

```
#include "y.tab.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void yyerror (char *s);
int yylex();
```

```
%}
```

```
%option noyywrap
%option yylineno
```

```
integer [0-9]
%%
```

```
"false"      {return FALSE;}
"none"       {return NONE;}
"true"       {return TRUE;}
"and"        {return AND;}
"assert"     {return ASSERT;}
"async"      {return ASYNC;}
"await"      {return AWAIT;}
"break"      {return BREAK;}
"class"      {return CLASS;}
"continue"   {return CONTINUE;}
"def"        {return DEF;}
"del"        {return DEL;}
"elif"       {return ELIF;}
"else"       {return ELSE;}
"except"     {return EXCEPT;}
"finally"    {return FINALLY;}
"for"        {return FOR;}
"from"       {return FROM;}
"global"     {return GLOBAL;}
"if"         {return IF;}
"import"     {return IMPORT;}
"in"         {return IN;}

"lambda" {return LAMBDA;}
"nonlocal" {return NONLOCAL;}
"not"     {return NOT;}
"or"      {return OR;}
"pass"    {return PASS;}
"raise"   {return RAISE;}
"return"  {return RETURN;}
"try"     {return TRY;}
"while"   {return WHILE;}
"with"    {return WITH;}
"yield"   {return YIELD;}
```

```

"_"      {return KATWPAVLA;}
""       {return QUOTE;}
"\\"     {return DOUBLEQUOTE;}
"r"      {return r;}
"u"      {return u;}
"R"      {return R;}
"U"      {return U;}
"f"      {return f;}
"F"      {return F;}
"fr"     {return fr;}
"Fr"     {return Fr;}
"fR"     {return fR;}
"FR"     {return FR;}
"rf"     {return rf;}
"Rr"     {return Rf;}
"rF"     {return rF;}
"RF"     {return RF;}
"shortstring" {return SHORTSTRING;}
"longstring"  {return LONGSTRING;}
"("       {return LEFTBRACKET;}
")"       {return RIGHTBRACKET;}
"["       {return LSQUAREBRACKET;}
"]"       {return RSQUAREBRACKET;}
"{"       {return CURLLEFT;}
"}"       {return CURLRIGHT;}
"+"       {return PLUS;}
"*"       {return MUL;}
"/"       {return DIVISION;}
"-"       {return MINUS;}
"="       {return ISON;}
","       {return KOMMA;}
":"       {return ANWKATWTELEIA;}
"print"   {return PRINT;}
"+="      {return SINISON;}
"-= "     {return PLINISON;}
"."       {return TELEIA;}
"as"      {return AS;}
"init"    {return INIT;}
">"      {return MEGALUTERO;}
"<"      {return MIKROTERO;}
"diaforo" {return DIAFORO;}

```

```

\n      {}

```

```

{integer}+          {return INTEGER;}
[_a-zA-z0-9_]*      {return IDENTIFIER;}
[+-]?([0-9]*\.[0-9]+|[0-9]+) {return FLOATNUMBER;}
[+-]?([0-9][i,j]*)  {return IMAGINUMBER;}
.;
%%

```

- **Input:**  
programm  
u shortstring



```

r longstring
5.0
5i
( )
( sas )
( x for y in w for a in k )

[ ]
[ sdadadsd ]
[ x + y for y in w for a in k ]
{ }
{ ADSDASD }
{ x + a for y in w for a in k }
dict = { ford : mustang , ferrari : enzo , ned : stark , thor : legacy , imagine : stars }
#randomsxolia

class ceid:
    klmn = 5
    def adfd ( self , a , s ) :
        self . first = a
        self . second = s
    p1 = Person ( John , 36 )
    def myfunction ( fname ) :
        print ( " sasa " )
    if b > a : print ( " b greater than a " )
    for x in fruits: if b > a : print ( " b greater than a " )

```