ΑΡΧΕΣ ΓΛΩΣΣΩΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΚΑΙ ΜΕΤΑΦΡΑΣΤΩΝ



1. Περιγραφή της γραμματικής της γλώσσας σε BNF:

<input> ::=

<newlines>

| <statements>

<suite>::=

<stmt\_list> NEWLINE

| NEWLINE INDENT <statements> DEDENT

<statement>:

<stmt\_list> NEWLINE

| <compound\_stmt>

<statements>::=

<statement>

| <statements> <statement>

<stmt\_list>::=

<simple\_stmt>

| <simple\_stmt> ';'

| <simple\_stmt> <simple\_stmts>

| <simple\_stmt> <simple\_stmts> ';'

<simple\_stmts>::=

';' <simple\_stmt>

| <simple\_stmts> ';' <simple\_stmt>

<newlines>::=

NEWLINE

| <newlines> NEWLINE

<simple\_stmt>::=

<expression\_stmt>

| <assignment\_stmt>

| <print\_stmt>

| <return\_stmt>

| <break\_stmt>

| <import\_stmt>

<expression\_stmt>::=

<expression\_list>

<assignment\_stmt>::=

<assignment\_stmt\_targer\_list> <expression\_list>

<assignment\_stmt\_targer\_list>::=

<target\_list> '='

| <assignment\_stmt\_targer\_list> <target\_list> '='

<print\_stmt>::=

PRINT

| PRINT <expression>

| PRINT <expression> ','

| PRINT <expression> <expressions>

| PRINT <expression> <expressions> ','

| PRINT RIGHT\_OP <expression>

| PRINT RIGHT\_OP <expression> <expressions>

| PRINT RIGHT\_OP <expression> <expressions> ','

<return\_stmt>::=

RETURN

| RETURN <expression\_list>

<break\_stmt>::=

BREAK

<compound\_stmt>::=

<if\_stmt>

| <for\_stmt>

| <funcdef>

| <classdef>

<if\_stmt>::=

IF <expression> ':' <suite>

| IF <expression> ':' <suite> ELSE ':' <suite>

| IF <expression> ':' <suite> <elif\_stmt>

| IF <expression> ':' <suite> <elif\_stmt> ELSE ':' <suite>

<elif\_stmt>::=

ELIF <expression> ':' <suite>

<elif\_stmt> ELIF <expression> ':' <suite>

<for\_stmt>:

FOR <target\_list> IN <expression\_list> ':' <suite>

| FOR <target\_list> IN <expression\_list> ':' <suite> ELSE ':' <suite>

<funcdef>::=

DEF <funcname> '(' ')' ':' <suite>

| <decorators> DEF <funcname> '(' ')' ':' <suite>

| DEF <funcname> '(' <parameter\_list> ')' ':' <suite>

| <decorators> DEF <funcname> '(' <parameter\_list> ')' ':' <suite>

<decorators>::=

<decorator>

| <decorators> <decorator>

<decorator>::=

'@' <dotted\_name> NEWLINE

| '@' <dotted\_name> '(' ')' NEWLINE

| '@' <dotted\_name> '(' <argument\_list> ')' NEWLINE

| '@' <dotted\_name> '(' <argument\_list> ',' ')' NEWLINE

<dotted\_name>::=

<identifier>

| <identifier> <dot\_identifiers>

<dot\_identifiers>::=

'.' <identifier>

| <dot\_identifiers> '.' <identifier>

<parameter\_list>::=

STAR <identifier>

| STAR <identifier> ',' DOUBLESTAR <identifier>

| DOUBLESTAR <identifier>

| <defparameter>

| <defparameter>','

| <defparameters> STAR <identifier>

| <defparameters> STAR <identifier> ',' DOUBLESTAR <identifier>

| <defparameters> DOUBLESTAR <identifier>

| <defparameters> defparameter

| <defparameters> def<parameter >','

<defparameter>::=

<parameter>

| <parameter >'=' <expression>

<defparameters>::=

defparameter >','

| <defparameters> <defparameter >','

<sublist>::=

<parameter>

| <parameter >','

| <parameter ><parameters>

| <parameter ><parameters> ','

<parameter>::=

<identifier>

| '(' <sublist> ')'

<parameters>::=

',' <parameter >

| <parameters> ',' <parameter>

<funcname>::=

identifier

<classdef>::=

CLASS <classname> ':' <suite>

| CLASS <classname> <inheritance> ':' <suite>

<inheritance>::=

'(' ')'

| '(' <expression\_list> ')'

<classname>::=

<identifier>

<suite>::=

<stmt\_list> NEWLINE

| NEWLINE INDENT <statements> DEDENT

<import\_stmt>::=

IMPORT< module >

| IMPORT< module >AS <name>

| IMPORT< module ><modules>

| IMPORT< module >AS <name> modules

| FROM <relative\_module> IMPORT <identifier>

| FROM <relative\_module> IMPORT <identifier> AS <name>

| FROM <relative\_module> IMPORT <identifier> <import\_stmt\_identifiers>

| FROM <relative\_module> IMPORT <identifier> AS <name> <import\_stmt\_identifiers>

| FROM <relative\_module> IMPORT '(' <identifier> ')'

| FROM <relative\_module> IMPORT '(' <identifier> AS <name> ')'

| FROM <relative\_module> IMPORT '(' <identifier> <import\_stmt\_identifiers> ')'

| FROM <relative\_module> IMPORT '(' <identifier> AS <name> <import\_stmt\_identifiers> ')'

| FROM <relative\_module> IMPORT '(' <identifier> ',' ')'

| FROM <relative\_module> IMPORT '(' <identifier> AS <name> ',' ')'

| FROM <relative\_module> IMPORT '(' <identifier> <import\_stmt\_identifiers> ',' ')'

| FROM <relative\_module> IMPORT '(' <identifier> AS <name> <import\_stmt\_identifiers> ',' ')'

| FROM< module >IMPORT STAR

<module>::=

<identifier>

|< module >'.' <module>

<relative\_module>::=

<module>

| <dot\_modules>

| <dots>

<dot\_modules>::=

'.' <module>

| <dot\_modules> '.' module

<dots>::=

'.'

| <dots> '.'

<modules>::=

',' <module>

| ','< module >AS <name>

| <modules> ',' <module>

| <modules> ','< module >AS <name>

<import\_stmt\_identifiers>::=

',' <identifier>

| ',' <identifier> AS <name>

| <import\_stmt\_identifiers> ',' <identifier>

| <import\_stmt\_identifiers> ',' <identifier> AS <name>

<name>::=

<identifier>

<primary>::=

<atom>

| <attributeref>

| <call>

<call>::=

<primary> '(' ')'

| <primary> '(' <argument\_list> ')'

| <primary> '(' <argument\_list> ',' ')'

<argument\_list>::=

<positional\_arguments>

| <positional\_arguments> ',' <keyword\_arguments>

| <positional\_arguments> ',' STAR <expression>

| <positional\_arguments> ',' DOUBLESTAR <expression>

| <positional\_arguments> ',' <keyword\_arguments> ',' STAR <expression>

| <positional\_arguments> ',' <keyword\_arguments> ',' DOUBLESTAR <expression>

| <positional\_arguments> ',' STAR <expression> ',' DOUBLESTAR <expression>

| <positional\_arguments> ',' <keyword\_arguments> ',' STAR <expression> ',' DOUBLESTAR <expression>

| <keyword\_arguments>

| <keyword\_arguments> ',' STAR <expression>

| <keyword\_arguments> ',' DOUBLESTAR <expression>

| <keyword\_arguments> ',' STAR <expression> ',' DOUBLESTAR <expression>

| STAR <expression>

| STAR <expression> ',' DOUBLESTAR <expression>

| DOUBLESTAR <expression>

<positional\_arguments>::=

<expression>

| <expression> <expressions>

<keyword\_arguments>::=

<keyword\_item>

| <keyword\_item> <keyword\_items>

<keyword\_item>::=

<identifier> '=' <expression>

<keyword\_items>::=

',' <keyword\_item>

| <keyword\_items> ',' <keyword\_item>

<expression\_list>::=

<expression>

| <expression> ','

| <expression> <expressions>

| <expression> <expressions> ','

<expressions>::=

',' <expression>

| <expressions> ',' <expression>

<expression>::=

<conditional\_expression>

| <lambda\_form>

<conditional\_expression>::=

<or\_test>

| <or\_test> IF <or\_test> ELSE expression

<power>::=

<primary>

| DOUBLESTAR <u\_expr>

<u\_expr>::=

power

| '-' <u\_expr>

| '+' <u\_expr>

| '~' <u\_expr>

<m\_expr>::=

<u\_expr>

| <m\_expr> STAR <u\_expr>

| <m\_expr> DOUBLESLASH <u\_expr>

| <m\_expr> SLASH <u\_expr>

| <m\_expr> '%' <u\_expr>

<a\_expr>::=

<m\_expr>

| <a\_expr> '+' <m\_expr>

| <a\_expr> '-' <m\_expr>

<shift\_expr>::=

<a\_expr>

| <shift\_expr> RIGHT\_OP <a\_expr>

| <shift\_expr> LEFT\_OP <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>

| <comparison\_operators\_or\_exprs>

<comparison\_operators\_or\_exprs>::=

<comp\_operator>< or\_expr>

| <comparison\_operators\_or\_exprs> <comp\_operator> < or\_expr>

<comp\_operator>::=

"<" | ">" | "==" | ">=" | "<=" | "<>" | "!="

| IS | IS NOT | IN | NOT IN

<target\_list>::=

<target>

| <target\_list> ',' <target>

| <target\_list> ','

<target>::=

<identifier>

| '(' <target\_list> ')'

| '[' <target\_list> ']'

| <attributeref>

<attributeref>::=

<primary> '.' <identifier>

<atom>::=

<identifier>

| <literal>

| <enclosure>

<enclosure>::=

<parenth\_form>

| <dict\_display>

<parenth\_form>::=

'(' ')'

| '(' <expression\_list> ')'

<dict\_display>::=

'{' '}'

| '{' <key\_datum\_list> '}'

<key\_datum\_list>::=

<key\_datum>

| <key\_datum >','

| <key\_datum ><key\_datums>

| <key\_datum ><key\_datums> ','

<key\_datums>::=

',' <key\_datum>

| <key\_datums> ',' <key\_datum>

<key\_datum >::=

<expression >':' <expression>

<identifier>::=

IDENTIFIER

<stringliteral>::=

SHORTSTRING | LONGSTRING

<longinteger>::=

<integer> 'l' | <integer> 'L'

<integer>::=

DECINTEGER | OCTINTEGER | HEXINTEGER

<floatnumber>::=

POINTFLOAT | EXPONENTFLOAT

<imagnumber>::=

IMAGNUMBER

1. Περιγραφή του αντίστοιχου υποσυνόλου της Python σε BNF:

<start > ::= {printf(“Syntax check completed successfilly!!!\n”);}

|statements start

;

<statements> ::=

assign

|if

|for

|print

|import

|class

|func\_def

|call\_func

|class\_object

|use\_object

|lambda

|dict\_setdefault

|dict\_items

|break

|return

|NEWLINE

;

<expression> ::=

types

|dictionaries

|expression operators types

|OPENPAR expression CLOSPAR

;

<types> ::=  
    add\_letters      
    |add\_numbers    
    |var\_operations

|lists          
    ;  
      
<add\_letters> ::=  
    letters  
    |letters operators add\_letters  
    |letters operators var\_operations

| letters operators add\_obj  
    ;

<add\_numbers> ::=  
    numbers

|numbers PLUS add\_numbers

|numbers MINUS add\_numbers

|numbers STAR add\_numbers

|numbers DIV add\_numbers  
    |numbers operators add\_numbers  
    |numbers operators var\_operations  
    ;  
      
<var\_operations> ::=  
    ID

|ID PLUS types

|ID MINUS types

|ID STAR types

|ID DIV types  
    |ID operators types  
    ;

<add\_obj> ::=

use\_object

|use\_object operators types

;

<more\_types> ::=

types

|more\_types COMMA types

;

<numbers> ::=

DIGIT

|DEC

|FLOATINT

|OCT

|BINARY

|HEX

|typenumber

;

<letters> ::=

STRING

|STR OPENPAR types CLOSPAR

|QUOTE QUOTE

;

<lists> ::=

OPENAG more\_types CLOSAG

;

<dictionaries> ::=

OPENBR CLOSBR

OPENBR dictelements CLOSBR

;

<dictelements> ::=

moredictelem

|dictelements COMMA moredictelem

;

<moredictelem> ::=

types COLON types

;

<dict\_items> ::=

ID DOT ITEMS

;

<dict\_setdefault> ::=

ID DOT SETDEFAULT OPENPAR types COMMA types CLOSPAR

;

<operators> ::=

PLUS

|MINUS

|DIV

|STAR

|MOD

;

<break> ::=

BREAK;

<return> ::=

RETURN

| RETURN more\_types;

<lambda> ::=

LAMBDA COLON expression

|LAMBDA lambda\_parameters COLON expression

;

<lambda\_parameters> ::=

ID

|lambda\_parameters COMMA ID

<print> ::=

PRINT OPENPAR print\_star CLOSPAR

;

<print\_star> ::=

print\_options

|print\_star COMMA print\_options

;

<print\_options> ::=  
     types

|assign

|call\_func

|use\_object  
     |STRING MOD add\_letters               
     |STRING MOD add\_numbers            
     |STRING MOD var\_operations                         
     |STRING MOD OPENPAR more\_types CLOSPAR

     |LIST OPENPAR range CLOSPAR                          
     |range                                      
    ;

<import> ::=

IMPORT module

|FROM module IMPORT module

|FROM module IMPORT STAR

|IMPORT module AS ID

;

<module> ::=

id

|module COMMA id

;

id:

ID

|DOT

|DOT id

|id DOT ID

;

<assign> ::=

ID assign\_symb ass\_list

| ID assign\_symb expression

| ID EQUAL lambda

| ID EQUAL use\_object

| ID EQUAL QUOTE QUOTE

;

<assign\_symb> ::=

EQUAL

|PLUSEQ

|MINUSEQ

|STAREQ

|DIVEQ

;

<ass\_list> ::=

|TRUE

|FALSE

|NONE

;

<typenumber> ::=

INT OPENPAR numbers CLOSPAR

|FLOAT OPENPAR numbers CLOSPAR

;

<if> ::=

IF condition COLON more\_statm

|IF condition COLON more\_statm NEWLINE ELSE COLON more\_statm

|IF condition COLON more\_statm NEWLINE ELIF condition COLON more\_statm NEWLINE ELSE COLON more\_statm

;

<condition> ::=

ass\_list

|expression cond\_operators expression

|expression log\_operators expression

|OPENPAR condition CLOSPAR

;

<cond\_operators> ::=

GREATER

|SMALLER

|NOTEQ

|EQEQ

|GREQ

|SMEQ

;

<log\_operators> ::=

AND

|OR

;

<indent> ::=

|WHITESPACE

|indent WHITESPACE

;

<more\_statm> ::=

NEWLINE indent statements

|more\_statm NEWLINE indent statements

;

<for> ::=

FOR ID IN sequence COLON NEWLINE more\_statm

|FOR ID IN sequence COLON NEWLINE more\_statm NEWLINE ELSE COLON NEWLINE more\_statm

;

<sequence> ::=

ID

|lists

|dictionaries

|range

;

<range> ::=

RANGE OPENPAR numbers CLOSPAR

|RANGE OPENPAR numbers COMMA numbers CLOSPAR

|RANGE OPENPAR numbers COMMA numbers COMMA numbers CLOSPAR

|RANGE OPENPAR LEN OPENPAR ID CLOSPAR CLOSPAR

;

<class> ::=

CLASS classname COLON docstring

;

<classname> ::=

ID

|ID OPENPAR CLOSPAR

|ID OPENPAR class\_attr CLOSPAR

;

<class\_options> ::=

statements

;

<docstring> ::=

NEWLINE indent STRING NEWLINE more\_class\_options

|more\_class\_options

;

<more\_class\_options> ::=

NEWLINE indent class\_options

|more\_class\_options NEWLINE indent class\_options

;

<class\_attr> ::=

ID DOT ID

|more\_types class\_attr

|more\_types

;

<const\_options> ::=

NEWLINE indent ID DOT ID EQUAL types

|const\_options NEWLINE indent ID DOT ID EQUAL types

|NEWLINE indent statements

|const\_options NEWLINE indent statements

;

<class\_object> ::=

ID EQUAL ID OPENPAR CLOSPAR

|ID EQUAL ID OPENPAR object\_inside CLOSPAR

;

<object\_inside> ::=

ID

|letters

|numbers

|object\_inside COMMA object\_inside

;

<use\_object> ::=

ID DOT ID

|ID DOT call\_func

;

<func\_def> ::=

DEF ID OPENPAR CLOSPAR COLON func\_options

|DEF ID OPENPAR func\_attr CLOSPAR COLON func\_options

|DEF INIT OPENPAR class\_attr CLOSPAR COLON const\_options

;

<func\_attr> ::=

more\_types

|STAR ID

|DOUBLESTAR ID

|default\_attr

;

<default\_attr> ::=

ID EQUAL types

;

<func\_options> ::=

NEWLINE indent statements

|NEWLINE indent PASS

|func\_options NEWLINE indent statements

;

<call\_func> ::=

ID OPENPAR CLOSPAR

|ID OPENPAR call\_options CLOSPAR

;

<call\_options> ::=

ID

|letters

|numbers

|call\_options COMMA call\_options

|default\_attr

;

Χειρισμός Identation της Python:

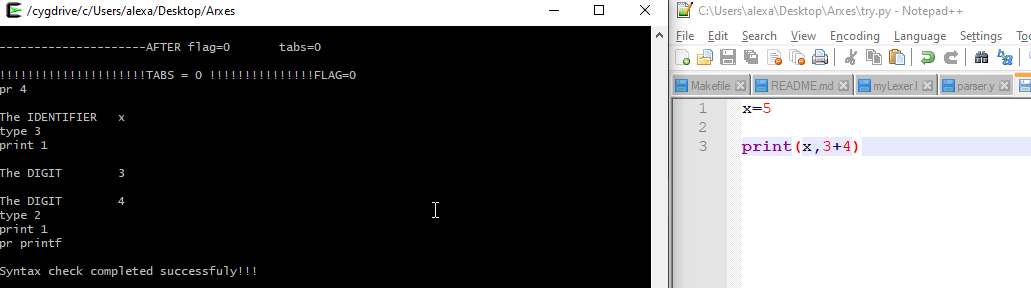
Για το χειρισμό του identation ,χρησιμοποιήσαμε τις βοηθητικές,global μεταβλητές flag,tabs,check,nested(τύπου integer και αρχικοποιημένες στο μηδέν).Στον Lexer μας για κάθε ειδικό token όπως def,for,if,class μαζί με το return token,αυξάνεται συν 1 το αντίστοιχο πεδίο(check,flag) εφόσον η python κάθε φορά που βλέπει ένα από αυτά τα keywords,περιμένει σωστό χειρισμό identation.Τέλος, η συνάρτηση identation() κάνει έλεγχο για εμφολεύσεις ,όπως if μέσα σε μια συνάρτηση και συγκρίνοντας κάθε φορά τη μεταβλητή flag με τα tabs ελέγχει αν το πρόγραμμα είναι συντακτικά ορθό.Παράλληλα η συνάρτηση identation() ελέγχει ώστε μετά από token αλλαγής γραμμγής (NEWLINE) να μην ξεκινάει γραμμή με κάποιο tab(σφάλμα).

Χειρισμός Σχολίων της Python:

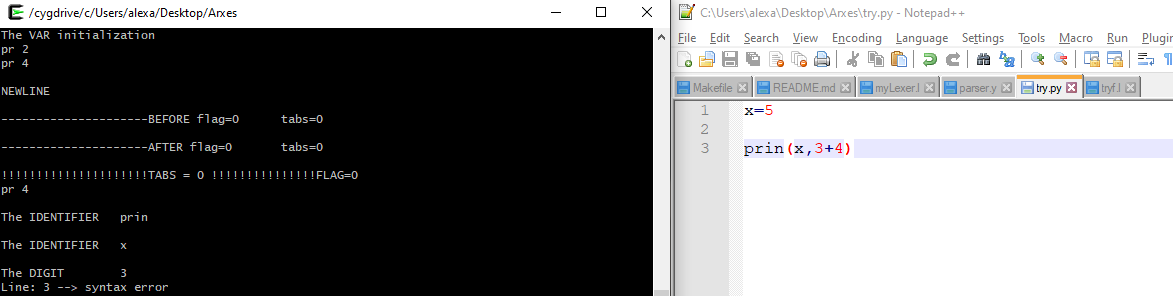
Ομοίως,στον Lexer μας γίνεται ο χειρισμός σχολίων,τα οποία ως γνωστόν ξεκινάνε με # .Αυτό γίνεται με την εντολή \#.+ ,δηλαδή όταν δεί # και ότι string ακολουθήσει λαμβάνεται ως σχόλιο και με την εντολή ^{WHITESPACE}\*\#.\* αναγνωρίζει σχόλιο γραμμής στο οποίο μπορεί να έχει προηγηθεί whitespace(tab ή space).

Έλεγχος Ορθότητας:

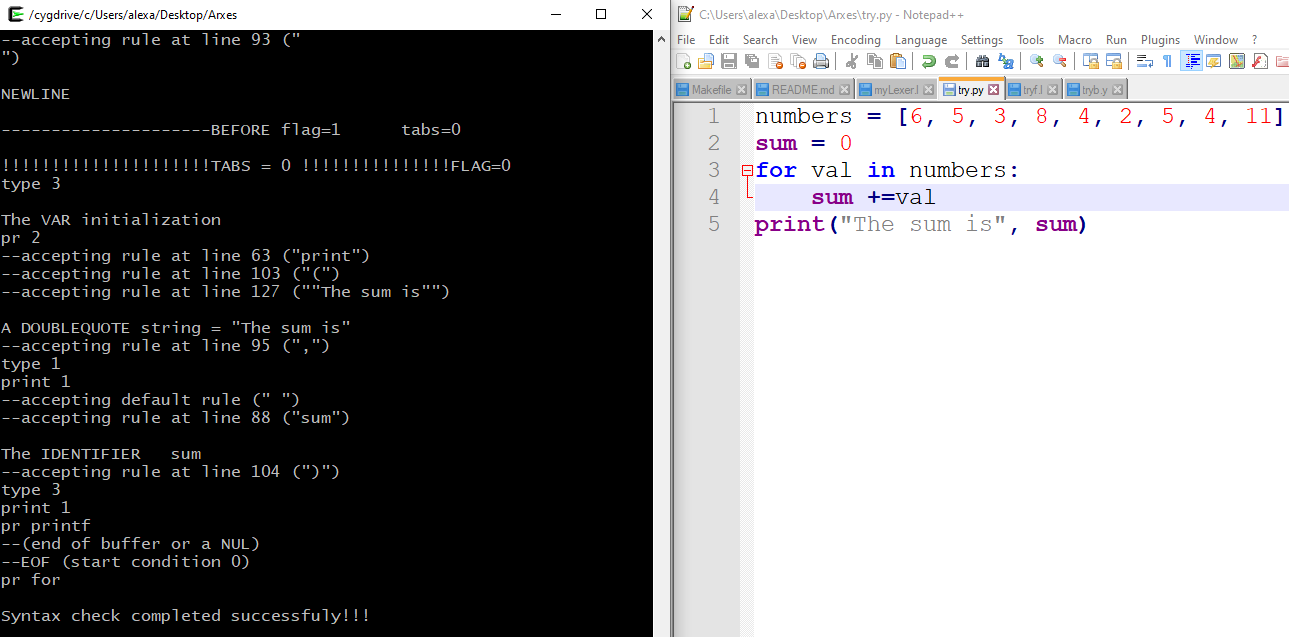
Επιτυχής συνάρτηση print:



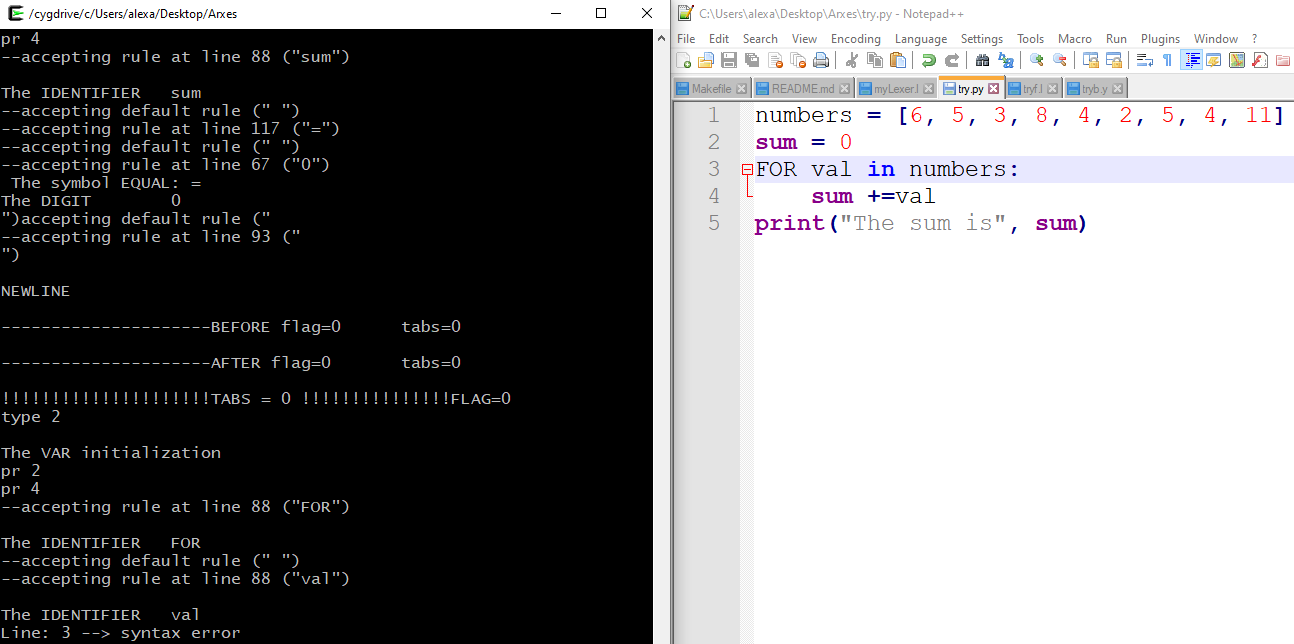
Ανεπιτυχής συνάρτηση print(γραμματικό λάθος-prin):



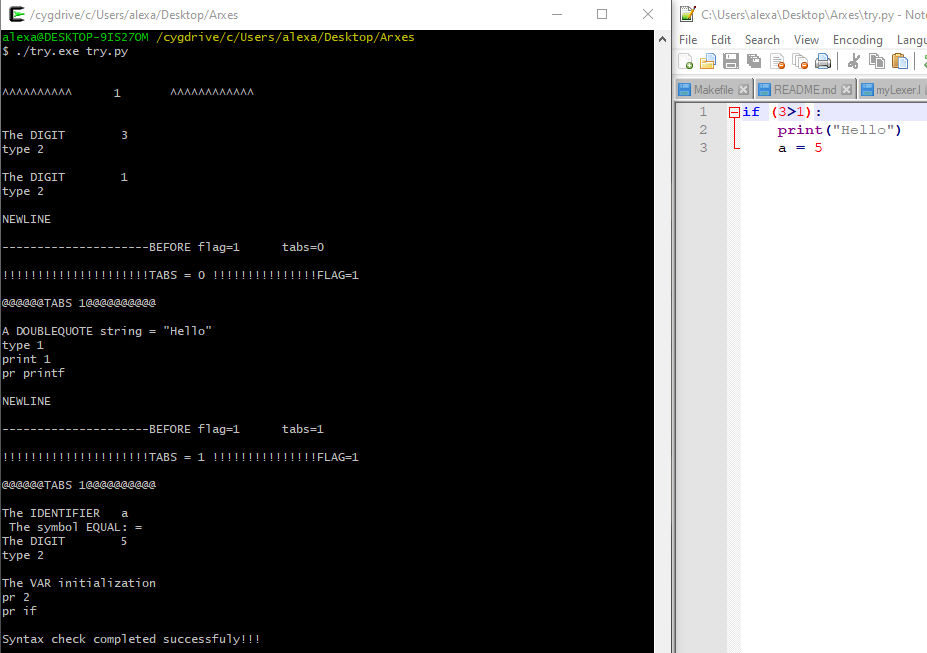
Επιτυχής εκτέλεση for βρόγχου:



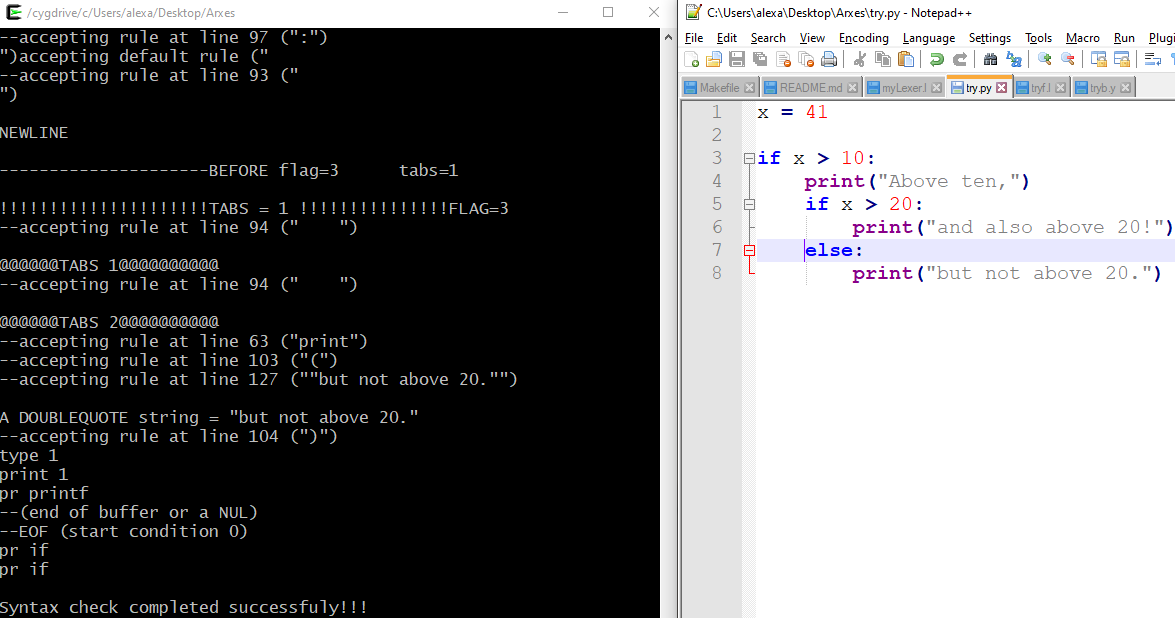
Ανεπιτυχής εκτέλεση for βρόγχου(γραμματικό λάθος FOR με κεφαλαία):



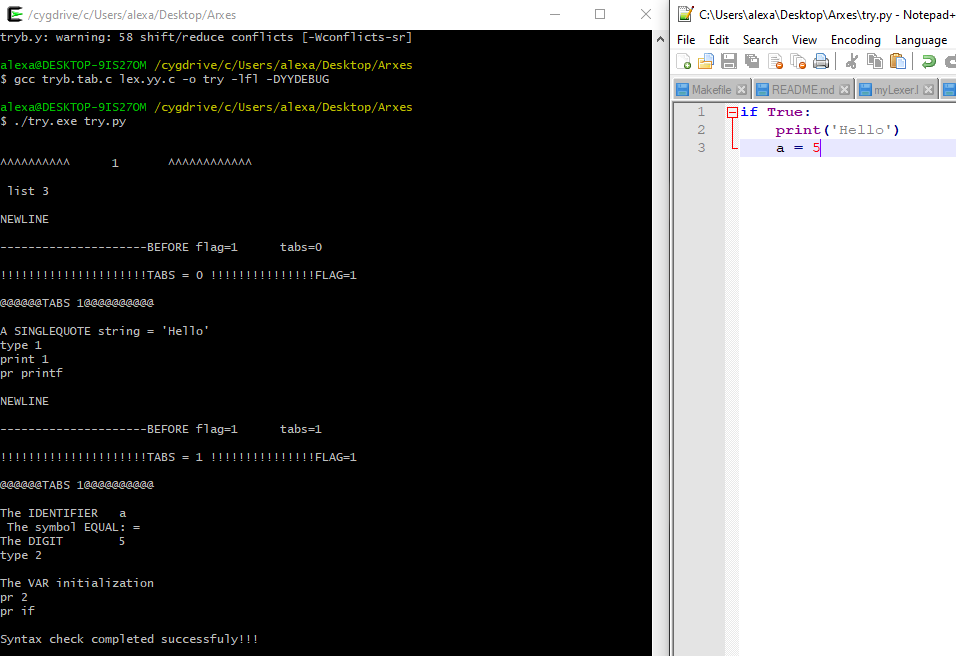
Επιτυχής εκτέλεση if :



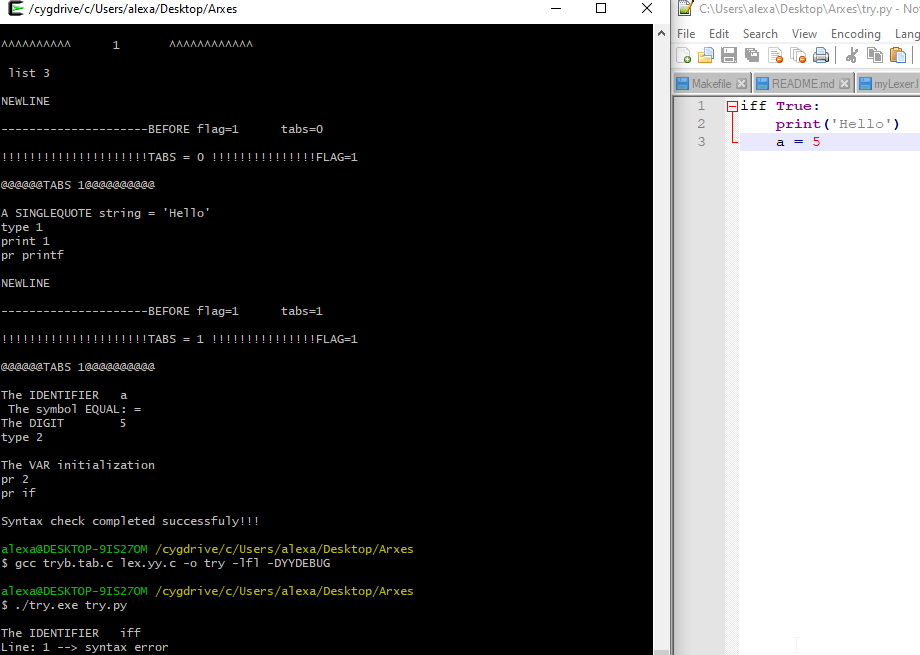
Επιτυχής εκτέλεση if-else:



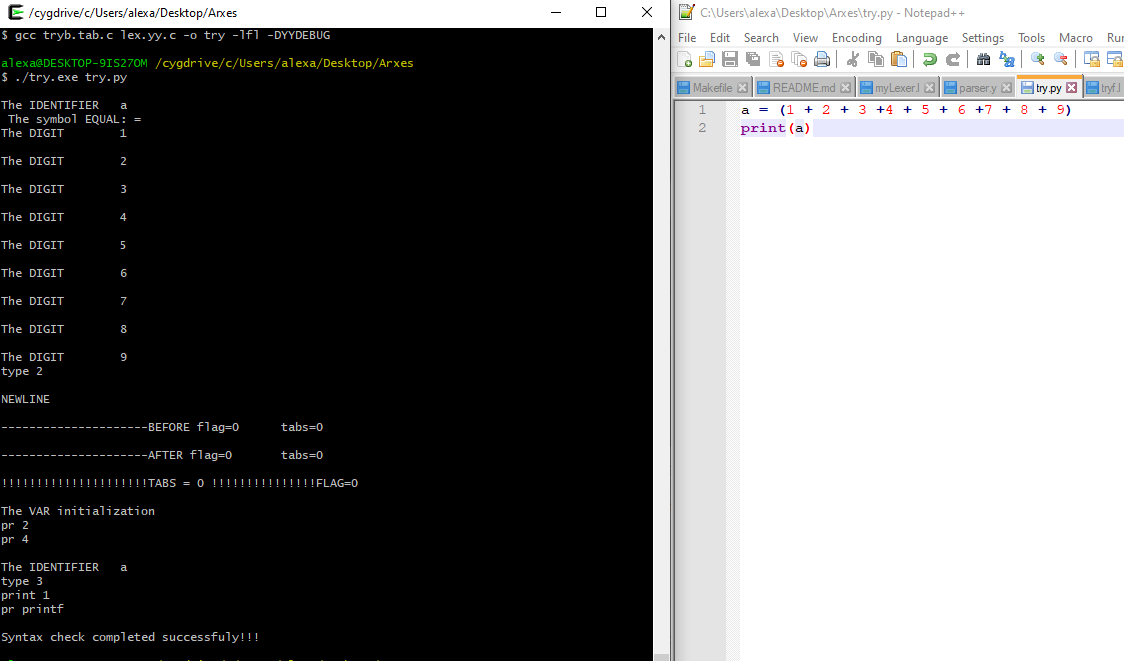
Επιτυχής εκτέλεση if :



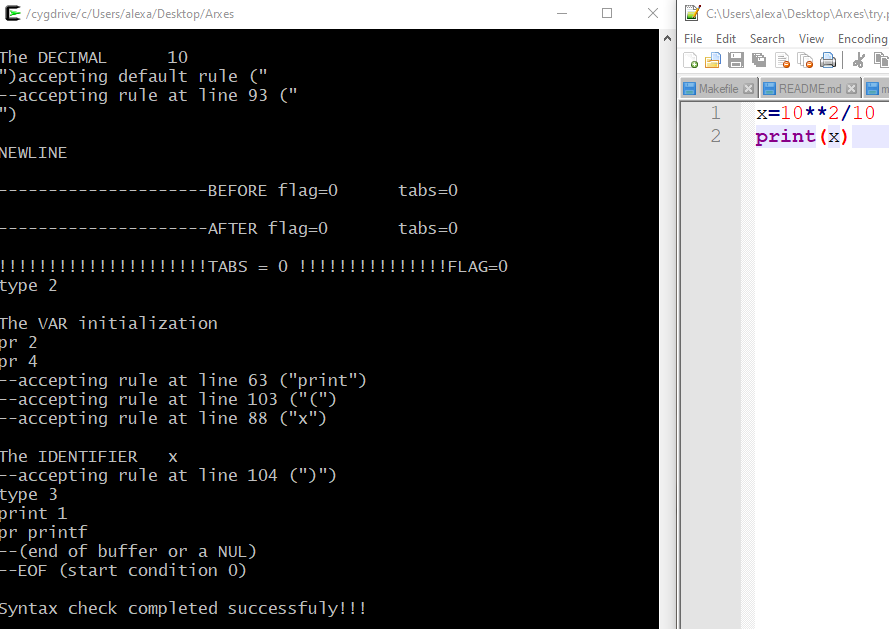
Ανεπιτυχής εκτέλεση if :



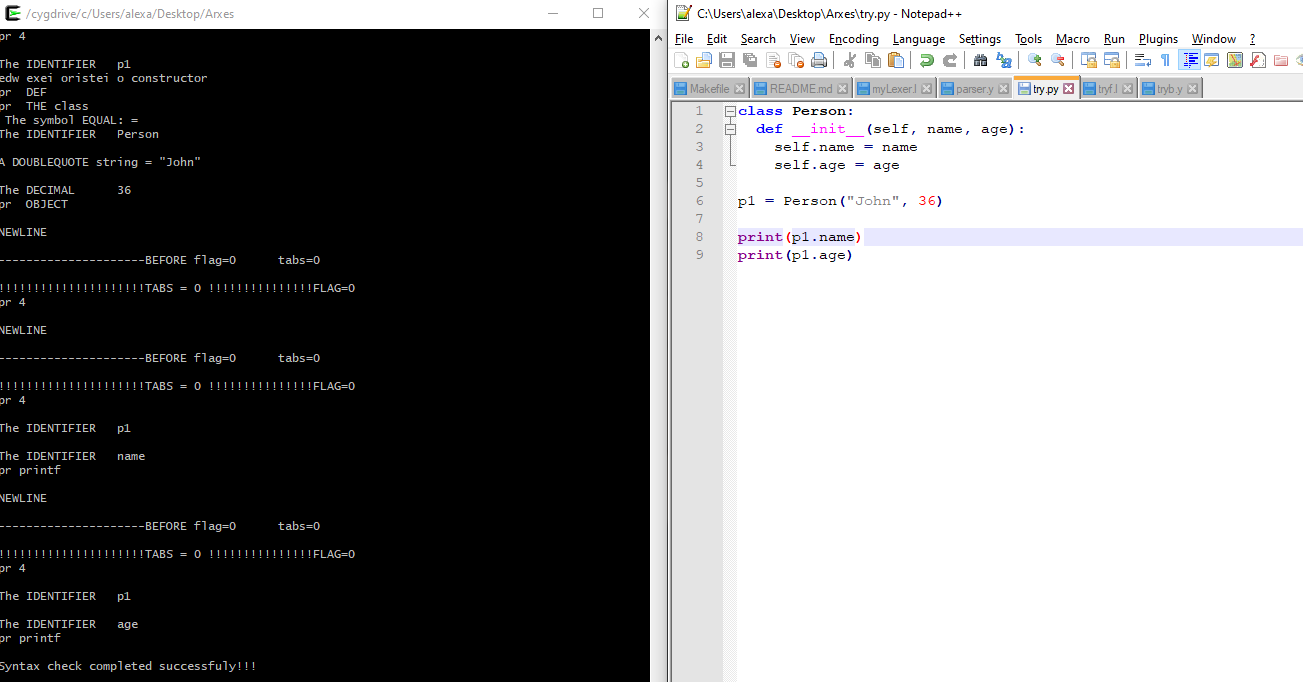
Επιτυχής εκτέλεση πράξης:



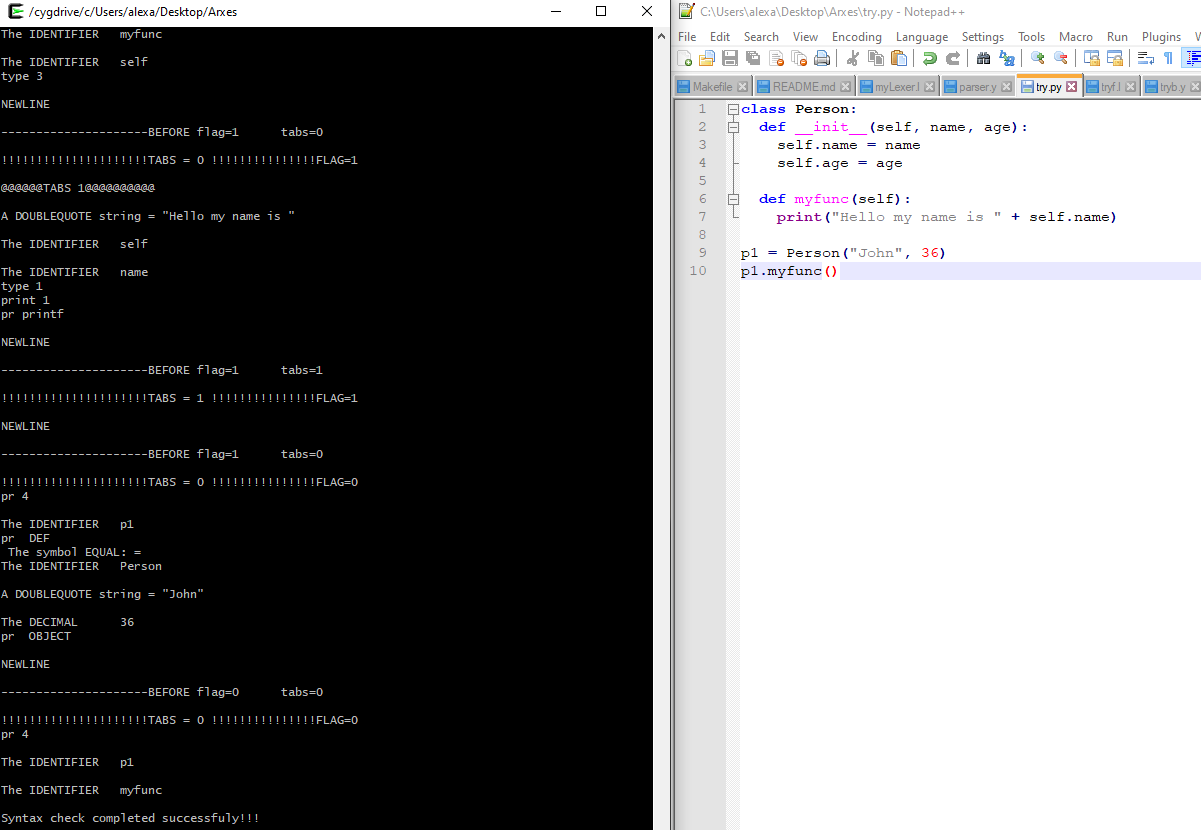
Επιτυχής εκτέλεση πράξης:



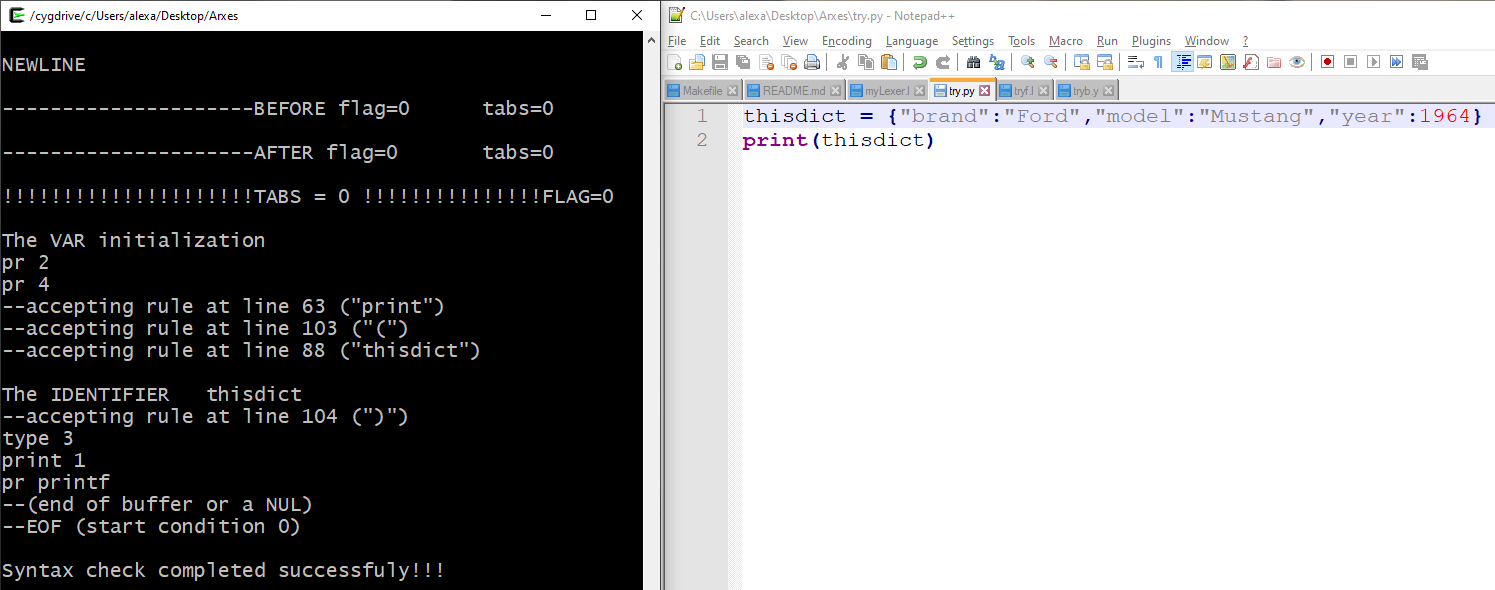
Επιτυχής εκτέλεση συνάρτησης:



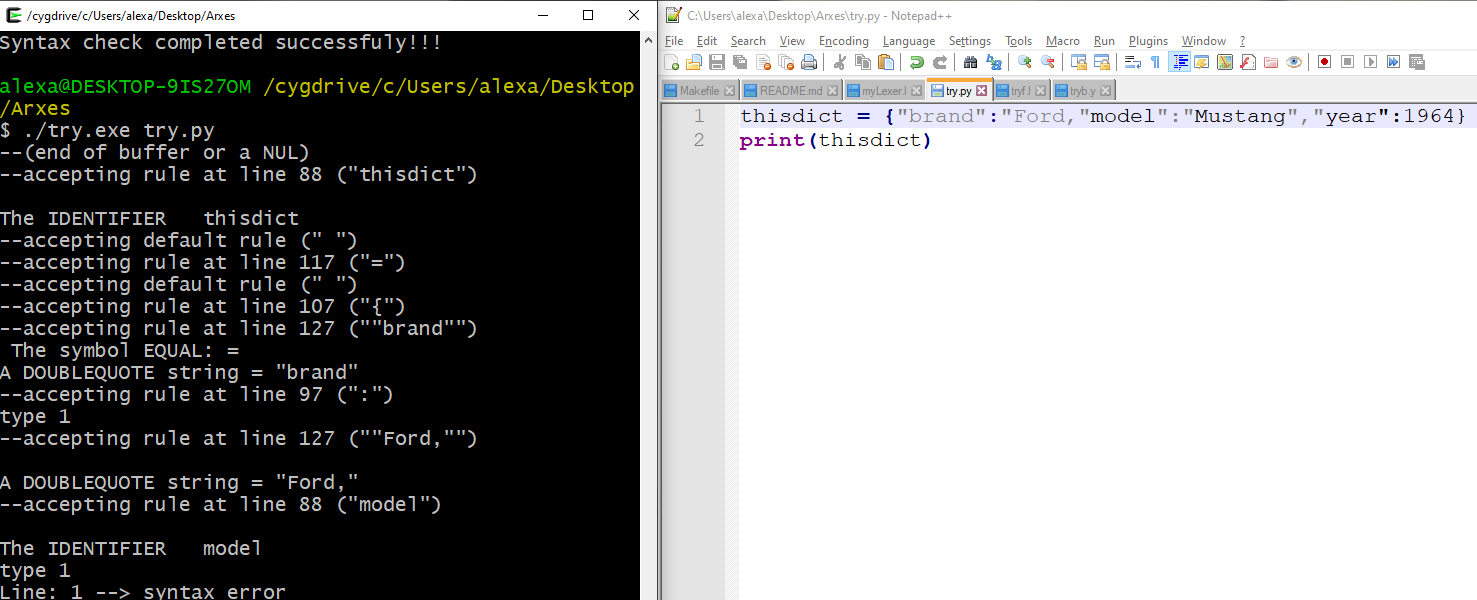
Ανεπιτυχής εκτέλεση συνάρτησης:



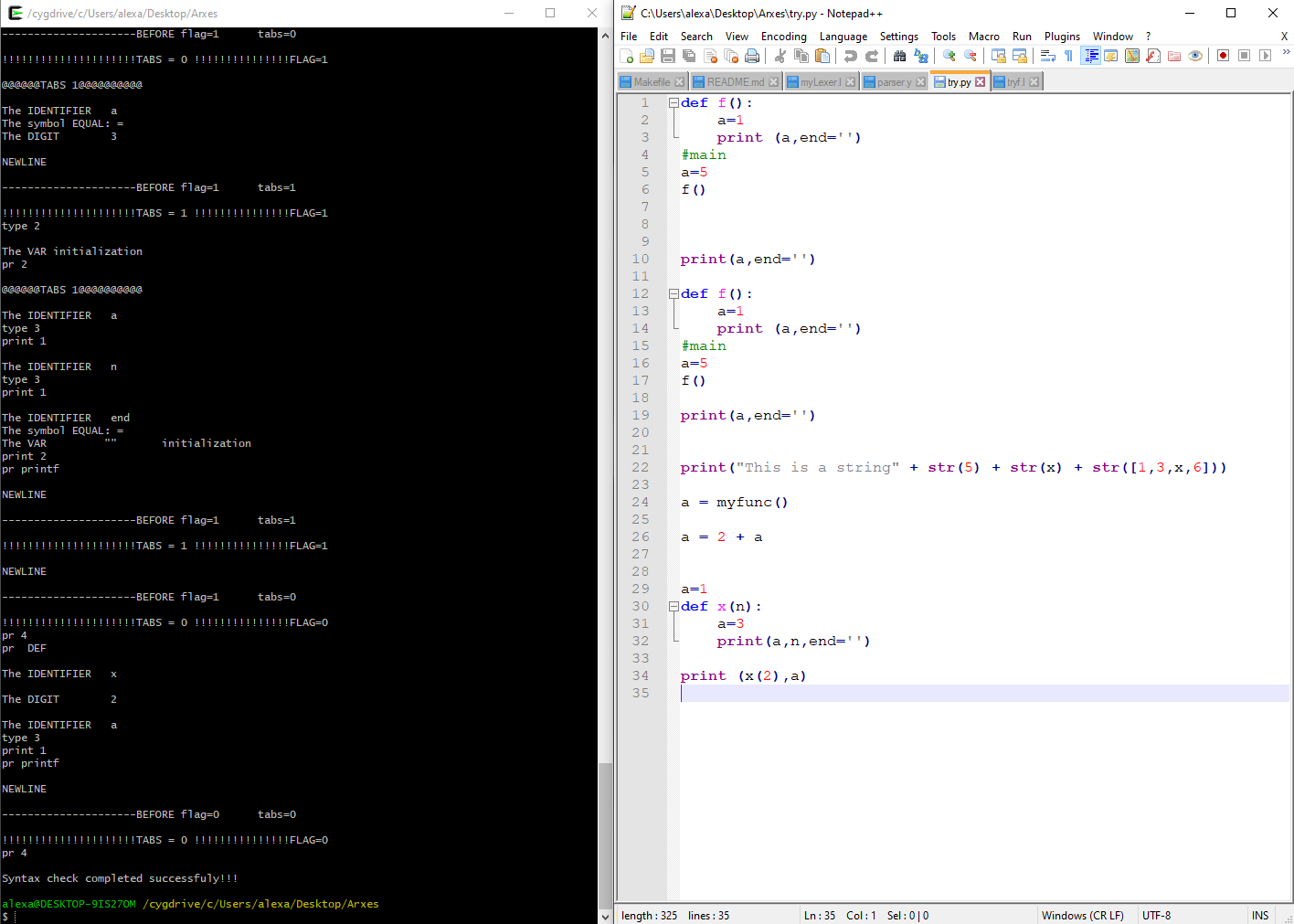
Επιτυχής εκτέλεση dictionary:



Ανεπιτυχής εκτέλεση dictionary(χωρίς κλείσουμε το “ στην δεύτερη εγγραφή):



Επιτυχής εκτέλεση σύνθετου προγράμματος:



ΑΝΑΦΟΡΕΣ ΚΑΙ ΒΙΒΛΙΟΓΡΑΦΙΑ

* <http://dinosaur.compilertools.net/flex/manpage.html> (Flex Official Manual)
* <https://www.gnu.org/software/bison/manual/bison.pdf> (Bison Official Manual)
* <https://www.oreilly.com/library/view/flex-bison/9780596805418/> (O’reilly manual on Flex/Bison)
* <https://docs.python.org/3/index.html> (Python Documentation)

©Αλεξανδρόπουλος Μάριος ΑΜ(1058116)

Αρβανίτης Ερμής-Ιωάννης ΑΜ(1059574)

Βαγιανού Εμμανουέλα AM(1059607)

Ακαδημαϊκό Έτος 2019-2020

Εργασία Εξαμήνου Αρχές Γλωσσών και Μεταφραστών- Τμήμα Μηχανικών Η/Υ & Πληροφορικής Πανεπιστήμιο Πατρών