

ΑΡΧΕΣ ΓΛΩΣΣΩΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΚΑΙ ΜΕΤΑΦΡΑΣΤΩΝ

Συμμετέχοντες

Άγγελος Μαργκάς 1059684
Ιάσων-Γεώργιος Παυλάκης 1059688
Κωνσταντίνος Τσάκωνας 1059666
Ιωάννης Χριστοδουλάκος 1062664

1.

Περιγραφή του υποσυνόλου της γλώσσας Python 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_target_list> <expression_list>

<assignment_stmt_target_list>::=
 <target_list> '='
 | <assignment_stmt_target_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> DOUBLESASH <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> 'I' | <integer> 'L'

<integer>::=
DECINTEGER | OCTINTEGER | HEXINTEGER

<floatnumber>::=
POINTFLOAT | EXPONENTFLOAT

<imagnumber>::=
IMAGNUMBER

2.

Περιγραφή της υλοποιημένης γλώσσας σε BNF

`<program> ::=`

`//empty`

`| <statement_list>`

`<statement_list> ::=`

`<statement_list> <statement>`

`| <statement>`

`<statement> ::=`

`<import_stmt>`

`| <assignment_stmt>`

`| <if_stmt>`

`| <for_stmt>`

`| <print_stmt>`

`| <funcdef>`

`| <classdef>`

`| <call>`

`| <return_stmt>`

`| <lambda_form`

`| <dict_setdefault>`

`| <dict_items>`

<return_stmt> ::=

RETURN

| RETURN <expression_list>

<call> ::=

<primary> LPAR RPAR

| <primary> LPAR <expression_list> RPAR

| <identifier> EQUAL <primary> LPAR RPAR

| <identifier> EQUAL <primary> LPAR <expression_list> RPAR

<primary> ::=

<identifier>

|<attr_identifier>

<lambda_form> ::=

LAMBDA COLON <expression>

| LAMBDA <parameter_list> COLON <expression>

//----- Print field -----

<print_stmt> ::=

PRINT

| PRINT <expression>

| PRINT <expression_list>

| PRINT RIGHT_OP <expression>

| PRINT RIGHT_OP <expression_list>

| PRINT LPAR <call> RPAR

//----- Assignment field -----

<assignment_stmt> ::=

 <assignment_stmt_targer_list> <expression_list>

 |<assignment_stmt_targer_list> <call>

<assignment_stmt_targer_list> ::=

 <target_list> EQUAL

 |<assignment_stmt_targer_list> <target_list> EQUAL

<target_list> ::=

 <target>

 |<target_list> COMMA <target>

 |<target_list> COMMA

<target> ::=

 <identifier>

 |<attr_identifier>

 |LPAR <target>_list RPAR

//----- Import filed -----

<import_stmt> ::=

 IMPORT <module>

 | IMPORT <module> AS <name>

 | IMPORT <modules> <modules>

 | IMPORT <modules> 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 LPAR <identifier> RPAR

| FROM <relative_module> IMPORT LPAR <identifier> AS <name> RPAR

| FROM <relative_module> IMPORT LPAR <identifier> <import_stmt_identifiers> RPAR

| FROM <relative_module> IMPORT LPAR <identifier> AS <name> <import_stmt_identifiers> RPAR

| FROM <relative_module> IMPORT LPAR <identifier> COMMA RPAR

| FROM <relative_module> IMPORT LPAR <identifier> AS <name> COMMA RPAR

| FROM <relative_module> IMPORT LPAR <identifier> <import_stmt_identifiers> COMMA RPAR

| FROM <relative_module> IMPORT LPAR <identifier> AS <name> <import_stmt_identifiers>
COMMA RPAR

| FROM <relative_module> IMPORT STAR

<module> ::=

<module> DOT <identifier>

| <identifier>

<relative_module> ::=

<module>

| <dots> <module>

| <dots>

<dots> ::=

DOT

| <dots> DOT

<modules> ::=

<modules> COMMA <module>

| <modules> COMMA <module> AS <name>

| COMMA <module>

| COMMA <module> AS <name>

<import_stmt_identifiers> ::=

COMMA <identifier>

| COMMA <identifier> AS <name>

| <import_stmt_identifiers> COMMA <identifier>

| <import_stmt_identifiers> COMMA <identifier> AS <name>

<name> ::=

<identifier>

//----- Compound_stmt field -----

//===== If

=====

<if_stmt> ::=

IF <expression> COLON <statement_list>

| IF <expression> COLON <statement_list> ELSE COLON <statement_list>

| IF <expression> COLON <statement_list> <elif_stmt>

| IF <expression> COLON <statement_list> <elif_stmt> ELSE COLON <statement_list>

<elif_stmt> ::=

ELIF <expression> COLON <statement_list>

| elif_stmt ELIF <expression> COLON <statement_list>

//===== For

=====

<for_stmt> ::=

FOR <for_target_list> IN <expression_list> COLON <statement_list>

|FOR <for_target_list> IN RANGE LPAR <expression_list> RPAR COLON <statement_list>

| FOR <for_target_list> IN <expression_list> COLON <statement_list> ELSE COLON
<statement_list>

<for_target_list> ::=

<for_target>

| <for_target_list> COMMA <target>

| <for_target_list> COMMA

<for_target> ::=

<identifier>

|LPAR <for_target_list> RPAR

//===== Function =====

<funcdef> ::=

DEF <funcname> LPAR RPAR COLON <statement_list>

|<decorators> DEF <funcname> LPAR RPAR COLON <statement_list>

| DEF <funcname> LPAR <parameter>_list RPAR COLON <statement_list>

| <decorators> DEF <funcname> LPAR <parameter_list> RPAR COLON <statement_list>

<decorators> ::=

<decorator>

| <decorators> <decorator>

<decorator> ::=

PAPAKI <dotted_name> NEWLINE

| PAPAKI <dotted_name> LPAR RPAR NEWLINE

<dotted_name> ::=

<identifier>

| <identifier> <dot_identifiers>

<dot_identifiers> ::=

DOT <identifier>

| <dot_identifier> DOT <identifier>

<parameter_list> ::=

STAR <identifier>

| STAR <identifier> COMMA DOUBLESTAR <identifier>

| DOUBLESTAR <identifier>

| <defparameter>

| <defparameter> COMMA

| <defparameters> STAR <identifier>

| <defparameters> STAR <identifier> COMMA DOUBLESTAR <identifier>

| <defparameters> DOUBLESTAR <identifier>

| <defparameters> <defparameter>

| <defparameters> <defparameter> COMMA

<defparameter> ::=

<parameter>

| <parameter> EQUAL expression

<defparameters> ::=

<defparameter> COMMA

| <defparameters> <defparameter> COMMA

<sublist> ::=

<parameter>

| <parameter> COMMA

| <parameter> <parameters>

| <parameter> <parameters> COMMA

<parameters> ::=

COMMA <parameter>

| <parameters> COMMA <parameter>

<parameter> ::=

<identifier>

| LPAR <sublist> RPAR

<funcname> ::=

<identifier>

< /= ===== Class

=====

<classdef> ::=

CLASS <classname> COLON <statement_list>

| CLASS classname> <inheritance COLON> <statement_list>

<inheritance> ::=

LPAR RPAR

| LPAR <expression_list> RPAR

<classname> ::=

<identifier>

//----- etc -----

<dict_items> ::=

<identifier> DOT ITEMS LPAR RPAR

<dict_setdefault> ::=

<identifier> DOT SETDEFAULT LPAR <expression> COMMA <expression> RPAR

<dict_display> ::=

LBRA RBRA

| LBRA <key_datum_list> RBRA

<key_datum_list> ::=

<key_datum>

| <key_datum > COMMA

| <key_datum > <key_datums>

| <key_datum > <key_datums> COMMA

<key_datums> ::=

COMMA <key_datum>

| <key_datums> COMMA <key_datum >

<key_datum> ::=

<expression> COLON <expression>

<expression_list> ::=

<expression_list> COMMA expression

| LPAR <expression_list> COMMA <expression> RPAR

| <expression>

<expression > ::=

<atom>

| LPAR <expression> RPAR

| <expression> PLUS <expression>

| <expression> MINUS <expression>

| <expression> SLASH <expression>

| <expression> STAR <expression>

| <expression> <assignment_op> <expression>

| <expression> <arithmetic_op> <expression>

| <expression> <comparison_op> <expression>

| <expression> logical_op <expression>

| <expression> <bitwise_op> <expression>

<atom> ::=

<literal>

| <identifier>

| <integer>

| <attr_identifier>

| <dict_display>

| <dict_setdefault>

<literal> ::=

<string>

| <longinteger>

| <imagnumber>

<attr_identifier> ::=

<identifier>

| attr_<identifier> DOT <identifier>

| <identifier> DOT <identifier>

<stringliteral> ::=
 <shortstring>
 |<longstring>

<shortstring>::=
 <any source character except "" or newline>

longstringitem ::=
 <any source character except \' '>

imagnumber ::= (floatnumber | intpart) ("j" | "J")

longinteger ::=
 integer ("l" | "L")

integer ::=
 decimalinteger | octinteger | hexinteger

decimalinteger ::=
 nonzerodigit digit* | "0"

octinteger ::=
 "0" octdigit+

hexinteger ::=
 "0" ("x" | "X") hexdigit+

nonzerodigit ::=
 "1"... "9"

octdigit ::=
 "0"... "7"

hexdigit ::=
 digit | "a"... "f" | "A"... "F"

floatnumber ::=
 pointfloat | exponentfloat

pointfloat ::=
 [intpart] fraction | intpart "."

exponentfloat ::=
 (intpart | pointfloat)
 exponent

intpart ::=
 digit+

fraction ::=
 "." digit+

exponent ::=
 ("e" | "E") ["+" | "-"] digit+

identifier ::=
 (letter|"_") (letter | digit | "_")*

(where the matched string is not a keyword)

letter ::=
 lowercase | uppercase

lowercase ::=
 "a"|"b"|..."z"

uppercase ::=
 "A"|"B"|..."Z"

digit ::=
 "0"|"1"|..."9"

<assignment_op> ::=

 ADD_ASSIGN
 | SUB_ASSIGN
 | MUL_ASSIGN
 | POW_ASSIGN
 | DIV_ASSIGN
 | MOD_ASSIGN
 | AND_ASSIGN
 | XOR_ASSIGN
 | OR_ASSIGN
 | RIGHT_ASSIGN
 | LEFT_ASSIGN

<arithmetic_op> ::=
 PERCENT
 | DOUBLESTAR
 | DOUBLESASH

<comparison_op> ::=

EQ_OP
| NE_OP
| GREATER_THAN_OP
| LESS_THAN_OP
| LE_OP
| GE_OP

<logical_op> ::=

AND
| NOT
| OR
| IS
| IN
| IS NOT
| NOT IN

<bitwise_op> ::=

AND_EXP
| OR_SIGN
| XOR
| NOT_SIGN
| LEFT_OP
| RIGHT_OP

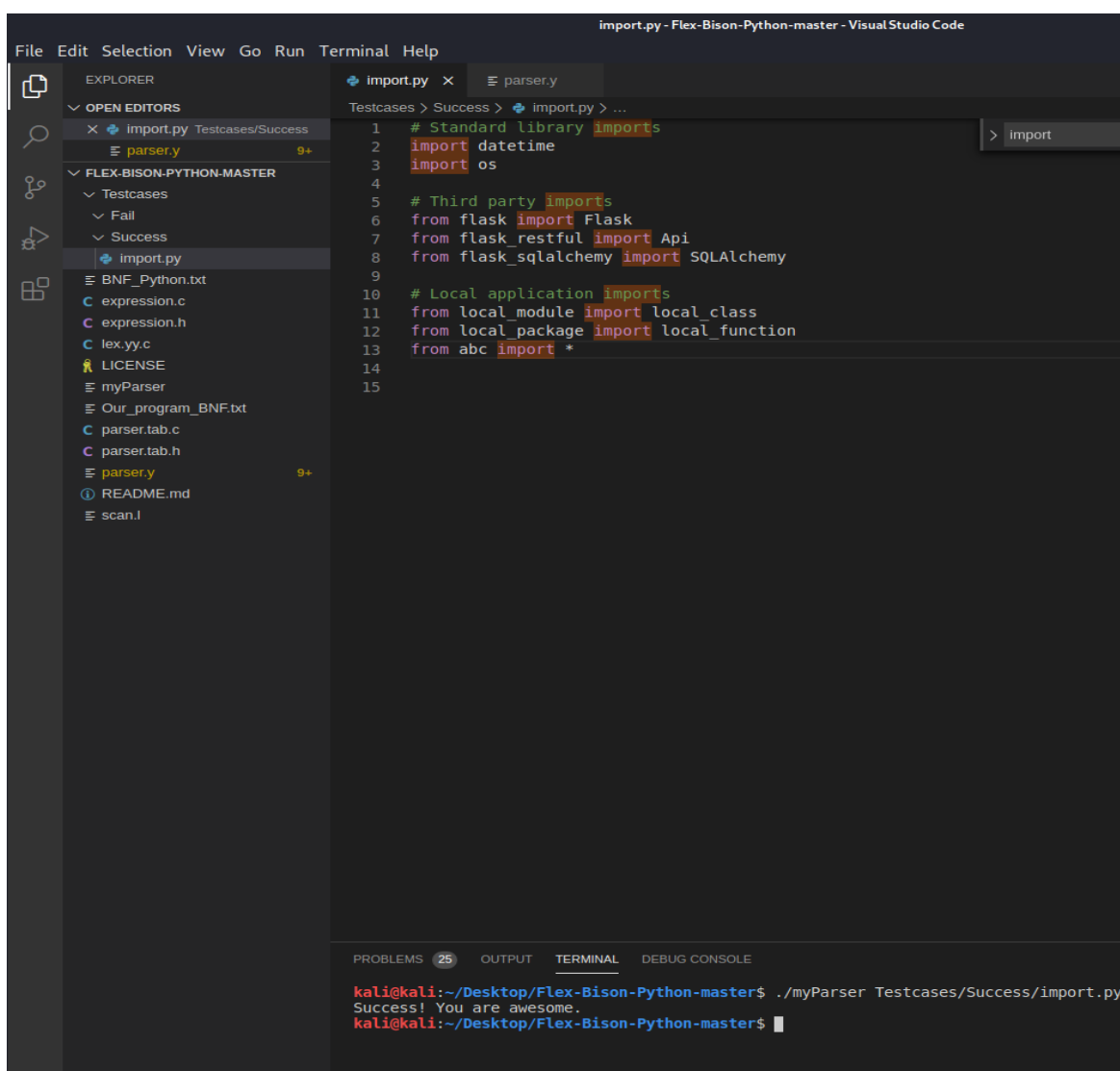
3.

Αρχεία περιγραφής της γλώσσας, τα οποία δίνονται ως είσοδος στα Flex και Bison

A)

Εντολή import

Επιτυχημένη προσπάθεια:



```
import.py - Flex-Bison-Python-master - VisualStudio Code

File Edit Selection View Go Run Terminal Help

EXPLORER
  OPEN EDITORS
    import.py Testcases/Success
    parser.y 9+
  FLEX-BISON-PYTHON-MASTER
    Testcases
      Fail
      Success
        import.py
    BNF_Python.txt
    expression.c
    expression.h
    lex.yy.c
    LICENSE
    myParser
    Our_program_BNF.txt
    parser.tab.c
    parser.tab.h
    parser.y 9+
    README.md
    scan.l

Testcases > Success > import.py > ...
1 # Standard library imports
2 import datetime
3 import os
4
5 # Third party imports
6 from flask import Flask
7 from flask_restful import Api
8 from flask_sqlalchemy import SQLAlchemy
9
10 # Local application imports
11 from local_module import local_class
12 from local_package import local_function
13 from abc import *
14
15

PROBLEMS 25 OUTPUT TERMINAL DEBUG CONSOLE
kali@kali:~/Desktop/Flex-Bison-Python-master$ ./myParser Testcases/Success/import.py
Success! You are awesome.
kali@kali:~/Desktop/Flex-Bison-Python-master$
```

Αναγνώριση όλων των τύπων εντολών import και ενημέρωση του χρήστη για την επιτυχημένη προσπάθεια μεταγλώττισης.

Αποτυχημένες προσπάθειες:

The screenshot shows the Visual Studio Code interface with the file explorer on the left and the editor on the right. The file explorer shows the project structure, including the 'FLEX-BISON-PYTHON-MASTER' directory. The editor displays the file 'import.py' with the following code:

```
1 # Standard library imports
2 import datetime
3 import os
4
5 # Third party imports
6 from flask import Flask
7 from flask_restful import Api
8 from flask_sqlalchemy import SQLAlchemy
9
10 # Local application imports
11 from local_module import local_class
12 from local_package import local_function
13 from abc import *
14
```

The terminal at the bottom shows the command `./myParser Testcases/Fail/import.py` and the output `Line: 3 --> Parser error`.

The screenshot shows the Visual Studio Code interface with the file explorer on the left and the editor on the right. The file explorer shows the project structure, including the 'FLEX-BISON-PYTHON-MASTER' directory. The editor displays the file 'import.py' with the following code:

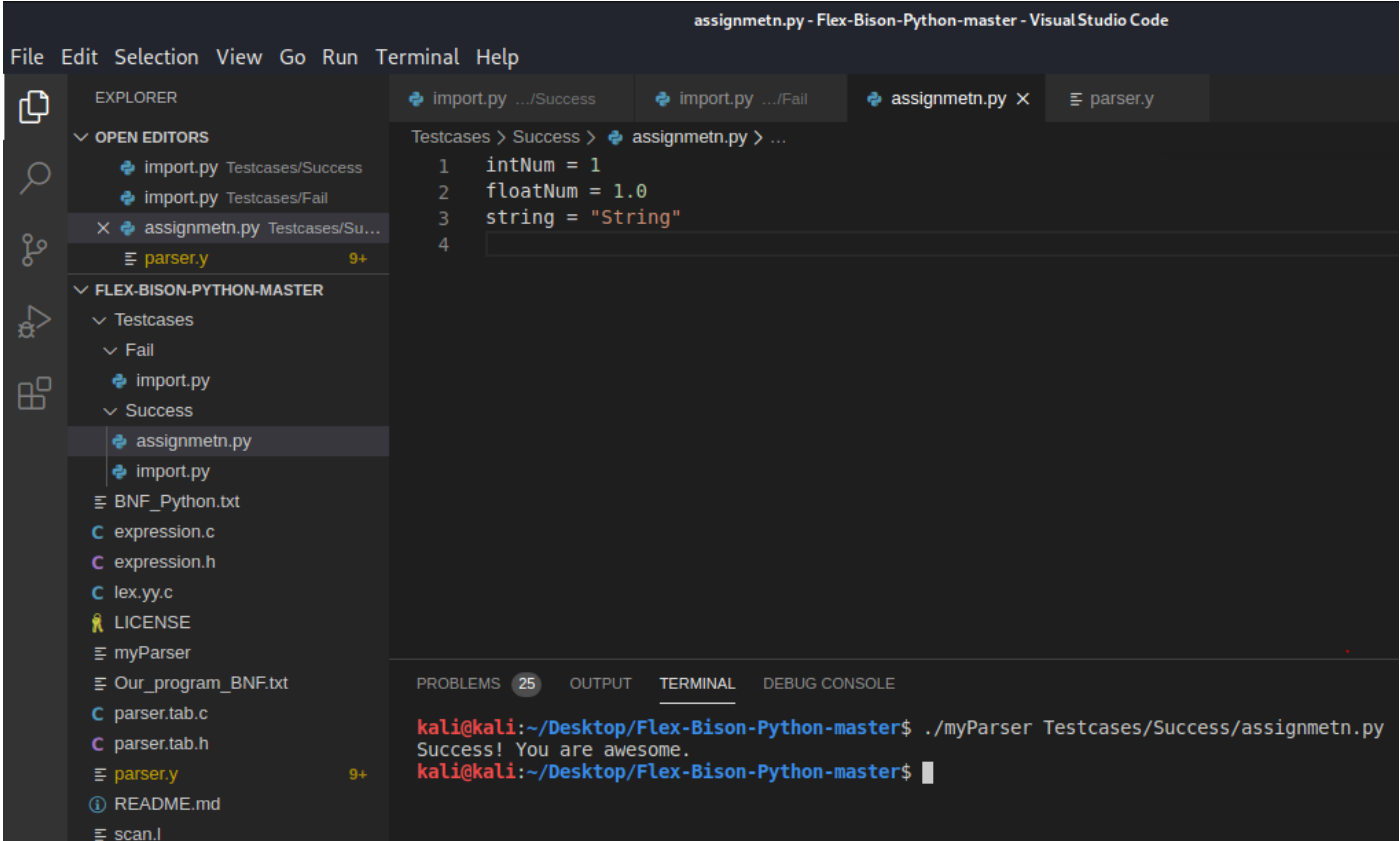
```
1 # Standard library imports
2 import datetime
3 import os
4
5 # Third party imports
6 from flask import Flask
7 from from flask_restful import Api
8 from flask_sqlalchemy import SQLAlchemy
9
10 # Local application imports
11 from local_module import local_class
12 from local_package import local_function
13 from abc import *
14
```

The terminal at the bottom shows the command `./myParser Testcases/Fail/import.py` and the output `Line: 7 --> Parser error`.

B)

Αρχικοποίηση μεταβλητών

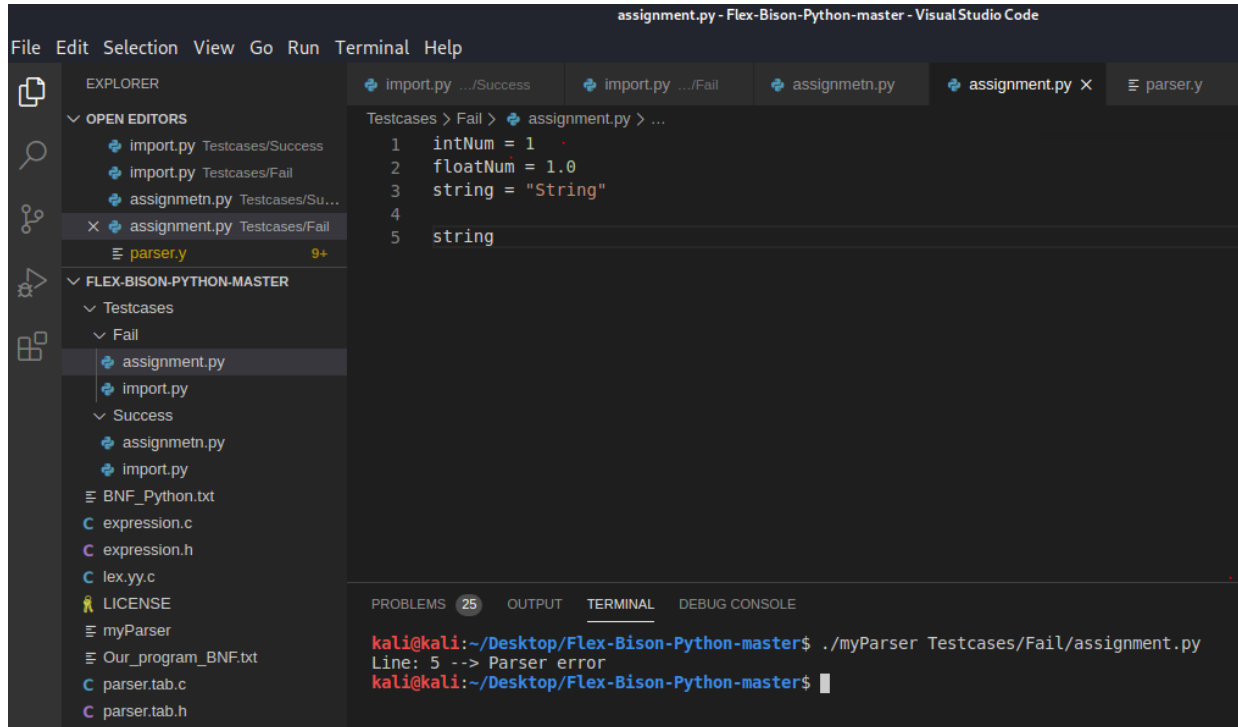
Επιτυχημένη προσπάθεια:



The screenshot shows the Visual Studio Code interface with the file explorer on the left, the editor in the center, and the terminal at the bottom. The file explorer shows the project structure with folders for Testcases, Fail, Success, and various files. The editor displays the content of assignmetn.py, which contains three lines of code: intNum = 1, floatNum = 1.0, and string = "String". The terminal shows the command ./myParser Testcases/Success/assignmetn.py being executed, resulting in the output: Success! You are awesome.

```
assignmetn.py - Flex-Bison-Python-master - VisualStudio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
OPEN EDITORS
  import.py Testcases/Success
  import.py Testcases/Fail
  X assignmetn.py Testcases/Su...
  parser.y 9+
FLEX-BISON-PYTHON-MASTER
  Testcases
  Fail
  Success
    import.py
    assignmetn.py
    import.py
  BNF_Python.txt
  expression.c
  expression.h
  lex.yy.c
  LICENSE
  myParser
  Our_program_BNF.txt
  parser.tab.c
  parser.tab.h
  parser.y 9+
  README.md
  scan.l
Testcases > Success > assignmetn.py > ...
1 intNum = 1
2 floatNum = 1.0
3 string = "String"
4
PROBLEMS 25 OUTPUT TERMINAL DEBUG CONSOLE
kali@kali:~/Desktop/Flex-Bison-Python-master$ ./myParser Testcases/Success/assignmetn.py
Success! You are awesome.
kali@kali:~/Desktop/Flex-Bison-Python-master$
```

Αποτυχημένες προσπάθειες:



assignment.py - Flex-Bison-Python-master - VisualStudio Code

File Edit Selection View Go Run Terminal Help

EXPLORER

- import.py .../Success
- import.py .../Fail
- assignment.py
- assignment.py X
- parser.y

OPEN EDITORS

- import.py Testcases/Success
- import.py Testcases/Fail
- assignment.py Testcases/Su...
- assignment.py Testcases/Fail
- parser.y 9+

FLEX-BISON-PYTHON-MASTER

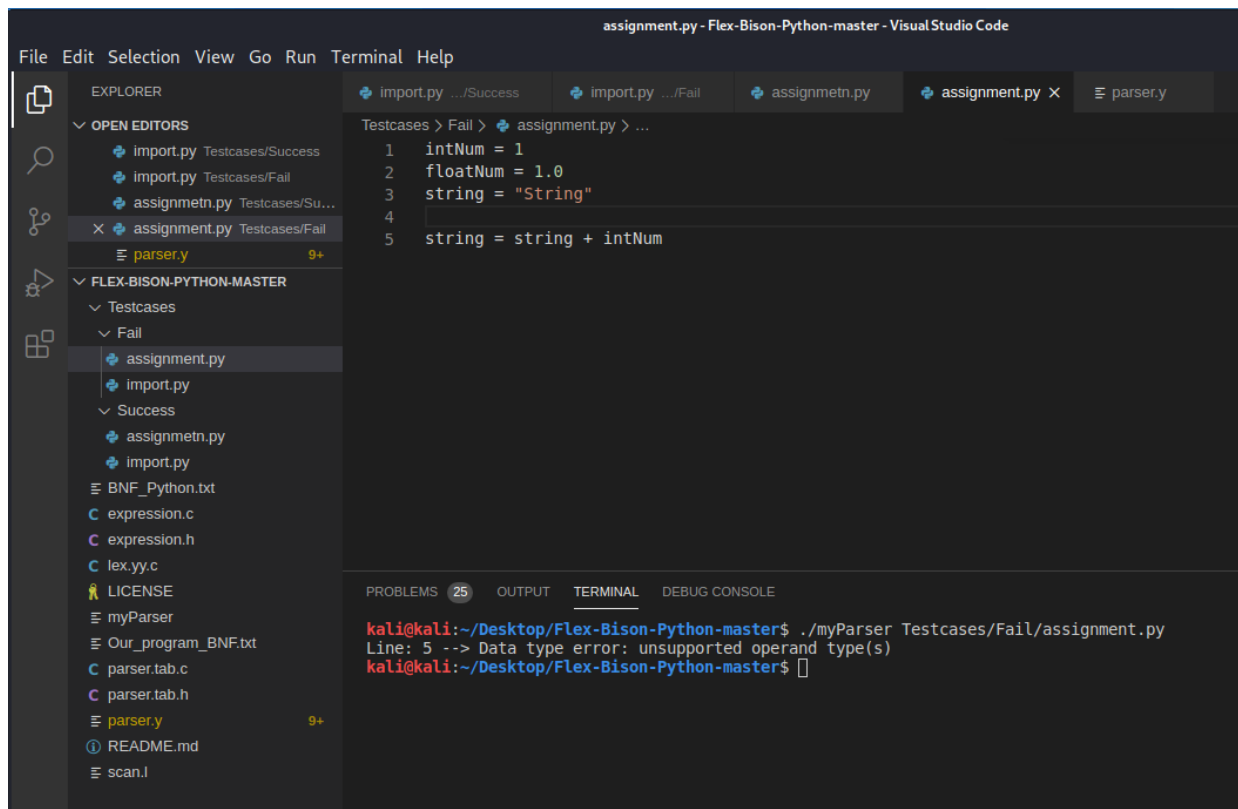
- Testcases
 - Fail
 - assignment.py
 - import.py
 - Success
 - assignment.py
 - import.py
- BNF_Python.txt
- expression.c
- expression.h
- lex.yy.c
- LICENSE
- myParser
- Our_program_BNF.txt
- parser.tab.c
- parser.tab.h

Testcases > Fail > assignment.py > ...

```
1 intNum = 1
2 floatNum = 1.0
3 string = "String"
4
5 string
```

PROBLEMS 25 OUTPUT TERMINAL DEBUG CONSOLE

```
kali@kali:~/Desktop/Flex-Bison-Python-master$ ./myParser Testcases/Fail/assignment.py
Line: 5 --> Parser error
kali@kali:~/Desktop/Flex-Bison-Python-master$
```



assignment.py - Flex-Bison-Python-master - VisualStudio Code

File Edit Selection View Go Run Terminal Help

EXPLORER

- import.py .../Success
- import.py .../Fail
- assignment.py
- assignment.py X
- parser.y

OPEN EDITORS

- import.py Testcases/Success
- import.py Testcases/Fail
- assignment.py Testcases/Su...
- assignment.py Testcases/Fail
- parser.y 9+

FLEX-BISON-PYTHON-MASTER

- Testcases
 - Fail
 - assignment.py
 - import.py
 - Success
 - assignment.py
 - import.py
- BNF_Python.txt
- expression.c
- expression.h
- lex.yy.c
- LICENSE
- myParser
- Our_program_BNF.txt
- parser.tab.c
- parser.tab.h
- parser.y 9+
- README.md
- scan.l

Testcases > Fail > assignment.py > ...

```
1 intNum = 1
2 floatNum = 1.0
3 string = "String"
4
5 string = string + intNum
```

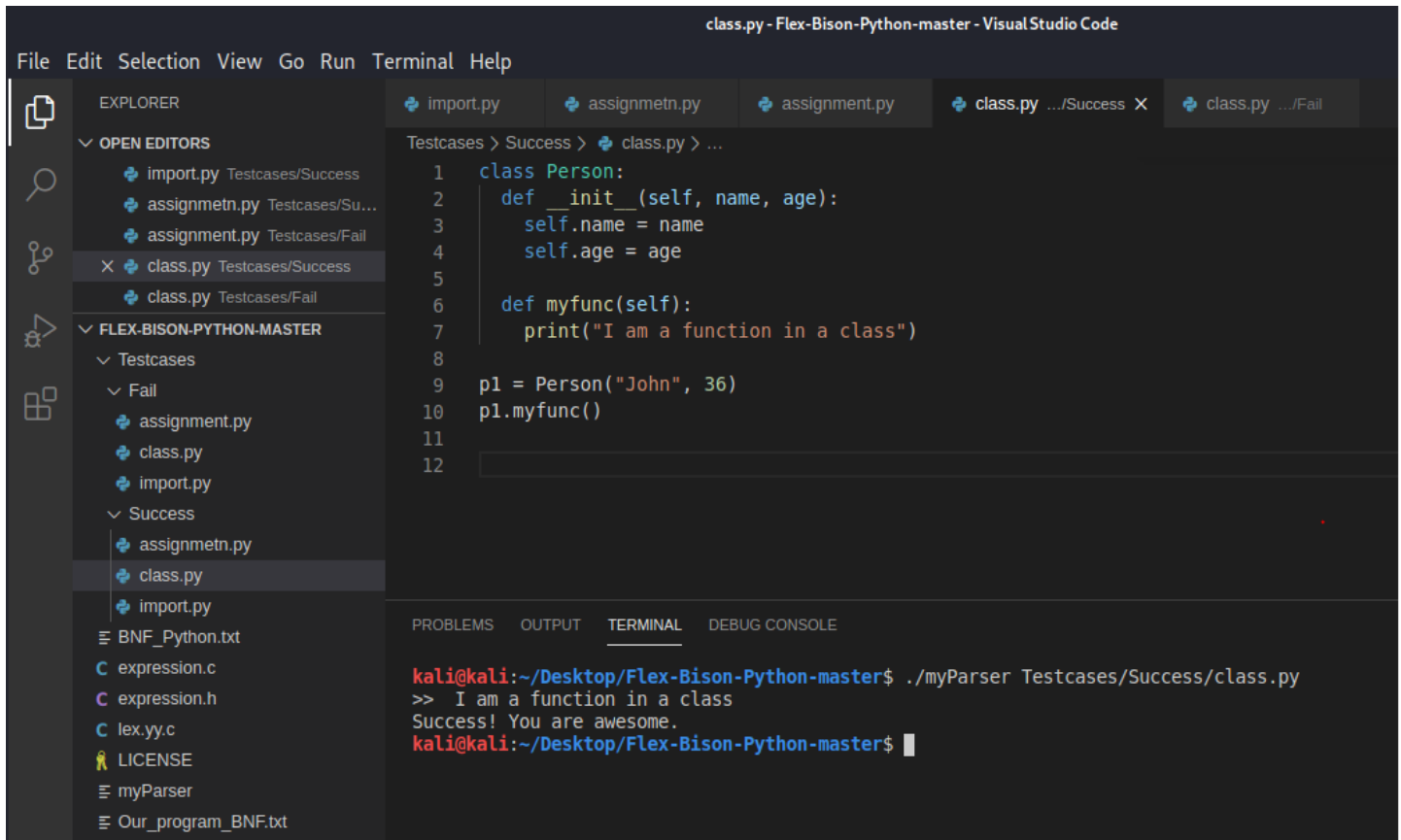
PROBLEMS 25 OUTPUT TERMINAL DEBUG CONSOLE

```
kali@kali:~/Desktop/Flex-Bison-Python-master$ ./myParser Testcases/Fail/assignment.py
Line: 5 --> Data type error: unsupported operand type(s)
kali@kali:~/Desktop/Flex-Bison-Python-master$
```

Γ)

Αρχικοποίηση κλάσης και αντικειμένου

Επιτυχημένη προσπάθεια:



The screenshot shows the Visual Studio Code interface with the file explorer on the left, the editor in the center, and the terminal at the bottom. The file explorer shows the project structure with folders for 'Testcases' (containing 'Fail' and 'Success') and 'FLEX-BISON-PYTHON-MASTER'. The editor displays the content of 'class.py' in the 'Success' folder, which defines a 'Person' class with an '__init__' method and a 'myfunc' method. The terminal shows the command to run the script and its output.

```
class.py - Flex-Bison-Python-master - Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER
  OPEN EDITORS
    import.py Testcases/Success
    assignmetn.py Testcases/Su...
    assignment.py Testcases/Fail
    class.py Testcases/Success
    class.py Testcases/Fail
  FLEX-BISON-PYTHON-MASTER
    Testcases
      Fail
        assignment.py
        class.py
        import.py
      Success
        assignmetn.py
        class.py
        import.py
    BNF_Python.txt
    expression.c
    expression.h
    lex.yy.c
    LICENSE
    myParser
    Our_program_BNF.txt

Testcases > Success > class.py > ...
1 class Person:
2     def __init__(self, name, age):
3         self.name = name
4         self.age = age
5
6     def myfunc(self):
7         print("I am a function in a class")
8
9 p1 = Person("John", 36)
10 p1.myfunc()
11
12

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

kali@kali:~/Desktop/Flex-Bison-Python-master$ ./myParser Testcases/Success/class.py
>> I am a function in a class
Success! You are awesome.
kali@kali:~/Desktop/Flex-Bison-Python-master$
```

Αποτυχημένες προσπάθειες:

The screenshot shows the Visual Studio Code interface with the file explorer on the left. The 'EXPLORER' panel shows the project structure: 'FLEX-BISON-PYTHON-MASTER' > 'Testcases' > 'Fail' > 'class.py'. The 'class.py' file is open in the editor, showing the following code:

```
1 class Person:
2     def __init__(self, name, age):
3         .name = name
4         self.age = age
5
6     def myfunc(self):
7         print("I am a function in a class")
8
9 p1 = Person("John", 36)
10 p1.myfunc()
11
12
```

The terminal at the bottom shows the command `./myParser Testcases/Fail/class.py` and the output:

```
kali@kali:~/Desktop/Flex-Bison-Python-master$ ./myParser Testcases/Fail/class.py
Line: 3 --> Parser error
kali@kali:~/Desktop/Flex-Bison-Python-master$
```

The screenshot shows the Visual Studio Code interface with the file explorer on the left. The 'EXPLORER' panel shows the project structure: 'FLEX-BISON-PYTHON-MASTER' > 'Testcases' > 'Fail' > 'class.py'. The 'class.py' file is open in the editor, showing the following code:

```
1 class Person::
2
3     def __init__(self, name, age):
4         self.name = name
5         self.age = age
6
7     def myfunc(self):
8         print("I am a function in a class")
9
10 p1 = Person("John", 36)
11 p1.myfunc()
12
13
```

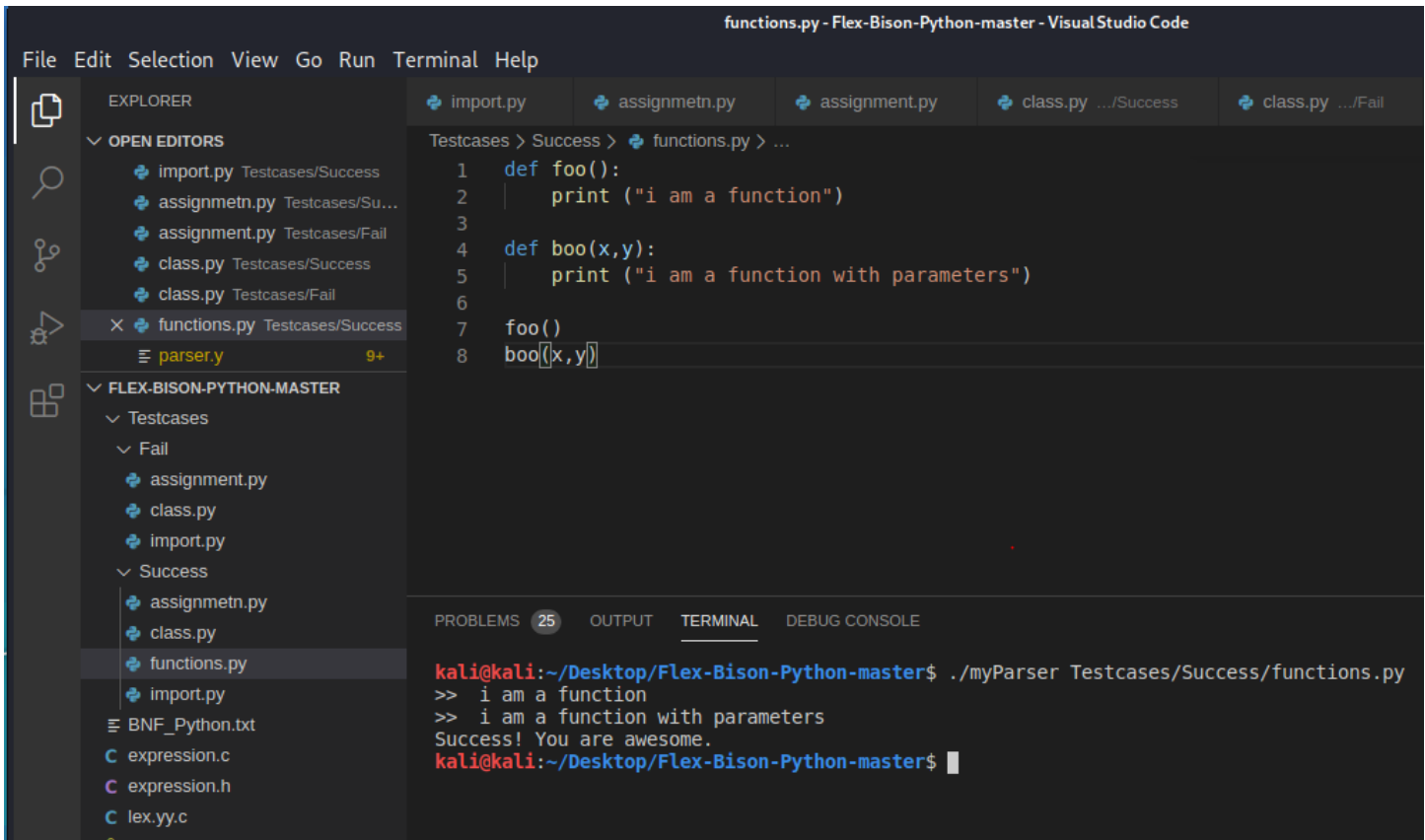
The terminal at the bottom shows the command `./myParser Testcases/Fail/class.py` and the output:

```
kali@kali:~/Desktop/Flex-Bison-Python-master$ ./myParser Testcases/Fail/class.py
Line: 1 --> Parser error
kali@kali:~/Desktop/Flex-Bison-Python-master$
```

Δ)

Ορισμός συνάρτησης και κλήση της

Επιτυχημένη προσπάθεια:



The screenshot shows the Visual Studio Code interface with the file explorer on the left, the editor in the center, and the terminal at the bottom. The file explorer shows the project structure with 'functions.py' selected under 'Testcases/Success'. The editor displays the code for 'functions.py' with two functions: 'foo()' and 'boo(x,y)'. The terminal shows the command './myParser Testcases/Success/functions.py' being executed, resulting in the output 'i am a function' and 'i am a function with parameters', followed by the message 'Success! You are awesome.'.

```
functions.py - Flex-Bison-Python-master - Visual Studio Code
```

File Edit Selection View Go Run Terminal Help

EXPLORER

- import.py
- assignmetn.py
- assignment.py
- class.py .../Success
- class.py .../Fail

OPEN EDITORS

- import.py Testcases/Success
- assignmetn.py Testcases/Su...
- assignment.py Testcases/Fail
- class.py Testcases/Success
- class.py Testcases/Fail
- functions.py Testcases/Success
- parser.y 9+

FLEX-BISON-PYTHON-MASTER

- Testcases
 - Fail
 - assignment.py
 - class.py
 - import.py
 - Success
 - assignmetn.py
 - class.py
 - functions.py
 - import.py
- BNF_Python.txt
- expression.c
- expression.h
- lex.yy.c
- LICENSE

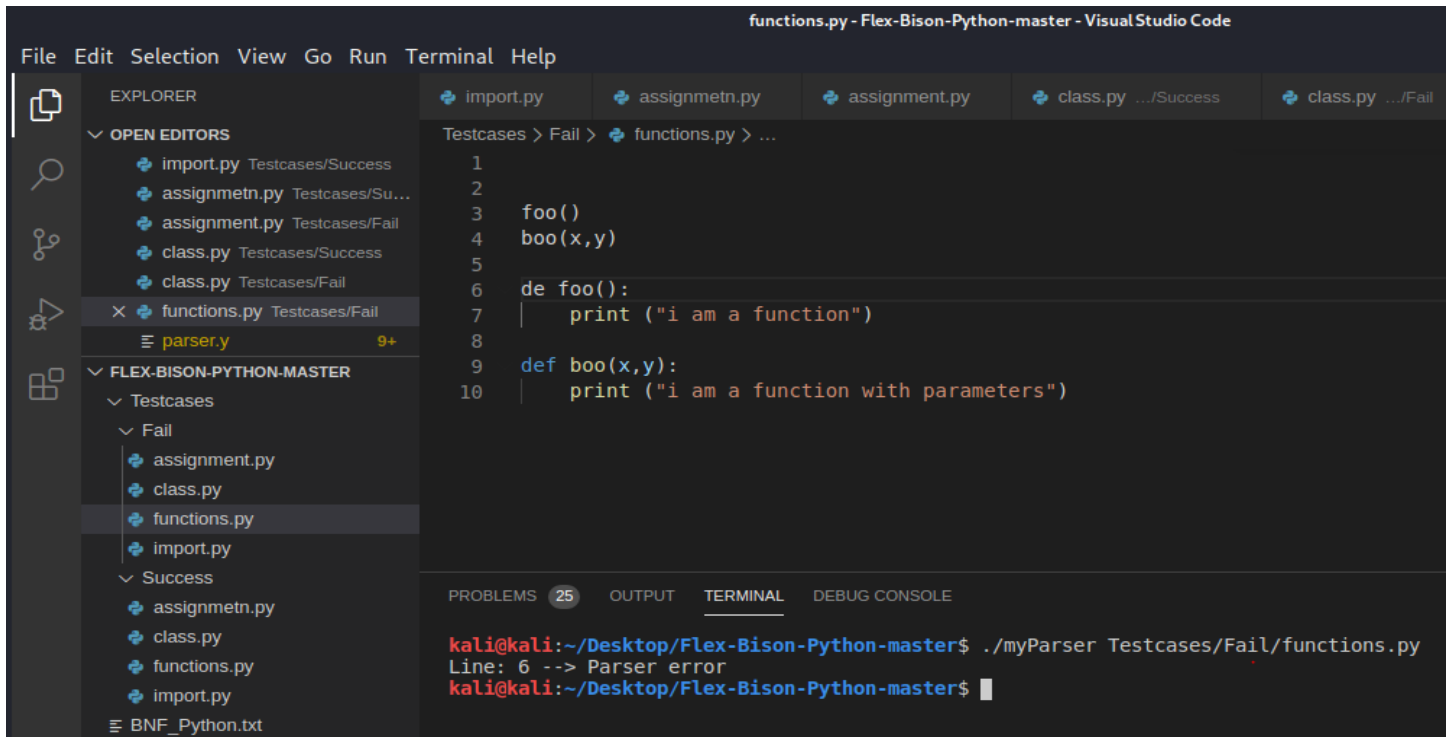
Testcases > Success > functions.py > ...

```
1 def foo():
2     print ("i am a function")
3
4 def boo(x,y):
5     print ("i am a function with parameters")
6
7 foo()
8 boo(x,y)
```

PROBLEMS 25 OUTPUT TERMINAL DEBUG CONSOLE

```
kali@kali:~/Desktop/Flex-Bison-Python-master$ ./myParser Testcases/Success/functions.py
>> i am a function
>> i am a function with parameters
Success! You are awesome.
kali@kali:~/Desktop/Flex-Bison-Python-master$
```

Αποτυχημένες προσπάθειες:

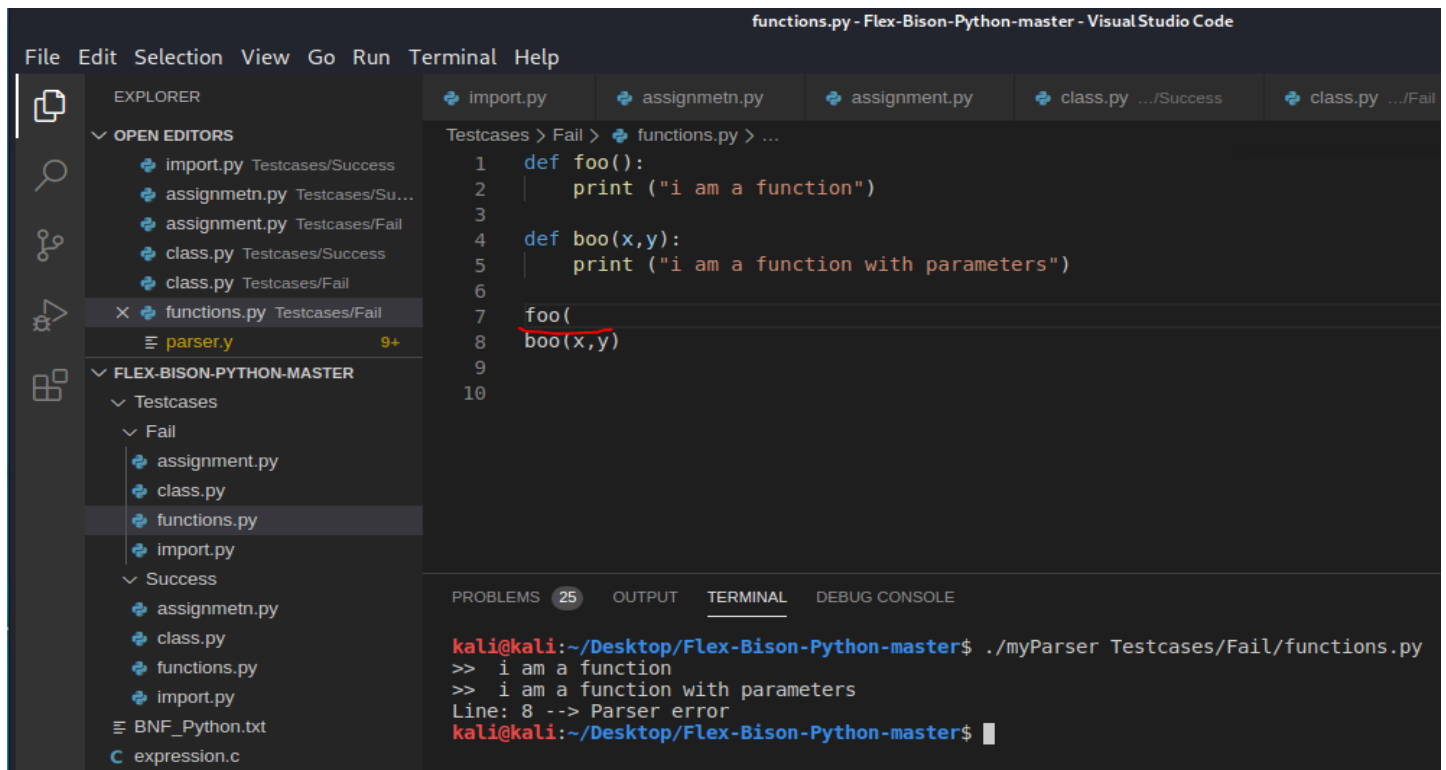


The screenshot shows the Visual Studio Code interface with the file explorer on the left. The 'EXPLORER' panel shows the project structure, including 'OPEN EDITORS' and 'FLEX-BISON-PYTHON-MASTER'. The 'FUNCTIONS.PY' file is selected under the 'Fail' category. The editor displays the following code:

```
1
2
3  foo()
4  boo(x,y)
5
6  de foo():
7      print ("i am a function")
8
9  def boo(x,y):
10     print ("i am a function with parameters")
```

The terminal at the bottom shows the command `./myParser Testcases/Fail/functions.py` and the output:

```
kali@kali:~/Desktop/Flex-Bison-Python-master$ ./myParser Testcases/Fail/functions.py
Line: 6 --> Parser error
kali@kali:~/Desktop/Flex-Bison-Python-master$
```



The screenshot shows the Visual Studio Code interface with the file explorer on the left. The 'EXPLORER' panel shows the project structure, including 'OPEN EDITORS' and 'FLEX-BISON-PYTHON-MASTER'. The 'FUNCTIONS.PY' file is selected under the 'Fail' category. The editor displays the following code:

```
1  def foo():
2      print ("i am a function")
3
4  def boo(x,y):
5      print ("i am a function with parameters")
6
7  foo(
8  boo(x,y)
9
10
```

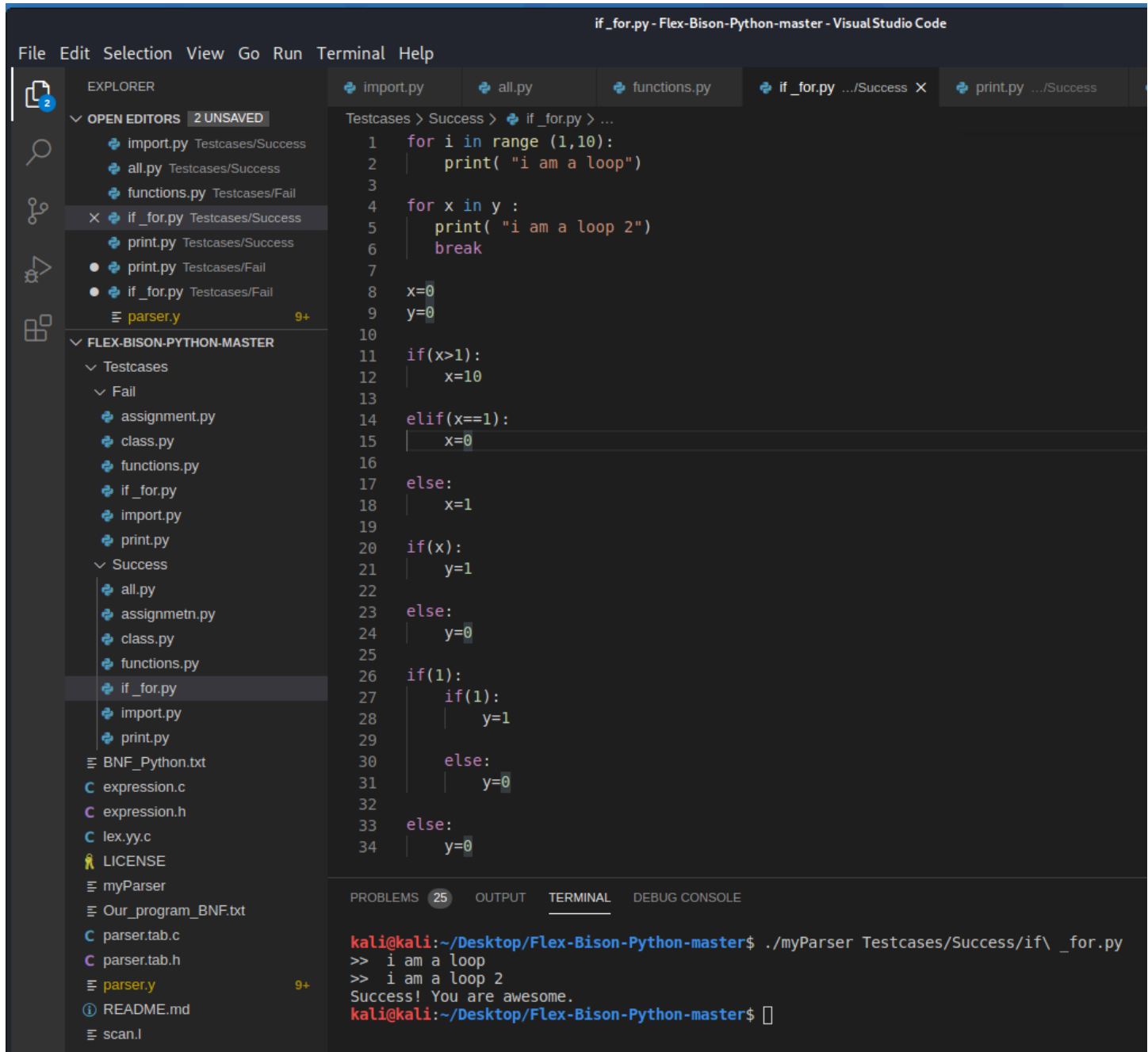
The terminal at the bottom shows the command `./myParser Testcases/Fail/functions.py` and the output:

```
kali@kali:~/Desktop/Flex-Bison-Python-master$ ./myParser Testcases/Fail/functions.py
>> i am a function
>> i am a function with parameters
Line: 8 --> Parser error
kali@kali:~/Desktop/Flex-Bison-Python-master$
```

E)

Εντολές βρόγχου και συνθήκη

Επιτυχημένη προσπάθεια:



The screenshot shows the Visual Studio Code interface with the file `if_for.py` open. The code in the editor is as follows:

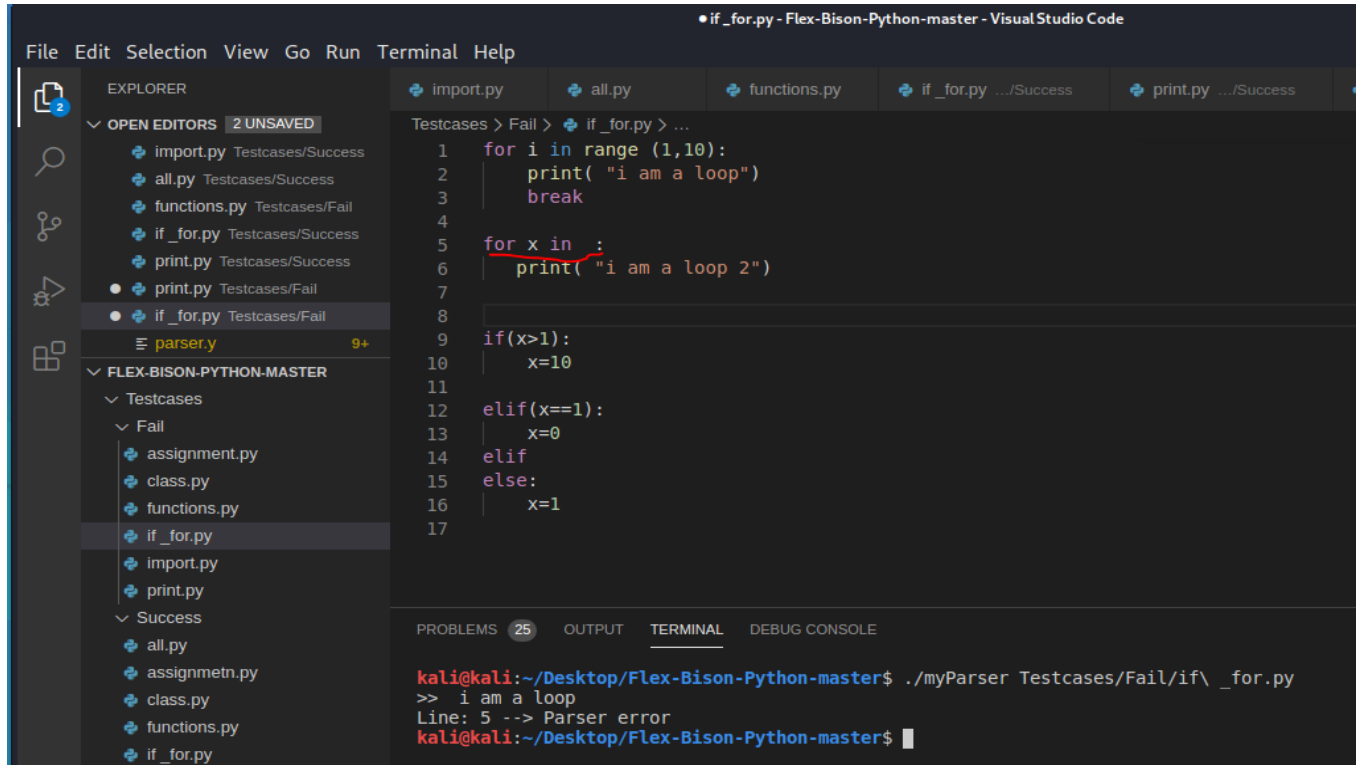
```
1 for i in range (1,10):
2     print( "i am a loop")
3
4 for x in y :
5     print( "i am a loop 2")
6     break
7
8 x=0
9 y=0
10
11 if(x>1):
12     x=10
13
14 elif(x==1):
15     x=0
16
17 else:
18     x=1
19
20 if(x):
21     y=1
22
23 else:
24     y=0
25
26 if(1):
27     if(1):
28         y=1
29
30     else:
31         y=0
32
33 else:
34     y=0
```

The Explorer sidebar on the left shows the project structure for `FLEX-BISON-PYTHON-MASTER`. The `Testcases` folder is expanded, showing subfolders `Fail` and `Success`. The `if_for.py` file is highlighted in the `Success` folder.

The Terminal at the bottom shows the command `./myParser Testcases/Success/if_for.py` being executed, with the following output:

```
kali@kali:~/Desktop/Flex-Bison-Python-master$ ./myParser Testcases/Success/if\_for.py
>> i am a loop
>> i am a loop 2
Success! You are awesome.
kali@kali:~/Desktop/Flex-Bison-Python-master$
```


Αποτυχημένες προσπάθειες:



if_for.py - Flex-Bison-Python-master - Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER

OPEN EDITORS 2 UNSAVED

- import.py Testcases/Success
- all.py Testcases/Success
- functions.py Testcases/Fail
- if_for.py Testcases/Success
- print.py Testcases/Success
- print.py Testcases/Fail
- if_for.py Testcases/Fail
- parser.y 9+

FLEX-BISON-PYTHON-MASTER

Testcases

Fail

- assignment.py
- class.py
- functions.py
- if_for.py
- import.py
- print.py

Success

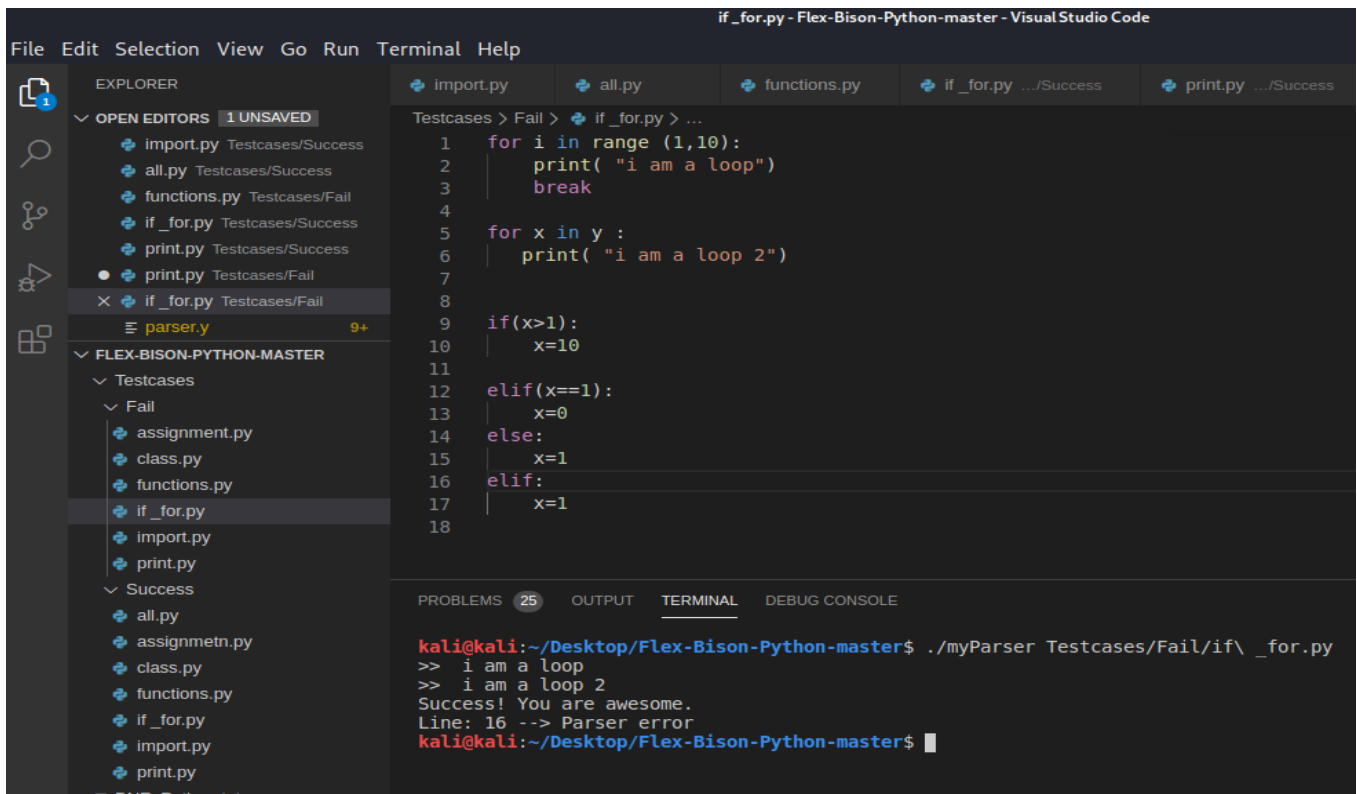
- all.py
- assignment.py
- class.py
- functions.py
- if_for.py

Testcases > Fail > if_for.py > ...

```
1 for i in range (1,10):
2     print( "i am a loop")
3     break
4
5 for x in :
6     print( "i am a loop 2")
7
8
9 if(x>1):
10     x=10
11
12 elif(x==1):
13     x=0
14 elif
15 else:
16     x=1
17
```

PROBLEMS 25 OUTPUT TERMINAL DEBUG CONSOLE

```
kali@kali:~/Desktop/Flex-Bison-Python-master$ ./myParser Testcases/Fail/if_for.py
>> i am a loop
Line: 5 --> Parser error
kali@kali:~/Desktop/Flex-Bison-Python-master$
```



if_for.py - Flex-Bison-Python-master - Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER

OPEN EDITORS 1 UNSAVED

- import.py Testcases/Success
- all.py Testcases/Success
- functions.py Testcases/Fail
- if_for.py Testcases/Success
- print.py Testcases/Success
- print.py Testcases/Fail
- if_for.py Testcases/Fail
- parser.y 9+

FLEX-BISON-PYTHON-MASTER

Testcases

Fail

- assignment.py
- class.py
- functions.py
- if_for.py
- import.py
- print.py

Success

- all.py
- assignment.py
- class.py
- functions.py
- if_for.py
- import.py
- print.py

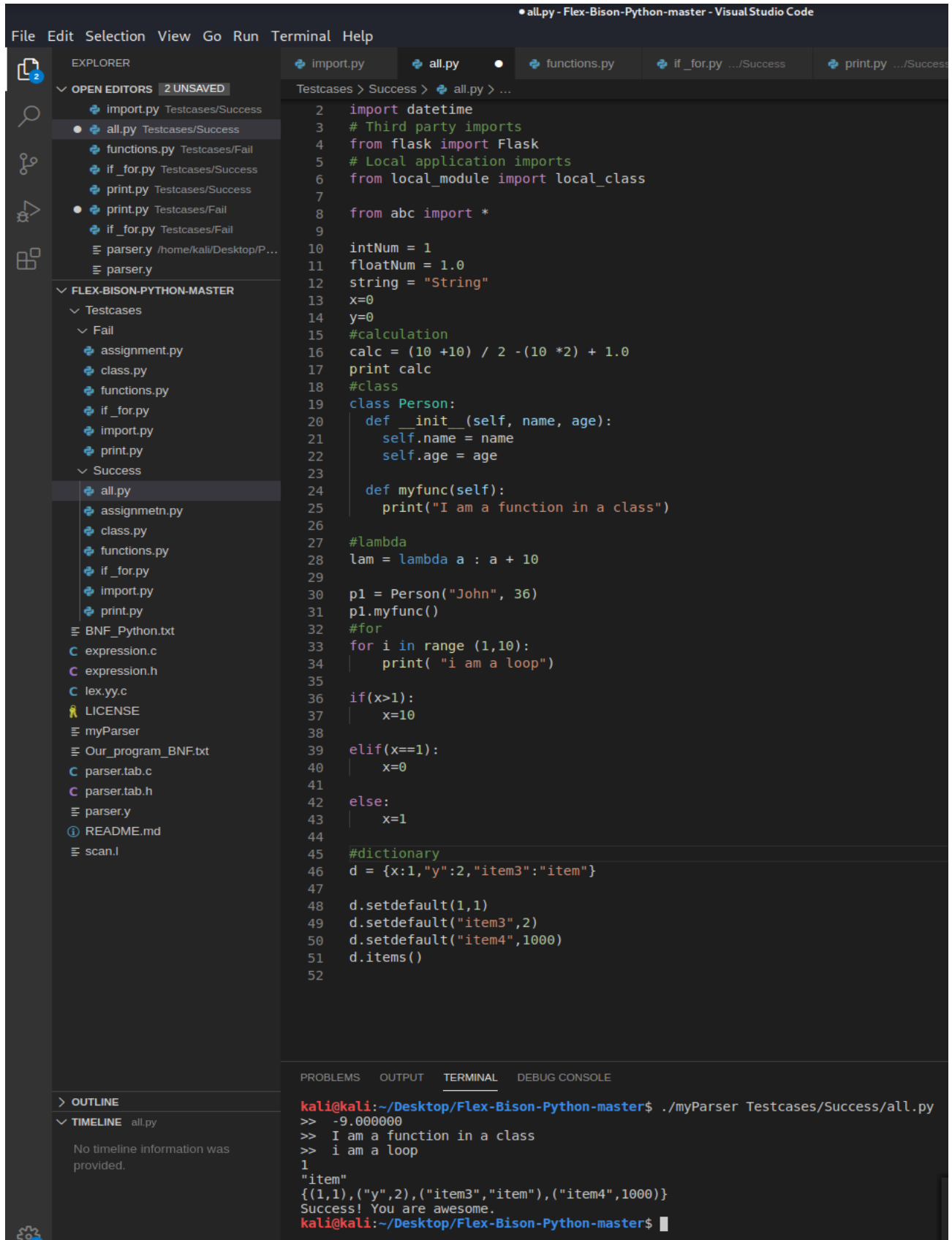
Testcases > Fail > if_for.py > ...

```
1 for i in range (1,10):
2     print( "i am a loop")
3     break
4
5 for x in y :
6     print( "i am a loop 2")
7
8
9 if(x>1):
10     x=10
11
12 elif(x==1):
13     x=0
14 else:
15     x=1
16 elif:
17     x=1
18
```

PROBLEMS 25 OUTPUT TERMINAL DEBUG CONSOLE

```
kali@kali:~/Desktop/Flex-Bison-Python-master$ ./myParser Testcases/Fail/if_for.py
>> i am a loop
>> i am a loop 2
Success! You are awesome.
Line: 16 --> Parser error
kali@kali:~/Desktop/Flex-Bison-Python-master$
```

Αρχείο με πολλαπλές εντολές (Dictionaries και Lambda)



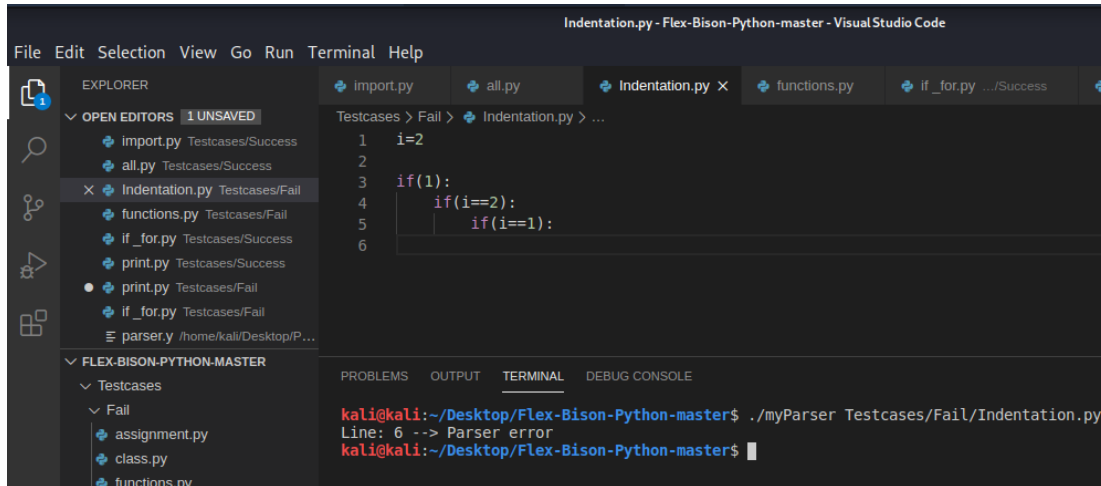
```
File Edit Selection View Go Run Terminal Help
EXPLORER
2 OPEN EDITORS 2 UNSAVED
  import.py Testcases/Success
  all.py Testcases/Success
  functions.py Testcases/Fail
  if_for.py Testcases/Success
  print.py Testcases/Success
  print.py Testcases/Fail
  if_for.py Testcases/Fail
  parser.y /home/kali/Desktop/P...
  parser.y
  FLEX-BISON-PYTHON-MASTER
    Testcases
      Fail
        assignment.py
        class.py
        functions.py
        if_for.py
        import.py
        print.py
      Success
        all.py
        assignmetn.py
        class.py
        functions.py
        if_for.py
        import.py
        print.py
    BNF_Python.txt
    expression.c
    expression.h
    lex.yy.c
    LICENSE
    myParser
    Our_program_BNF.txt
    parser.tab.c
    parser.tab.h
    parser.y
    README.md
    scan.l
    OUTLINE
    TIMELINE all.py
      No timeline information was provided.

all.py
2 import datetime
3 # Third party imports
4 from flask import Flask
5 # Local application imports
6 from local_module import local_class
7
8 from abc import *
9
10 intNum = 1
11 floatNum = 1.0
12 string = "String"
13 x=0
14 y=0
15 #calculation
16 calc = (10 +10) / 2 -(10 *2) + 1.0
17 print calc
18 #class
19 class Person:
20     def __init__(self, name, age):
21         self.name = name
22         self.age = age
23
24     def myfunc(self):
25         print("I am a function in a class")
26
27 #lambda
28 lam = lambda a : a + 10
29
30 p1 = Person("John", 36)
31 p1.myfunc()
32 #for
33 for i in range (1,10):
34     print( "i am a loop")
35
36 if(x>1):
37     x=10
38
39 elif(x==1):
40     x=0
41
42 else:
43     x=1
44
45 #dictionary
46 d = {x:1,"y":2,"item3":"item"}
47
48 d.setdefault(1,1)
49 d.setdefault("item3",2)
50 d.setdefault("item4",1000)
51 d.items()
52

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
kali@kali:~/Desktop/Flex-Bison-Python-master$ ./myParser Testcases/Success/all.py
>> -9.000000
>> I am a function in a class
>> i am a loop
1
"item"
{(1,1),("y",2),("item3","item"),("item4",1000)}
Success! You are awesome.
kali@kali:~/Desktop/Flex-Bison-Python-master$
```

Python Indentation

Στα προηγούμενα παραδείγματα παρουσιάζεται η σωστή εκτέλεση του indentation στις εντολές if. Παρακάτω αναφέρονται περιπτώσεις που απαιτούν την τύπωση σφάλματος.

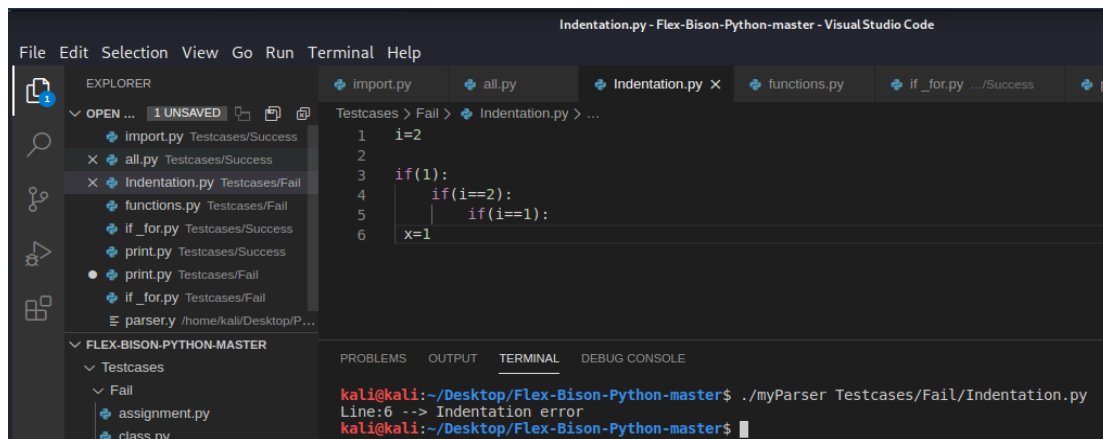


The screenshot shows the Visual Studio Code interface with the file 'Indentation.py' open. The code in the editor is as follows:

```
1 i=2
2
3 if(1):
4     if(i==2):
5         if(i==1):
6
```

The terminal output shows the command `./myParser Testcases/Fail/Indentation.py` being executed, resulting in a 'Parser error' on line 6.

```
kali@kali:~/Desktop/Flex-Bison-Python-master$ ./myParser Testcases/Fail/Indentation.py
Line: 6 --> Parser error
kali@kali:~/Desktop/Flex-Bison-Python-master$
```

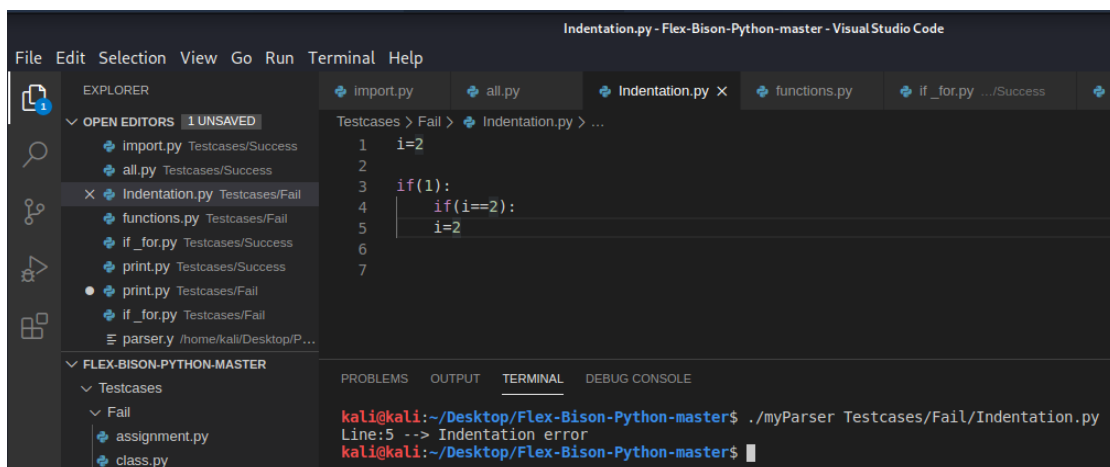


The screenshot shows the Visual Studio Code interface with the file 'Indentation.py' open. The code in the editor is as follows:

```
1 i=2
2
3 if(1):
4     if(i==2):
5         if(i==1):
6     x=1
```

The terminal output shows the command `./myParser Testcases/Fail/Indentation.py` being executed, resulting in an 'Indentation error' on line 6.

```
kali@kali:~/Desktop/Flex-Bison-Python-master$ ./myParser Testcases/Fail/Indentation.py
Line:6 --> Indentation error
kali@kali:~/Desktop/Flex-Bison-Python-master$
```



The screenshot shows the Visual Studio Code interface with the file 'Indentation.py' open. The code in the editor is as follows:

```
1 i=2
2
3 if(1):
4     if(i==2):
5     i=2
6
7
```

The terminal output shows the command `./myParser Testcases/Fail/Indentation.py` being executed, resulting in an 'Indentation error' on line 5.

```
kali@kali:~/Desktop/Flex-Bison-Python-master$ ./myParser Testcases/Fail/Indentation.py
Line:5 --> Indentation error
kali@kali:~/Desktop/Flex-Bison-Python-master$
```

```
Indentation.py - Flex-Bison-Python-master - VisualStudio Code

File Edit Selection View Go Run Terminal Help

EXPLORER
  OPEN EDITORS 1 UNSAVED
    Import.py Testcases/Success
    all.py Testcases/Success
    X Indentation.py Testcases/Fail
    functions.py Testcases/Fail
    if_for.py Testcases/Success
    print.py Testcases/Success
    print.py Testcases/Fail
    if_for.py Testcases/Fail
    parser.y /home/kali/Desktop/P...

  FLEX-BISON-PYTHON-MASTER
    Testcases
      Fail
        assignment.py
        class.py

Testcases > Fail > Indentation.py > ...
1 i=2
2
3 if(1):
4     if(i==2):
5         if(i==1):
6
7         x=1

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

kali@kali:~/Desktop/Flex-Bison-Python-master$ ./myParser Testcases/Fail/Indentation.py
Line:7 --> Indentation error
kali@kali:~/Desktop/Flex-Bison-Python-master$
```

Παρουσίαση Σημασιολογικής Λειτουργικότητας

Παρακάτω θα δείξουμε ορισμένες περιπτώσεις που ο compiler υλοποιεί και την σημασιολογική ανάλυση όπως:

- A) Ανίχνευση μη ορισμένων μεταβλητών.
- B) Ανίχνευση πράξεων που δεν επιτρέπονται (πχ. Integer με String).
- Γ) Ανίχνευση πράξεων μεταξύ integer και float και μετατροπή του αποτελέσματος σε τιμή float.

A)

```
senmatic.py - Flex-Bison-Python-master - VisualStudio Code

File Edit Selection View Go Run Terminal Help

EXPLORER
  OPEN EDITORS 1 UNSAVED
    print.py Testcases/Fail
    if_for.py Testcases/Fail
    parser.y /home/kali/Desktop/P...
    parser.y 9+
    X senmatic.py Testcases/Fail

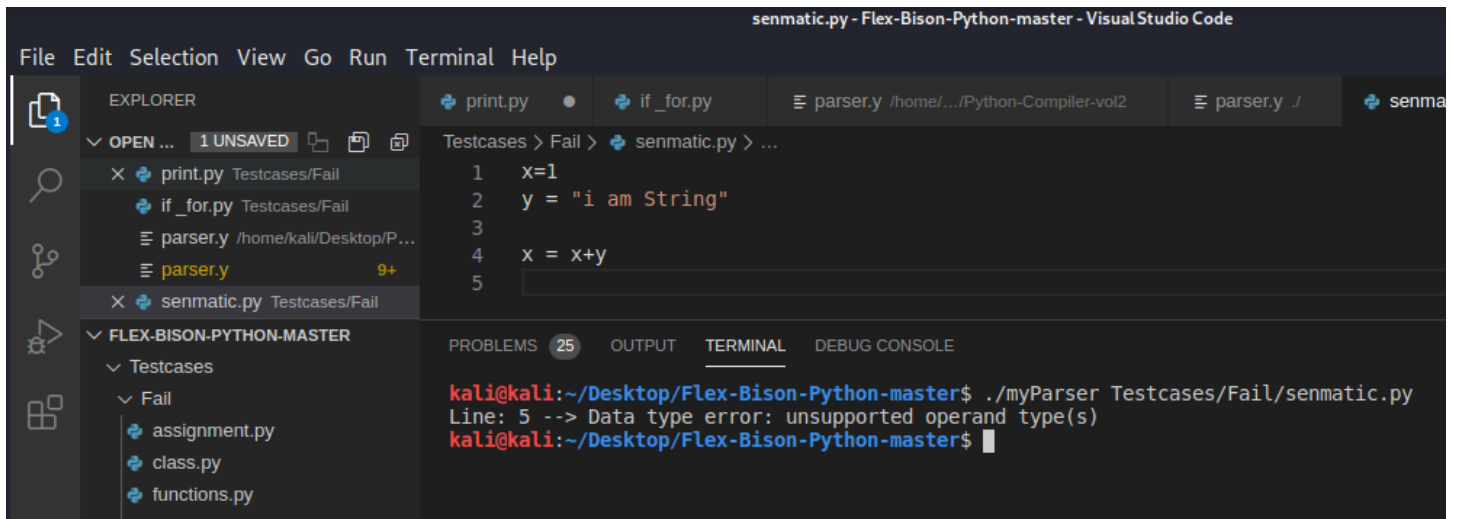
  FLEX-BISON-PYTHON-MASTER
    Testcases
      Fail
        assignment.py
        class.py
        functions.py
        if_for.py

Testcases > Fail > senmatic.py > ...
1 x=1
2 print([x,y])

PROBLEMS 25 OUTPUT TERMINAL DEBUG CONSOLE

kali@kali:~/Desktop/Flex-Bison-Python-master$ ./myParser Testcases/Fail/senmatic.py
Line: 2 --> Error: Variable y has not been defined
kali@kali:~/Desktop/Flex-Bison-Python-master$
```

B)

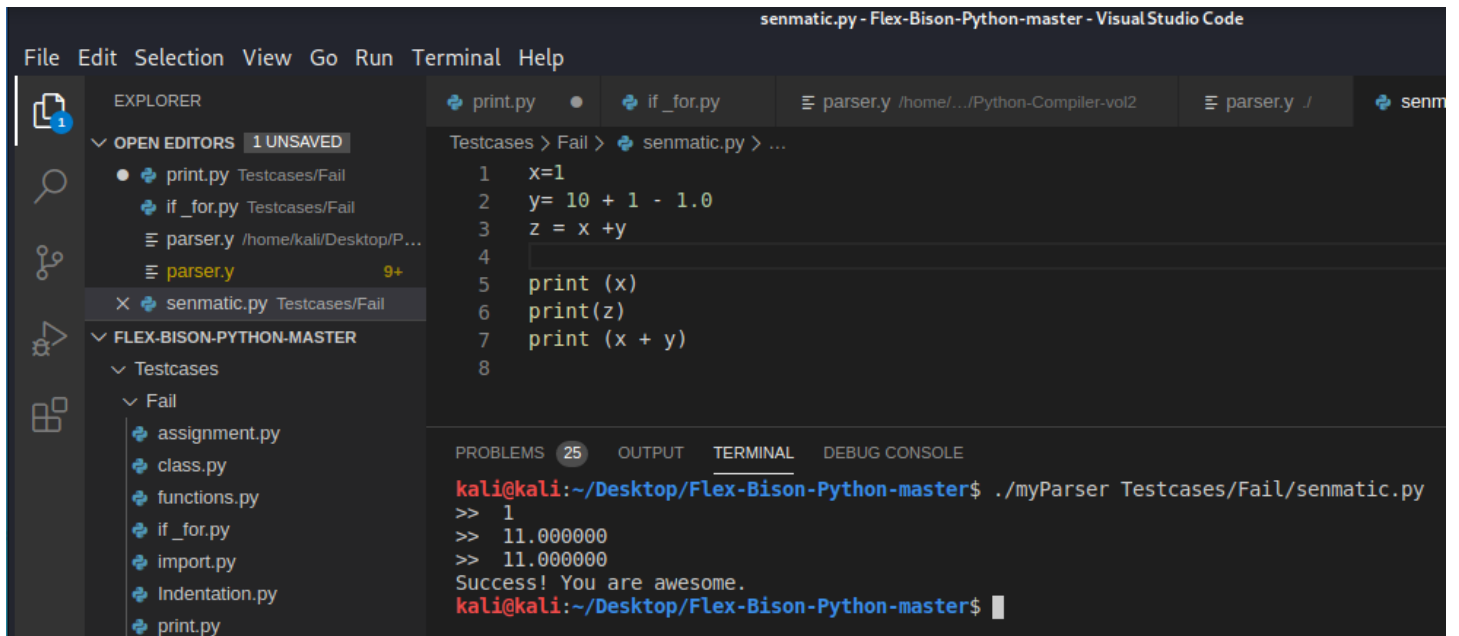


The screenshot shows the Visual Studio Code interface with the file explorer on the left, the editor in the center, and the terminal at the bottom. The file explorer shows a project named 'FLEX-BISON-PYTHON-MASTER' with a subdirectory 'Testcases' containing a 'Fail' folder. The editor displays the file 'senmatic.py' with the following code:

```
1 x=1
2 y = "i am String"
3
4 x = x+y
5
```

The terminal shows the command `./myParser Testcases/Fail/senmatic.py` being executed, resulting in the error message: `Line: 5 --> Data type error: unsupported operand type(s)`.

Γ)



The screenshot shows the Visual Studio Code interface with the file explorer on the left, the editor in the center, and the terminal at the bottom. The file explorer shows a project named 'FLEX-BISON-PYTHON-MASTER' with a subdirectory 'Testcases' containing a 'Fail' folder. The editor displays the file 'senmatic.py' with the following code:

```
1 x=1
2 y= 10 + 1 - 1.0
3 z = x + y
4
5 print (x)
6 print(z)
7 print (x + y)
8
```

The terminal shows the command `./myParser Testcases/Fail/senmatic.py` being executed, resulting in the output: `>> 1`, `>> 11.000000`, `>> 11.000000`, and `Success! You are awesome.`

Διευκρινήσεις σχετικά με τα warnings

```
kali@kali:~/Desktop/Flex-Bison-Python-master$ bison -d parser.y
parser.y:144.9-31: warning: type clash on default action: <nval> != <> [-Wother]
144 |         LAMBDA COLON expression
    |         ^~~~~~
parser.y:145.11-48: warning: type clash on default action: <nval> != <> [-Wother]
145 |         | LAMBDA lam_parameters COLON expression
    |         ^~~~~~
parser.y:160.17-21: warning: type clash on default action: <nval> != <> [-Wother]
160 |         PRINT
    |         ^~~~~
parser.y:483.9-17: warning: type clash on default action: <nval> != <> [-Wother]
483 |         LBRA RBRA
    |         ^~~~~~
parser.y:484.11-34: warning: type clash on default action: <nval> != <> [-Wother]
484 |         | LBRA key_datum_list RBRA;
    |         ^~~~~~
parser.y: warning: 501 shift/reduce conflicts [-Wconflicts-sr]
parser.y: warning: 157 reduce/reduce conflicts [-Wconflicts-rr]
kali@kali:~/Desktop/Flex-Bison-Python-master$
```

Σχετικά με τα warnings που εμφανίζονται στην γραμμή 144, 145, 160, 483, 484 αφορούν τις περιπτώσεις που ορισμένα τερματικά και μη τερματικά σύμβολα δεν έχουν λάβει τον τύπο δεδομένων “nval” ενώ οι κανόνες που περιέχουν ανήκουν σε αυτό τον τύπο δεδομένων.

Τέλος, για τις προειδοποιήσεις της γραμμής 501 και 157 συμβαίνουν εάν υπάρχουν δύο ή περισσότεροι κανόνες που ισχύουν για την ίδια ακολουθία εισόδου.