Laporan Tugas Besar IF2124
Teori Bahasa Formal dan Otomata

Link Github : bayusamudra5502/Python-Syntax-Checker

Python Syntax Checker

oleh

Kelompok 2 K03 (Lelah TUbes :V)

Bayu Samudra (13520128)
Febryola Kurnia Putri (13520140)
Aloysius Gilang Pramudya (13520147)

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG

2021

DAFTAR ISI

BAB I
Deskripsi Masalah

Python adalah bahasa *interpreter* tingkat tinggi (*high-level*), dan juga *general-purpose*. Python diciptakan oleh Guido van Rossum dan dirilis pertama kali pada tahun 1991. Filosofi desain pemrograman Python mengutamakan *code readability* dengan penggunaan *whitespace*-nya. Python adalah bahasa multiparadigma karena mengimplementasi paradigma fungsional, imperatif, berorientasi objek, dan reflektif.

Dalam proses pembuatan program dari sebuah bahasa menjadi instruksi yang dapat dieksekusi oleh mesin, terdapat pemeriksaan sintaks atau kompilasi bahasa yang dibuat oleh programmer. Kompilasi ini bertujuan untuk memastikan instruksi yang dibuat oleh programmer mengikuti aturan yang sudah ditentukan oleh bahasa tersebut. Baik bahasa berjenis *interpreter*maupun *compiler*, keduanya pasti melakukan pemeriksaan sintaks. Perbedaannya terletak pada apa yang dilakukan setelah proses pemeriksaan (kompilasi/*compile*) tersebut selesai dilakukan.

Dibutuhkan *grammar* bahasa dan algoritma *parser* untuk melakukan kompilasi. Sudah sangat banyak *grammar* dan algoritma yang dikembangkan untuk menghasilkan *compiler* dengan performa yang tinggi. Terdapat CFG, CNF-e, CNF+e, 2NF, 2LF, dll untuk *grammar* yang dapat digunakan, dan terdapat LL(0), LL(1), CYK, Earley's Algorithm, LALR, GLR, Shift-reduce, SLR, LR(1), dll untuk algoritma yang dapat digunakan untuk melakukan *parsing*.

Pada tugas besar ini, implementasikanlah ***compiler* untuk Python** untuk *statement-statement* dan sintaks-sintaks bawaan Python. Gunakanlah konsep **CFG** untuk pengerjaan **compiler yang mengevaluasi syntax program**. Untuk **nama variabel** dalam program**,** gunakanlah **FA**.

Algoritma yang dipakai dibebaskan, namun tim asisten menyarankan menggunakan algoritma **CYK (Cocke-Younger-Kasami)**. Algoritma CYK harus menggunakan *grammar* CNF (Chomsky Normal Form) sebagai *grammar* masukannya. Oleh karena itu, jika ingin menggunakan CYK buatlah **terlebih dahulu *grammar* dalam CFG (Context Free Grammar), kemudian konversikan *grammar* CFG tersebut ke *grammar* CNF.** Berikut adalah daftar kata kunci bawaan Python yang harus terdaftar dalam *grammar* (yang dicoret tidak perlu diimplementasi). Rincian mengenai implementasi dan contohnya dapat dilihat pada pranala ini.

False	class	~~finally~~	is	return
None	continue	for	~~lambda~~	~~try~~
True	def	from	~~nonlocal~~	while

and	~~del~~	~~global~~	not	with
as	elif	if	or	~~yield~~
~~assert~~	else	import	pass	
break	~~except~~	in	raise	

BAB II
Teori Singkat

2.1. Python

Python adalah sebuah bahasa pemrograman yang diciptakan oleh Guido van Rossum. Bahasa pemrograman yang terinspirasi oleh bahasa pemrograman ABC ini diciptakan pada awal tahun 1990-an saat Guido van Rossum masih berada di *Centrum Wiskunde & Informatica* (CWI) di Belanda.

2.2. Context-Free Grammar (CFG)

Context-Free Grammar (CFG) adalah salah satu tipe grammar yang digunakan untuk menerima sebuah bahasa/*language*. Bahasa/*language* yang diterima oleh sebuah CFG disebut sebagai Context-Free Language (CFL). CFG didefinisikan sebagai quadtuple, yaitu

G = (V, T, P, S)

dengan

G	= Context-Free Grammar
V	= Simbol Non-Terminal
T	= Simbol Terminal
P	= Set Produksi
S	= *Start Symbol*

Set produksi pada sebuah CFG dapat dikelompokkan menjadi 3 jenis, yaitu:

1) Produksi dari sebuah simbol non-terminal menjadi satu atau kumpulan simbol terminal (A → a|ac|abc|ba)
2) Produksi dari simbol non-terminal menjadi satu atau kumpulan simbol non-terminal (A → A|AB|ABC|BC)
3) Produksi dari simbol non-terminal menjadi gabungan antara simbol non-terminal dan simbol terminal (A → aB|ACb|Abc)

2.3. Chomsky Normal Form (CNF)

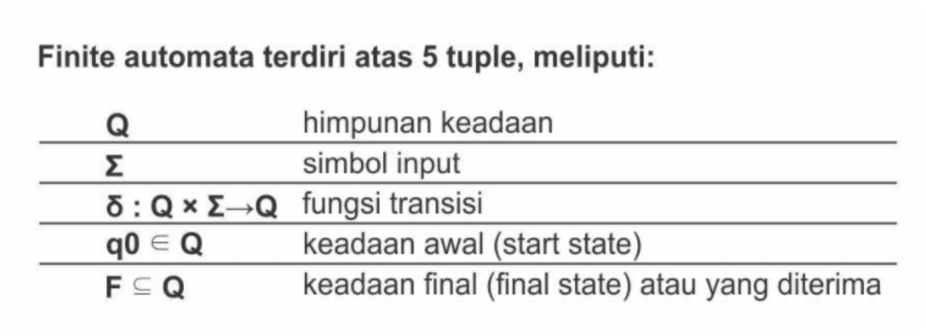
Chomsky Normal Form (CNF) adalah sebuah bentuk CFG yang hanya memiliki dua jenis set produksi, yaitu:

1) A → BC
2) A→ a

dengan A, B, dan C adalah variabel, dan a adalah terminal.

2.4. Finite Automata (FA)

Finite automata adalah model yang baik untuk komputer yang memiliki jumlah atau kapasitas memori terbatas. Salah satu contoh yang dapat dilakukan dari model finite automata adalah pintu otomatis yang dapat buka tutup tanpa menggunakan bantuan dari manusia.



Finite automata terdiri atas 5 tuple, meliputi:

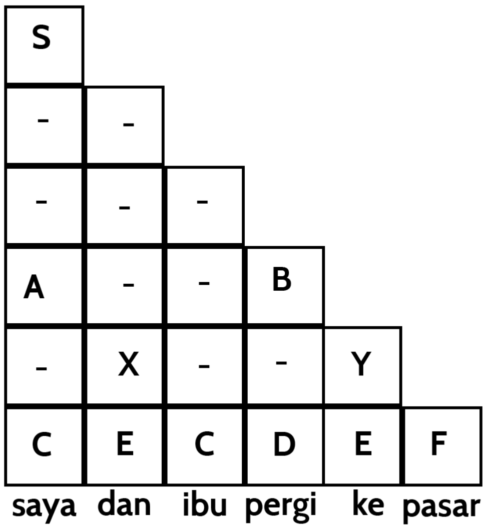
Q	himpunan keadaan
Σ	simbol input
$\delta : Q \times \Sigma \to Q$	fungsi transisi
$q0 \in Q$	keadaan awal (start state)
$F \subseteq Q$	keadaan final (final state) atau yang diterima

2.5. Algoritma Cocke-Younger-Kasami (CYK)

Algoritma Cocke-Younger-Kasami (CYK) adalah sebuah algoritma yang dapat digunakan untuk mengecek apakah sebuah string dapat dihasilkan dari set produksi dari Chomsky Normal Form (CNF). Algoritma ini dinamakan berdasarkan nama tiga orang penciptanya, yaitu John Cocke, Daniel Younger, serta Tadao Kasami. Pembuatan algoritma CYK dapat dibantu menggunakan tabel CYK.

Produksi pada CFG:

- S → AB
- A → CX
- B → DY
- X → EC
- Y → EF
- C → saya
- C → ibu
- E → dan
- E → ke
- D → pergi
- F → pasar



S					
-	-				
-	-	-			
A	-	-	B		
-	X	-	-	Y	
C	E	C	D	E	F
saya	dan	ibu	pergi	ke	pasar

Gambar 1.1 Contoh Tabel CYK dan Penggunaannya

Bila menggunakan Tabel CYK, maka sebuah string yang diuji diterima oleh sebuah CFG bila pada tabel paling atas merupakan *Start Symbol* dari CFG yang bersangkutan.

BAB III
Analisis Persoalan

3.1. Grammar CFG

Berikut ini adalah production grammar dari CFG yang kami gunakan dalam membentuk python linter.

G = (V, T, P, S)

Start Symbol (S) : Start

Terminal (T) :

```
.
,
+
-
*
/
[
]
{
}
(
)
'
"
:
;
=
!
>
<
%
~
&
if
elif
else
for
in
is
while
continue
break
pass
class
number
return
def
with
raise
import
as
from
string
and
or
not
true
```

```
false
none
}
^
\
```

Production Symbol (P) :

```
START -> TEXTED
TEXTED -> TEXTED TEXTED | TEXTED | TEXTED | CLASS_TEXTED | DEF_FUNC_TEXTED |
ASS_OPERATION | FLOW_TEXTED | IMP_OPERATION | RAI_OP | FUNCTION | METH_TEXTED |
WITH_TEXTED | PASS | STRING | LIST | DICT | VARIABLE | CONSTANT | NONE | COND_OPERATION
| IF_TEXTED | ITERATE_TEXTED | COMMENT
ITERATE_TEXTED -> FOR_TEXTED | WHILE_TEXTED | LIST | DICT | VARIABLE | CONSTANT
FLOW_TEXTED -> BREAK_TEXTED | CONTINUE_TEXTED

// SYMBOLS
DOT -> .
PLUS -> +
MINUS -> -
MULTIPLY -> *
DIVISION -> /
OPEN_SQUARE_BRACKET -> [
CLOSE_SQUARE_BRACKET -> ]
OPEN_CURLY_BRACKET -> {
CLOSE_CURLY_BRACKET -> }
OPEN_NORMAL_BRACKET -> (
CLOSE_NORMAL_BRACKET -> )
SINGLE_QUOTE -> '
DOUBLE_QUOTE -> "
COLON -> :
SEMICOLON -> ;
COMMA -> ,
SPACE -> SPACE SPACE
OR_SYM -> or
BANG_SYM -> !
GT_SYM -> <
LT_SYM -> >
EQ_SYM -> =
TILDE_SYM -> ~
PERCENT_SYM -> %
CARET_SYM -> ^
BACKSLASH -> \
REL_OP -> EQ_SYM EQ_SYM | BANG_SYM EQ_SYM | GT_SYM EQ_SYM | LT_SYM EQ_SYM | GT_SYM |
LT_SYM | IS
ASS_OP -> EQ_SYM | PLUS EQ_SYM | MINUS EQ_SYM | MULTIPLY EQ_SYM | MULTIPLY MULTIPLY
EQ_SYM | DIVISION EQ_SYM | DIVISION DIVISION EQ_SYM | PERCENT_SYM EQ_SYM
ART_OP -> PLUS | MINUS | MULTIPLY | MULTIPLY MULTIPLY | DIVISION | DIVISION DIVISION |
PERCENT_SYM
LOG_OP -> AND | OR
MEM_OP -> IN | NOT SPACE IN
IDN_OP -> IS | IS SPACE NOT
BIT_OP -> & | OR_SYM | CARET_SYM | LT_SYM LT_SYM | GT_SYM GT_SYM | TILDE_SYM

// CONDITIONAL
IF -> if
ELIF -> elif
ELSE -> else
IF_HEADER -> IF COND_OPERATION COLON
IF_TEXTED -> IF_TEXTED ELIF_TEXTED | IF_TEXTED ELSE_TEXTED | IF_HEADER TEXTED |
IF_HEADER COMMENT
ELIF_HEADER -> ELIF COND_OPERATION COLON
ELIF_TEXTED -> ELIF_TEXTED ELIF_TEXTED | ELIF_TEXTED ELSE_TEXTED | ELIF_HEADER TEXTED
ELSE_HEADER -> ELSE COLON
ELSE_TEXTED -> ELSE_HEADER TEXTED
```

```
// CONDITIONAL FUNCTION
IF_TEXTED_FUNC -> IF_HEADER TEXTED_FUNC | IF_TEXTED_FUNC ELIF_TEXTED_FUNC |
IF_TEXTED_FUNC ELSE_TEXTED_FUNC
ELIF_TEXTED_FUNC -> ELIF_HEADER TEXTED_FUNC | ELIF_TEXTED_FUNC ELIF_TEXTED_FUNC |
ELIF_TEXTED_FUNC ELSE_TEXTED_FUNC
ELSE_TEXTED_FUNC -> ELSE_HEADER TEXTED_FUNC

// CONDITIONAL OPERATION
COND_OPERATION -> OPEN_NORMAL_BRACKET COND_OPERATION CLOSE_NORMAL_BRACKET | COND_OPERAND
COND_OPERATOR COND_OPERATION | COND_OPERAND
COND_OPERATOR -> REL_OP | LOG_OP | MEM_OP | IDN_OP
COND_OPERAND -> NOT COND_OPERAND | VARIABLE | CONSTANT | ART_OPERATION | METH_TEXTED |
FUNCTION | LIST | STRING | NONE | OPEN_NORMAL_BRACKET COND_OPERAND CLOSE_NORMAL_BRACKET

// LOOP FOR
FOR -> for
FOR_HEADER -> FOR FOR_VARIABLE IN ITERABLE COLON
FOR_TEXTED -> FOR_HEADER TEXTED
FOR_VARIABLE -> VARIABLE | VARIABLE COMMA FOR_VARIABLE
ITERABLE -> VARIABLE | FUNCTION | METH_TEXTED | DICT | STRING | LIST
IN -> in

// LOOP FOR FUNCTION
FOR_TEXTED_FUNC -> FOR_HEADER TEXTED_FUNC

// LOOP WHILE
WHILE -> while
WHILE_HEADER -> WHILE COND_OPERATION COLON
WHILE_TEXTED -> WHILE_HEADER TEXTED

// LOOP WHILE FUNCTION
WHILE_TEXTED_FUNC -> WHILE_HEADER TEXTED_FUNC

// OTHER LOOP
CONTINUE -> continue
CONTINUE_TEXTED -> ITERATE CONTINUE | ITERATE CONTINUE TEXTED
BREAK -> break
BREAK_TEXTED -> ITERATE BREAK | ITERATE BREAK TEXTED
ITERATE -> FOR_HEADER | FOR_TEXTED | WHILE_HEADER | WHILE_TEXTED

// OTHER LOOP FUNCTION
CONTINUE_TEXTED_FUNC -> ITERATE_FUNC CONTINUE | ITERATE CONTINUE TEXTED_FUNC
BREAK_TEXTED_FUNC -> ITERATE_FUNC BREAK | ITERATE_FUNC BREAK TEXTED_FUNC
ITERATE_FUNC -> FOR_HEADER | FOR_HEADER TEXTED_FUNC | WHILE_HEADER | WHILE_HEADER
TEXTED_FUNC

// PASS
PASS -> pass

// CLASS
CLASS -> class
CLASS_HEADER -> CLASS NAME COLON | CLASS NAME OPEN_NORMAL_BRACKET CLASS_PARAM
CLOSE_NORMAL_BRACKET COLON
CLASS_TEXTED -> CLASS_HEADER CLASS_BODY
CLASS_PARAM -> NAME | CLASS_PARAM COMMA CLASS_PARAM
CLASS_BODY -> COMMENT | IF_TEXTED | ITERATE_TEXTED | CLASS_TEXTED | DEF_FUNC_TEXTED |
ASS_OPERATION | FLOW_TEXTED | IMP_OPERATION | FUNCTION | METH_TEXTED | PASS | STRING |
CLASS_BODY CLASS_BODY | CLASS_BODY

// NAME
NAME -> name

// DEF
DEF -> def
DEF_FUNC_HEADER -> DEF NAME OPEN_NORMAL_BRACKET DEF_FUNC_PARAM CLOSE_NORMAL_BRACKET
COLON | DEF NAME OPEN_NORMAL_BRACKET CLOSE_NORMAL_BRACKET COLON
DEF_FUNC_TEXTED -> DEF_FUNC_HEADER DEF_FUNC_BODY
DEF_FUNC_BODY -> DEF_FUNC_BODY DEF_FUNC_BODY | TEXTED_FUNC | COMMENT
```

```
DEF_FUNC_RETURN -> RETURN ART_OPERATION | RETURN COND_OPERATION | RETURN
DEF_FUNC_RETURN_VAL | RETURN
DEF_FUNC_RETURN_VAL -> METH_TEXTED | FUNCTION | VARIABLE | LIST | STRING | DICT |
CONSTANT | NONE
DEF_FUNC_PARAM -> DEF_FUNC_PARAM COMMA DEF_FUNC_PARAM | NAME

// RETURN
RETURN -> return

// TEXTED
TEXTED_FUNC -> TEXTED_FUNC TEXTED_FUNC | TEXTED_FUNC | IF_TEXTED_FUNC | FOR_TEXTED_FUNC
| WHILE_TEXTED_FUNC | CONTINUE_TEXTED_FUNC | BREAK_TEXTED_FUNC | CLASS_TEXTED |
DEF_FUNC_TEXTED | ASS_OPERATION | IMP_OPERATION | RAI_OP | FUNCTION | METH_TEXTED |
WITH_TEXTED_FUNC | PASS | STRING | DEF_FUNC_RETURN | LIST | DICT | VARIABLE | CONSTANT |
NONE | COMMENT

// WITH
WITH -> with
WITH_TEXTED -> WITH_HEADER TEXTED
WITH_HEADER -> WITH WITH_ST AS VARIABLE COLON
WITH_ST -> FUNCTION | METH_TEXTED_FUNC

// WITH FUNCTION
WITH_TEXTED_FUNC -> WITH_HEADER TEXTED_FUNC

// NUMBER
NUMBER -> PLUS NUMBER | MINUS NUMBER | NUMBER_CTN
POSITIVE_NUMBER -> PLUS POSITIVE_NUMBER | NUMBER_CTN
NUMBER_CTN -> number
CONSTANT -> OPEN_NORMAL_BRACKET CONSTANT CLOSE_NORMAL_BRACKET | CON_CTN
CON_CTN -> TRUE | FALSE | NUMBER

// VARIABEL
VARIABLE -> OPEN_NORMAL_BRACKET VARIABLE CLOSE_NORMAL_BRACKET | VAR_CTN
IDX -> VARIABLE | CONSTANT | ART_OPERATION
VAR_CTN -> NAME | NAME VAR_IDX | METH_TEXTED_NAME | METH_TEXTED_NAME VAR_IDX
VAR_IDX -> VAR_IDX VAR_IDX | OPEN_SQUARE_BRACKET IDX CLOSE_SQUARE_BRACKET |
OPEN_SQUARE_BRACKET COLON OPEN_SQUARE_BRACKET | OPEN_SQUARE_BRACKET IDX COLON
CLOSE_SQUARE_BRACKET | OPEN_SQUARE_BRACKET COLON IDX CLOSE_SQUARE_BRACKET |
OPEN_SQUARE_BRACKET IDX COLON IDX CLOSE_SQUARE_BRACKET | OPEN_SQUARE_BRACKET COLON COLON
CLOSE_SQUARE_BRACKET | OPEN_SQUARE_BRACKET COLON COLON IDX CLOSE_SQUARE_BRACKET |
OPEN_SQUARE_BRACKET COLON IDX COLON CLOSE_SQUARE_BRACKET | OPEN_SQUARE_BRACKET COLON IDX
COLON IDX CLOSE_SQUARE_BRACKET | OPEN_SQUARE_BRACKET IDX COLON COLON
CLOSE_SQUARE_BRACKET | OPEN_SQUARE_BRACKET IDX COLON COLON IDX CLOSE_SQUARE_BRACKET |
OPEN_SQUARE_BRACKET IDX COLON IDX COLON CLOSE_SQUARE_BRACKET | OPEN_SQUARE_BRACKET IDX
COLON IDX COLON IDX CLOSE_SQUARE_BRACKET

// STRING
STRING_TEXT -> string | AS | IMPORT | FOR | IF | RAISE | WITH | FROM | BREAK | PASS |
CONTINUE | ELIF | ELSE | IN | AND | OR | TRUE | FALSE | STRING_TEXT STRING_TEXT | DOT |
MINUS | PLUS | COMMA | MULTIPLY | DIVISION | COMMA | ART_OP | ASS_OP | REL_OP |
OPEN_NORMAL_BRACKET | OPEN_CURLY_BRACKET | OPEN_SQUARE_BRACKET | CLOSE_CURLY_BRACKET |
CLOSE_NORMAL_BRACKET | CLOSE_SQUARE_BRACKET | COLON | SEMICOLON | BACKSLASH
STRING_WITH_QUOTES -> DOUBLE_QUOTE STRING_TEXT DOUBLE_QUOTE | SINGLE_QUOTE STRING_TEXT
SINGLE_QUOTE
STRING -> STRING_WITH_QUOTES VAR_IDX | STRING_WITH_QUOTES MULTIPLY POSITIVE_NUMBER |
STRING_WITH_QUOTES PLUS STRING_WITH_QUOTES | STRING_WITH_QUOTES

// COMMENT
COMMENT -> DOUBLE_QUOTE DOUBLE_QUOTE DOUBLE_QUOTE STRING_TEXT DOUBLE_QUOTE DOUBLE_QUOTE
DOUBLE_QUOTE | SINGLE_QUOTE SINGLE_QUOTE SINGLE_QUOTE STRING_TEXT SINGLE_QUOTE
SINGLE_QUOTE SINGLE_QUOTE

// AND OR NOT
AND -> and
OR -> or
NOT -> not
IS -> is
```

```
// TRUE FALSE NONE
TRUE -> true
FALSE -> false
NONE -> none

// FUNCTION
FUNCTION -> FUNCTION_BASE | FUNCTION_BASE VAR_IDX
FUNCTION_BASE -> VARIABLE OPEN_NORMAL_BRACKET FUNCTION_PARAM CLOSE_NORMAL_BRACKET |
VARIABLE OPEN_NORMAL_BRACKET CLOSE_NORMAL_BRACKET
FUNCTION_PARAM -> FUNCTION_PARAM COMMA FUNCTION_PARAM | VARIABLE | CONSTANT | STRING |
LIST | DICT | NONE | FUNCTION | METH_TEXTED | ART_OPERATION | COND_OPERATION |
ASS_OPERATION

// DICT
DICT -> OPEN_CURLY_BRACKET DICT_TEXTED CLOSE_CURLY_BRACKET | OPEN_CURLY_BRACKET
CLOSE_CURLY_BRACKET
DICT_TEXTED -> DICT_TYPE COMMA DICT_TYPE | DICT_TYPE
DICT_TYPE -> DICT_TYPES COLON DICT_TYPES
DICT_TYPES -> VARIABLE | CONSTANT | STRING | FUNCTION | METH_TEXTED | NONE

// LIST
LIST -> OPEN_SQUARE_BRACKET LIST_TEXTED CLOSE_SQUARE_BRACKET | OPEN_SQUARE_BRACKET
CLOSE_SQUARE_BRACKET
LIST_TYPE -> VARIABLE | CONSTANT | STRING | FUNCTION | METH_TEXTED | LIST | DICT | NONE
LIST_TEXTED -> LIST_TYPE | LIST_TEXTED COMMA LIST_TYPE | LIST_TYPE FOR VARIABLE IN
FUNCTION

// ASSIGNMENT OPERATION
ASS_DESTRUCTURE -> VARIABLE COMMA VARIABLE | ASS_DESTRUCTURE COMMA VARIABLE
ASS_OPERATION -> VARIABLE ASS_OPERATOR ASS_OPERAND | ASS_DESTRUCTURE ASS_OPERATOR
ASS_OPERAND
ASS_OPERATOR -> ASS_OP
ASS_OPERAND -> OPEN_NORMAL_BRACKET ASS_OPERAND CLOSE_NORMAL_BRACKET | VARIABLE |
CONSTANT | COND_OPERATION | ART_OPERATION | METH_TEXTED | FUNCTION | LIST | STRING |
DICT | NONE

// ARITHMETIC OPERATION
ART_OPERATION -> OPEN_NORMAL_BRACKET ART_OPERATION CLOSE_NORMAL_BRACKET | ART_OPERATION
ART_OPERATOR ART_OPERATION | ART_OPERAND
ART_OPERATOR -> ART_OP | BIT_OP
ART_OPERAND -> VARIABLE | CONSTANT | METH_TEXTED | FUNCTION

// RAISE
RAISE -> raise
RAI_OP -> RAISE RAI_BODY
RAI_BODY -> OPEN_NORMAL_BRACKET RAI_BODY CLOSE_NORMAL_BRACKET | VARIABLE | CONSTANT |
COND_OPERATION | ART_OPERATION | METH_TEXTED | FUNCTION | LIST | STRING | DICT | NONE

// IMPORT
IMP_OPERATION -> FROM IMP_TEXTED IMPORT IMPT_MOD | IMPORT IMP_TEXTED | IMPORT AS_BLOCK
AS_BLOCK -> AS_BLOCK COMMA AS_BLOCK | IMP_TEXTED AS NAME
IMP_TEXTED -> IMP_TEXTED DOT IMP_TEXTED | NAME
IMPT_MOD -> NAME | ALL | AS_BLOCK | IMPT_MOD COMMA IMPT_MOD
ALL -> *
AS -> as
IMPORT -> import
FROM -> from

// METHOD
METH_TEXTED -> METH_INIT DOT METH_TEXTED | METH_BACK_FUNC | METH_BACK_NAME
METH_TEXTED_NAME -> METH_INIT DOT METH_TEXTED_NAME | METH_BACK_NAME
METH_TEXTED_FUNC -> METH_INIT DOT METH_TEXTED_FUNC | METH_BACK_FUNC
METH_INIT -> OPEN_NORMAL_BRACKET METH_INIT CLOSE_NORMAL_BRACKET | METH_BACK_FUNC |
METH_BACK_NAME
METH_BACK_FUNC -> FUNCTION | FUNCTION VAR_IDX
METH_BACK_NAME -> NAME | NAME VAR_IDX
```

3.2. CNF Form

Untuk melakukan konversi CFG menjadi dalam bentuk CNF, kita perlu melakukan beberapa tahap yaitu sebagai berikut:

- Epsilon Symbol Elimination
- Ungenerating Symbol Elimination
- Unit Production Symbol Elimination
- Useless Symbol Eliination

Untuk melakukan hal diatas, kami menjalankan perintah berikut pada program

```
python cfgutil.py  --mode translate -if formatted -of json
data/cfg.txt data/cfg.json

python cfgutil.py  --mode cnf -if json -of json data/cfg.json
data/cnf.json

python cfgutil.py  --mode translate -if json -of formatted
data/cnf.json data/cnf.txt
```

Hasil yang didapatkan dari perintah diatas adalah sebagai berikut:

G = (V, T, P, S)

Start Symbol (S) : Start

Rules :

```
START -> OPEN_NORMAL_BRACKET S_0 | TEXTED TEXTED | ITERATE S_1 | FOR_HEADER
TEXTED | ART_OPERATION S_2 | pass | name | ITERATE S_3 | OPEN_CURLY_BRACKET
CLOSE_CURLY_BRACKET | OPEN_NORMAL_BRACKET S_4 | FUNCTION_BASE VAR_IDX |
IF_HEADER TEXTED | OPEN_NORMAL_BRACKET S_5 | VARIABLE S_6 | IF_TEXTED
ELIF_TEXTED | IMPORT AS_BLOCK | false | COND_OPERAND S_7 | ITERATE BREAK | MINUS
NUMBER | WITH_HEADER TEXTED | METH_TEXTED_NAME VAR_IDX | STRING_WITH_QUOTES
VAR_IDX | NOT COND_OPERAND | VARIABLE S_8 | VARIABLE S_9 | STRING_WITH_QUOTES
S_10 | PLUS NUMBER | none | OPEN_NORMAL_BRACKET S_11 | OPEN_SQUARE_BRACKET S_12
| DOUBLE_QUOTE S_13 | IF_TEXTED ELSE_TEXTED | ASS_DESTRUCTURE S_14 | IF_HEADER
COMMENT | ITERATE CONTINUE | OPEN_NORMAL_BRACKET S_15 | METH_INIT S_16 |
DOUBLE_QUOTE S_17 | FROM S_18 | METH_INIT S_19 | number | OPEN_SQUARE_BRACKET
CLOSE_SQUARE_BRACKET | NAME VAR_IDX | SINGLE_QUOTE S_20 | OPEN_CURLY_BRACKET
S_21 | STRING_WITH_QUOTES S_22 | RAISE RAI_BODY | IMPORT IMP_TEXTED | true |
SINGLE_QUOTE S_23 | DEF_FUNC_HEADER DEF_FUNC_BODY | FUNCTION VAR_IDX |
WHILE_HEADER TEXTED | CLASS_HEADER CLASS_BODY
TEXTED -> OPEN_NORMAL_BRACKET S_24 | TEXTED TEXTED | ITERATE S_25 | FOR_HEADER
TEXTED | ART_OPERATION S_26 | pass | name | ITERATE S_27 | OPEN_CURLY_BRACKET
CLOSE_CURLY_BRACKET | OPEN_NORMAL_BRACKET S_28 | FUNCTION_BASE VAR_IDX |
IF_HEADER TEXTED | OPEN_NORMAL_BRACKET S_29 | VARIABLE S_30 | IF_TEXTED
ELIF_TEXTED | IMPORT AS_BLOCK | false | COND_OPERAND S_31 | ITERATE BREAK |
MINUS NUMBER | WITH_HEADER TEXTED | METH_TEXTED_NAME VAR_IDX |
STRING_WITH_QUOTES VAR_IDX | NOT COND_OPERAND | VARIABLE S_32 | VARIABLE S_33 |
STRING_WITH_QUOTES S_34 | PLUS NUMBER | none | OPEN_NORMAL_BRACKET S_35 |
OPEN_SQUARE_BRACKET S_36 | DOUBLE_QUOTE S_37 | IF_TEXTED ELSE_TEXTED |
ASS_DESTRUCTURE S_38 | IF_HEADER COMMENT | ITERATE CONTINUE |
OPEN_NORMAL_BRACKET S_39 | METH_INIT S_40 | DOUBLE_QUOTE S_41 | FROM S_42 |
METH_INIT S_43 | number | OPEN_SQUARE_BRACKET CLOSE_SQUARE_BRACKET | NAME
VAR_IDX | SINGLE_QUOTE S_44 | OPEN_CURLY_BRACKET S_45 | STRING_WITH_QUOTES S_46
| RAISE RAI_BODY | IMPORT IMP_TEXTED | true | SINGLE_QUOTE S_47 |
DEF_FUNC_HEADER DEF_FUNC_BODY | FUNCTION VAR_IDX | WHILE_HEADER TEXTED |
CLASS_HEADER CLASS_BODY
DOT -> .
PLUS -> +
MINUS -> -
MULTIPLY -> *
DIVISION -> /
OPEN_SQUARE_BRACKET -> [
CLOSE_SQUARE_BRACKET -> ]
OPEN_CURLY_BRACKET -> {
CLOSE_CURLY_BRACKET -> }
OPEN_NORMAL_BRACKET -> (
CLOSE_NORMAL_BRACKET -> )
SINGLE_QUOTE -> '
DOUBLE_QUOTE -> "
COLON -> :
COMMA -> ,
BANG_SYM -> !
GT_SYM -> <
LT_SYM -> >
EQ_SYM -> =
PERCENT_SYM -> %
IF -> if
ELIF -> elif
ELSE -> else
ELSE_HEADER -> ELSE COLON
ELSE_TEXTED -> ELSE_HEADER TEXTED
COND_OPERATION -> OPEN_NORMAL_BRACKET S_48 | ART_OPERATION S_49 | name |
OPEN_NORMAL_BRACKET S_50 | FUNCTION_BASE VAR_IDX | OPEN_NORMAL_BRACKET S_51 |
VARIABLE S_52 | false | COND_OPERAND S_53 | MINUS NUMBER | METH_TEXTED_NAME
VAR_IDX | STRING_WITH_QUOTES VAR_IDX | NOT COND_OPERAND | VARIABLE S_54 |
STRING_WITH_QUOTES S_55 | PLUS NUMBER | none | OPEN_NORMAL_BRACKET S_56 |
OPEN_SQUARE_BRACKET S_57 | DOUBLE_QUOTE S_58 | OPEN_NORMAL_BRACKET S_59 |
METH_INIT S_60 | METH_INIT S_61 | number | OPEN_SQUARE_BRACKET
CLOSE_SQUARE_BRACKET | NAME VAR_IDX | SINGLE_QUOTE S_62 | STRING_WITH_QUOTES
S_63 | true | FUNCTION VAR_IDX
COND_OPERATOR -> in | and | > | EQ_SYM EQ_SYM | or | < | is | GT_SYM EQ_SYM |
LT_SYM EQ_SYM | BANG_SYM EQ_SYM
COND_OPERAND -> OPEN_NORMAL_BRACKET S_64 | ART_OPERATION S_65 | name |
OPEN_NORMAL_BRACKET S_66 | FUNCTION_BASE VAR_IDX | OPEN_NORMAL_BRACKET S_67 |
VARIABLE S_68 | false | MINUS NUMBER | METH_TEXTED_NAME VAR_IDX |
STRING_WITH_QUOTES VAR_IDX | NOT COND_OPERAND | VARIABLE S_69 |
STRING_WITH_QUOTES S_70 | PLUS NUMBER | none | OPEN_SQUARE_BRACKET S_71 |
DOUBLE_QUOTE S_72 | OPEN_NORMAL_BRACKET S_73 | METH_INIT S_74 | METH_INIT S_75 |
number | OPEN_SQUARE_BRACKET CLOSE_SQUARE_BRACKET | NAME VAR_IDX | SINGLE_QUOTE
S_76 | STRING_WITH_QUOTES S_77 | true | FUNCTION VAR_IDX
FOR -> for
FOR_VARIABLE -> OPEN_NORMAL_BRACKET S_78 | NAME VAR_IDX | name |
METH_TEXTED_NAME VAR_IDX | VARIABLE S_79 | METH_INIT S_80
ITERABLE -> name | OPEN_CURLY_BRACKET CLOSE_CURLY_BRACKET | FUNCTION_BASE
VAR_IDX | OPEN_NORMAL_BRACKET S_81 | VARIABLE S_82 | METH_TEXTED_NAME VAR_IDX |
STRING_WITH_QUOTES VAR_IDX | VARIABLE S_83 | STRING_WITH_QUOTES S_84 |
OPEN_SQUARE_BRACKET S_85 | DOUBLE_QUOTE S_86 | METH_INIT S_87 | METH_INIT S_88 |
OPEN_SQUARE_BRACKET CLOSE_SQUARE_BRACKET | NAME VAR_IDX | SINGLE_QUOTE S_89 |
OPEN_CURLY_BRACKET S_90 | STRING_WITH_QUOTES S_91 | FUNCTION VAR_IDX
IN -> in
```

```
WHILE -> while
WHILE_HEADER -> WHILE S_92
CONTINUE -> continue
BREAK -> break
ITERATE -> FOR_HEADER TEXTED | FOR S_93 | WHILE_HEADER TEXTED | WHILE S_94
ITERATE_FUNC -> WHILE_HEADER TEXTED_FUNC | FOR S_95 | FOR_HEADER TEXTED_FUNC |
WHILE S_96
CLASS -> class
CLASS_PARAM -> name | CLASS_PARAM S_97
CLASS_BODY -> FOR_HEADER TEXTED | ITERATE S_98 | ITERATE S_99 | pass | name |
OPEN_CURLY_BRACKET CLOSE_CURLY_BRACKET | OPEN_NORMAL_BRACKET S_100 |
FUNCTION_BASE VAR_IDX | IF_HEADER TEXTED | OPEN_NORMAL_BRACKET S_101 | VARIABLE
S_102 | IF_TEXTED ELIF_TEXTED | IMPORT AS_BLOCK | false | ITERATE BREAK | MINUS
NUMBER | CLASS_BODY CLASS_BODY | METH_TEXTED_NAME VAR_IDX | STRING_WITH_QUOTES
VAR_IDX | VARIABLE S_103 | VARIABLE S_104 | STRING_WITH_QUOTES S_105 | PLUS
NUMBER | OPEN_SQUARE_BRACKET S_106 | IF_TEXTED ELSE_TEXTED | IF_HEADER COMMENT |
DOUBLE_QUOTE S_107 | ASS_DESTRUCTURE S_108 | ITERATE CONTINUE | METH_INIT S_109
| DOUBLE_QUOTE S_110 | FROM S_111 | METH_INIT S_112 | OPEN_SQUARE_BRACKET
CLOSE_SQUARE_BRACKET | number | NAME VAR_IDX | SINGLE_QUOTE S_113 |
OPEN_CURLY_BRACKET S_114 | STRING_WITH_QUOTES S_115 | IMPORT IMP_TEXTED | true |
SINGLE_QUOTE S_116 | DEF_FUNC_HEADER DEF_FUNC_BODY | FUNCTION VAR_IDX |
WHILE_HEADER TEXTED | CLASS_HEADER CLASS_BODY
NAME -> name
DEF -> def
DEF_FUNC_HEADER -> DEF S_117 | DEF S_118
DEF_FUNC_BODY -> CLASS_HEADER CLASS_BODY | pass | name | return |
OPEN_CURLY_BRACKET CLOSE_CURLY_BRACKET | OPEN_NORMAL_BRACKET S_119 |
FUNCTION_BASE VAR_IDX | OPEN_NORMAL_BRACKET S_120 | RETURN DEF_FUNC_RETURN_VAL |
TEXTED_FUNC TEXTED_FUNC | VARIABLE S_121 | IMPORT AS_BLOCK | false |
DEF_FUNC_BODY DEF_FUNC_BODY | MINUS NUMBER | METH_TEXTED_NAME VAR_IDX |
ITERATE_FUNC BREAK | STRING_WITH_QUOTES VAR_IDX | VARIABLE S_122 | VARIABLE
S_123 | STRING_WITH_QUOTES S_124 | PLUS NUMBER | none | WHILE_HEADER TEXTED_FUNC
| OPEN_SQUARE_BRACKET S_125 | DOUBLE_QUOTE S_126 | ITERATE_FUNC S_127 |
ASS_DESTRUCTURE S_128 | WITH_HEADER TEXTED_FUNC | METH_INIT S_129 | FOR_HEADER
TEXTED_FUNC | ITERATE S_130 | DOUBLE_QUOTE S_131 | FROM S_132 | ITERATE_FUNC
CONTINUE | RETURN ART_OPERATION | OPEN_SQUARE_BRACKET CLOSE_SQUARE_BRACKET |
number | IF_TEXTED_FUNC ELSE_TEXTED_FUNC | METH_INIT S_133 | NAME VAR_IDX |
SINGLE_QUOTE S_134 | OPEN_CURLY_BRACKET S_135 | IF_TEXTED_FUNC ELIF_TEXTED_FUNC
| STRING_WITH_QUOTES S_136 | RAISE RAI_BODY | IMPORT IMP_TEXTED | IF_HEADER
TEXTED_FUNC | true | SINGLE_QUOTE S_137 | DEF_FUNC_HEADER DEF_FUNC_BODY |
FUNCTION VAR_IDX | RETURN COND_OPERATION
DEF_FUNC_RETURN_VAL -> name | OPEN_CURLY_BRACKET CLOSE_CURLY_BRACKET |
OPEN_NORMAL_BRACKET S_138 | FUNCTION_BASE VAR_IDX | OPEN_NORMAL_BRACKET S_139 |
VARIABLE S_140 | false | MINUS NUMBER | METH_TEXTED_NAME VAR_IDX |
STRING_WITH_QUOTES VAR_IDX | VARIABLE S_141 | STRING_WITH_QUOTES S_142 | PLUS
NUMBER | none | OPEN_SQUARE_BRACKET S_143 | DOUBLE_QUOTE S_144 | METH_INIT S_145
| METH_INIT S_146 | number | OPEN_SQUARE_BRACKET CLOSE_SQUARE_BRACKET | NAME
VAR_IDX | SINGLE_QUOTE S_147 | OPEN_CURLY_BRACKET S_148 | STRING_WITH_QUOTES
S_149 | true | FUNCTION VAR_IDX
DEF_FUNC_PARAM -> name | DEF_FUNC_PARAM S_150
RETURN -> return
TEXTED_FUNC -> CLASS_HEADER CLASS_BODY | pass | name | return |
OPEN_CURLY_BRACKET CLOSE_CURLY_BRACKET | OPEN_NORMAL_BRACKET S_151 |
FUNCTION_BASE VAR_IDX | OPEN_NORMAL_BRACKET S_152 | RETURN DEF_FUNC_RETURN_VAL |
TEXTED_FUNC TEXTED_FUNC | VARIABLE S_153 | IMPORT AS_BLOCK | false | MINUS
NUMBER | METH_TEXTED_NAME VAR_IDX | ITERATE_FUNC BREAK | STRING_WITH_QUOTES
VAR_IDX | VARIABLE S_154 | VARIABLE S_155 | STRING_WITH_QUOTES S_156 | PLUS
NUMBER | none | WHILE_HEADER TEXTED_FUNC | OPEN_SQUARE_BRACKET S_157 |
DOUBLE_QUOTE S_158 | ITERATE_FUNC S_159 | ASS_DESTRUCTURE S_160 | WITH_HEADER
TEXTED_FUNC | METH_INIT S_161 | FOR_HEADER TEXTED_FUNC | ITERATE S_162 |
DOUBLE_QUOTE S_163 | FROM S_164 | ITERATE_FUNC CONTINUE | RETURN ART_OPERATION |
OPEN_SQUARE_BRACKET CLOSE_SQUARE_BRACKET | number | IF_TEXTED_FUNC
```

```
ELSE_TEXTED_FUNC | METH_INIT S_165 | NAME VAR_IDX | SINGLE_QUOTE S_166 |
OPEN_CURLY_BRACKET S_167 | IF_TEXTED_FUNC ELIF_TEXTED_FUNC | STRING_WITH_QUOTES
S_168 | RAISE RAI_BODY | IMPORT IMP_TEXTED | IF_HEADER TEXTED_FUNC | true |
SINGLE_QUOTE S_169 | DEF_FUNC_HEADER DEF_FUNC_BODY | FUNCTION VAR_IDX | RETURN
COND_OPERATION
WITH -> with
NUMBER -> MINUS NUMBER | PLUS NUMBER | number
POSITIVE_NUMBER -> PLUS POSITIVE_NUMBER | number
CONSTANT -> PLUS NUMBER | false | MINUS NUMBER | true | OPEN_NORMAL_BRACKET
S_170 | number
VARIABLE -> OPEN_NORMAL_BRACKET S_171 | NAME VAR_IDX | name | METH_TEXTED_NAME
VAR_IDX | METH_INIT S_172
IDX -> OPEN_NORMAL_BRACKET S_173 | VARIABLE S_174 | NAME VAR_IDX | PLUS NUMBER |
false | ART_OPERATION S_175 | MINUS NUMBER | name | true | METH_TEXTED_NAME
VAR_IDX | OPEN_NORMAL_BRACKET S_176 | METH_INIT S_177 | VARIABLE S_178 |
FUNCTION VAR_IDX | OPEN_NORMAL_BRACKET S_179 | FUNCTION_BASE VAR_IDX | METH_INIT
S_180 | number
VAR_IDX -> OPEN_SQUARE_BRACKET S_181 | OPEN_SQUARE_BRACKET S_182 |
OPEN_SQUARE_BRACKET S_183 | OPEN_SQUARE_BRACKET S_184 | OPEN_SQUARE_BRACKET
S_185 | OPEN_SQUARE_BRACKET S_186 | OPEN_SQUARE_BRACKET S_187 |
OPEN_SQUARE_BRACKET S_188 | OPEN_SQUARE_BRACKET S_189 | OPEN_SQUARE_BRACKET
S_190 | OPEN_SQUARE_BRACKET S_191 | VAR_IDX VAR_IDX | OPEN_SQUARE_BRACKET S_192
| OPEN_SQUARE_BRACKET S_193
STRING_TEXT -> MULTIPLY S_194 | } | PLUS EQ_SYM | raise | = | pass | . | ] |
STRING_TEXT STRING_TEXT | continue | and | false | [ | or | for | else | , | % |
DIVISION EQ_SYM | as | * | \ | elif | if | ; | in | break | import | MULTIPLY
MULTIPLY | EQ_SYM EQ_SYM | ( | < | from | / | string | with | MULTIPLY EQ_SYM |
DIVISION DIVISION | { | PERCENT_SYM EQ_SYM | > | ) | DIVISION S_195 | true | + |
: | - | LT_SYM EQ_SYM | GT_SYM EQ_SYM | BANG_SYM EQ_SYM | MINUS EQ_SYM | is
STRING_WITH_QUOTES -> DOUBLE_QUOTE S_196 | SINGLE_QUOTE S_197
COMMENT -> DOUBLE_QUOTE S_198 | SINGLE_QUOTE S_199
NOT -> not
IS -> is
FUNCTION -> VARIABLE S_200 | FUNCTION_BASE VAR_IDX | VARIABLE S_201
FUNCTION_BASE -> VARIABLE S_202 | VARIABLE S_203
FUNCTION_PARAM -> OPEN_NORMAL_BRACKET S_204 | ART_OPERATION S_205 | name |
OPEN_CURLY_BRACKET CLOSE_CURLY_BRACKET | OPEN_NORMAL_BRACKET S_206 |
FUNCTION_BASE VAR_IDX | OPEN_NORMAL_BRACKET S_207 | VARIABLE S_208 |
COND_OPERAND S_209 | false | FUNCTION_PARAM S_210 | MINUS NUMBER |
STRING_WITH_QUOTES VAR_IDX | METH_TEXTED_NAME VAR_IDX | NOT COND_OPERAND |
VARIABLE S_211 | VARIABLE S_212 | STRING_WITH_QUOTES S_213 | PLUS NUMBER | none
| OPEN_NORMAL_BRACKET S_214 | OPEN_SQUARE_BRACKET S_215 | DOUBLE_QUOTE S_216 |
ASS_DESTRUCTURE S_217 | OPEN_NORMAL_BRACKET S_218 | METH_INIT S_219 | METH_INIT
S_220 | number | OPEN_SQUARE_BRACKET CLOSE_SQUARE_BRACKET | NAME VAR_IDX |
SINGLE_QUOTE S_221 | OPEN_CURLY_BRACKET S_222 | STRING_WITH_QUOTES S_223 | true
| FUNCTION VAR_IDX
DICT_TYPES -> name | OPEN_NORMAL_BRACKET S_224 | FUNCTION_BASE VAR_IDX |
OPEN_NORMAL_BRACKET S_225 | VARIABLE S_226 | false | MINUS NUMBER |
STRING_WITH_QUOTES VAR_IDX | METH_TEXTED_NAME VAR_IDX | VARIABLE S_227 |
STRING_WITH_QUOTES S_228 | PLUS NUMBER | none | DOUBLE_QUOTE S_229 | METH_INIT
S_230 | METH_INIT S_231 | number | NAME VAR_IDX | SINGLE_QUOTE S_232 |
STRING_WITH_QUOTES S_233 | true | FUNCTION VAR_IDX
LIST_TYPE -> name | OPEN_CURLY_BRACKET CLOSE_CURLY_BRACKET | OPEN_NORMAL_BRACKET
S_234 | FUNCTION_BASE VAR_IDX | OPEN_NORMAL_BRACKET S_235 | VARIABLE S_236 |
false | MINUS NUMBER | STRING_WITH_QUOTES VAR_IDX | METH_TEXTED_NAME VAR_IDX |
VARIABLE S_237 | STRING_WITH_QUOTES S_238 | PLUS NUMBER | none |
OPEN_SQUARE_BRACKET S_239 | DOUBLE_QUOTE S_240 | METH_INIT S_241 | METH_INIT
S_242 | OPEN_SQUARE_BRACKET CLOSE_SQUARE_BRACKET | number | NAME VAR_IDX |
SINGLE_QUOTE S_243 | OPEN_CURLY_BRACKET S_244 | STRING_WITH_QUOTES S_245 | true
| FUNCTION VAR_IDX
LIST_TEXTED -> name | OPEN_CURLY_BRACKET CLOSE_CURLY_BRACKET |
OPEN_NORMAL_BRACKET S_246 | FUNCTION_BASE VAR_IDX | OPEN_NORMAL_BRACKET S_247 |
VARIABLE S_248 | false | LIST_TEXTED S_249 | MINUS NUMBER | STRING_WITH_QUOTES
VAR_IDX | METH_TEXTED_NAME VAR_IDX | VARIABLE S_250 | STRING_WITH_QUOTES S_251 |
PLUS NUMBER | none | LIST_TYPE S_252 | OPEN_SQUARE_BRACKET S_253 | DOUBLE_QUOTE
S_254 | METH_INIT S_255 | METH_INIT S_256 | OPEN_SQUARE_BRACKET
CLOSE_SQUARE_BRACKET | number | NAME VAR_IDX | SINGLE_QUOTE S_257 |
OPEN_CURLY_BRACKET S_258 | STRING_WITH_QUOTES S_259 | true | FUNCTION VAR_IDX
ASS_DESTRUCTURE -> VARIABLE S_260 | ASS_DESTRUCTURE S_261
ASS_OPERATOR -> PERCENT_SYM EQ_SYM | DIVISION EQ_SYM | MULTIPLY S_262 | PLUS
EQ_SYM | = | DIVISION S_263 | MINUS EQ_SYM | MULTIPLY EQ_SYM
ASS_OPERAND -> OPEN_NORMAL_BRACKET S_264 | ART_OPERATION S_265 | name |
OPEN_CURLY_BRACKET CLOSE_CURLY_BRACKET | OPEN_NORMAL_BRACKET S_266 |
FUNCTION_BASE VAR_IDX | OPEN_NORMAL_BRACKET S_267 | VARIABLE S_268 |
COND_OPERAND S_269 | false | MINUS NUMBER | STRING_WITH_QUOTES VAR_IDX |
METH_TEXTED_NAME VAR_IDX | NOT COND_OPERAND | VARIABLE S_270 |
STRING_WITH_QUOTES S_271 | PLUS NUMBER | none | OPEN_NORMAL_BRACKET S_272 |
OPEN_SQUARE_BRACKET S_273 | DOUBLE_QUOTE S_274 | OPEN_NORMAL_BRACKET S_275 |
METH_INIT S_276 | METH_INIT S_277 | number | OPEN_SQUARE_BRACKET
CLOSE_SQUARE_BRACKET | NAME VAR_IDX | SINGLE_QUOTE S_278 | OPEN_NORMAL_BRACKET
S_279 | OPEN_CURLY_BRACKET S_280 | STRING_WITH_QUOTES S_281 | true | FUNCTION
VAR_IDX
ART_OPERATION -> OPEN_NORMAL_BRACKET S_282 | VARIABLE S_283 | NAME VAR_IDX |
PLUS NUMBER | false | ART_OPERATION S_284 | MINUS NUMBER | name | true |
METH_TEXTED_NAME VAR_IDX | OPEN_NORMAL_BRACKET S_285 | METH_INIT S_286 |
VARIABLE S_287 | FUNCTION VAR_IDX | OPEN_NORMAL_BRACKET S_288 | FUNCTION_BASE
VAR_IDX | METH_INIT S_289 | number
ART_OPERATOR -> % | * | MULTIPLY MULTIPLY | GT_SYM GT_SYM | or | ~ | ^ | + | - |
/ | LT_SYM LT_SYM | & | DIVISION DIVISION
RAISE -> raise
RAI_BODY -> OPEN_NORMAL_BRACKET S_290 | ART_OPERATION S_291 | name |
OPEN_CURLY_BRACKET CLOSE_CURLY_BRACKET | OPEN_NORMAL_BRACKET S_292 |
FUNCTION_BASE VAR_IDX | OPEN_NORMAL_BRACKET S_293 | VARIABLE S_294 |
COND_OPERAND S_295 | false | MINUS NUMBER | STRING_WITH_QUOTES VAR_IDX |
METH_TEXTED_NAME VAR_IDX | NOT COND_OPERAND | VARIABLE S_296 |
OPEN_NORMAL_BRACKET S_297 | STRING_WITH_QUOTES S_298 | PLUS NUMBER | none |
OPEN_NORMAL_BRACKET S_299 | OPEN_SQUARE_BRACKET S_300 | DOUBLE_QUOTE S_301 |
OPEN_NORMAL_BRACKET S_302 | METH_INIT S_303 | METH_INIT S_304 | number |
OPEN_SQUARE_BRACKET CLOSE_SQUARE_BRACKET | NAME VAR_IDX | SINGLE_QUOTE S_305 |
OPEN_CURLY_BRACKET S_306 | STRING_WITH_QUOTES S_307 | true | FUNCTION VAR_IDX
IMP_TEXTED -> IMP_TEXTED S_308 | name
IMPT_MOD -> * | IMPT_MOD S_309 | name | AS_BLOCK S_310 | IMP_TEXTED S_311
AS -> as
IMPORT -> import
FROM -> from
METH_TEXTED -> VARIABLE S_312 | NAME VAR_IDX | name | METH_INIT S_313 | FUNCTION
VAR_IDX | FUNCTION_BASE VAR_IDX | VARIABLE S_314
METH_TEXTED_NAME -> name | METH_INIT S_315 | NAME VAR_IDX
METH_TEXTED_FUNC -> VARIABLE S_316 | METH_INIT S_317 | FUNCTION VAR_IDX |
FUNCTION_BASE VAR_IDX | VARIABLE S_318
METH_INIT -> VARIABLE S_319 | NAME VAR_IDX | name | OPEN_NORMAL_BRACKET S_320 |
FUNCTION VAR_IDX | FUNCTION_BASE VAR_IDX | VARIABLE S_321
IF_HEADER -> IF S_322
IF_TEXTED -> IF_HEADER COMMENT | IF_TEXTED ELSE_TEXTED | IF_TEXTED ELIF_TEXTED |
IF_HEADER TEXTED
ELIF_HEADER -> ELIF S_323
ELIF_TEXTED -> ELIF_TEXTED ELIF_TEXTED | ELIF_HEADER TEXTED | ELIF_TEXTED
ELSE_TEXTED
IF_TEXTED_FUNC -> IF_TEXTED_FUNC ELIF_TEXTED_FUNC | IF_TEXTED_FUNC
ELSE_TEXTED_FUNC | IF_HEADER TEXTED_FUNC
ELIF_TEXTED_FUNC -> ELIF_TEXTED_FUNC ELIF_TEXTED_FUNC | ELIF_HEADER TEXTED_FUNC
| ELIF_TEXTED_FUNC ELSE_TEXTED_FUNC
ELSE_TEXTED_FUNC -> ELSE_HEADER TEXTED_FUNC
FOR_HEADER -> FOR S_324
```

```
CLASS_HEADER -> CLASS S_325 | CLASS S_326
WITH_ST -> VARIABLE S_327 | METH_INIT S_328 | FUNCTION VAR_IDX | FUNCTION_BASE
VAR_IDX | VARIABLE S_329
DICT_TEXTED -> DICT_TYPE S_330 | DICT_TYPES S_331
DICT_TYPE -> DICT_TYPES S_332
AS_BLOCK -> IMP_TEXTED S_333 | AS_BLOCK S_334
WITH_HEADER -> WITH S_335
S_0 -> COND_OPERAND CLOSE_NORMAL_BRACKET
S_1 -> CONTINUE TEXTED
S_2 -> ART_OPERATOR ART_OPERATION
S_3 -> BREAK TEXTED
S_4 -> CONSTANT CLOSE_NORMAL_BRACKET
S_5 -> VARIABLE CLOSE_NORMAL_BRACKET
S_6 -> OPEN_NORMAL_BRACKET S_336
S_7 -> COND_OPERATOR COND_OPERATION
S_8 -> ASS_OPERATOR ASS_OPERAND
S_9 -> OPEN_NORMAL_BRACKET CLOSE_NORMAL_BRACKET
S_10 -> PLUS STRING_WITH_QUOTES
S_11 -> COND_OPERATION CLOSE_NORMAL_BRACKET
S_12 -> LIST_TEXTED CLOSE_SQUARE_BRACKET
S_13 -> STRING_TEXT DOUBLE_QUOTE
S_14 -> ASS_OPERATOR ASS_OPERAND
S_15 -> ART_OPERATION CLOSE_NORMAL_BRACKET
S_16 -> DOT METH_TEXTED
S_17 -> DOUBLE_QUOTE S_337
S_18 -> IMP_TEXTED S_338
S_19 -> DOT METH_TEXTED_NAME
S_20 -> STRING_TEXT SINGLE_QUOTE
S_21 -> DICT_TEXTED CLOSE_CURLY_BRACKET
S_22 -> MULTIPLY POSITIVE_NUMBER
S_23 -> SINGLE_QUOTE S_339
S_24 -> COND_OPERAND CLOSE_NORMAL_BRACKET
S_25 -> CONTINUE TEXTED
S_26 -> ART_OPERATOR ART_OPERATION
S_27 -> BREAK TEXTED
S_28 -> CONSTANT CLOSE_NORMAL_BRACKET
S_29 -> VARIABLE CLOSE_NORMAL_BRACKET
S_30 -> OPEN_NORMAL_BRACKET S_340
S_31 -> COND_OPERATOR COND_OPERATION
S_32 -> ASS_OPERATOR ASS_OPERAND
S_33 -> OPEN_NORMAL_BRACKET CLOSE_NORMAL_BRACKET
S_34 -> PLUS STRING_WITH_QUOTES
S_35 -> COND_OPERATION CLOSE_NORMAL_BRACKET
S_36 -> LIST_TEXTED CLOSE_SQUARE_BRACKET
S_37 -> STRING_TEXT DOUBLE_QUOTE
S_38 -> ASS_OPERATOR ASS_OPERAND
S_39 -> ART_OPERATION CLOSE_NORMAL_BRACKET
S_40 -> DOT METH_TEXTED
S_41 -> DOUBLE_QUOTE S_341
S_42 -> IMP_TEXTED S_342
S_43 -> DOT METH_TEXTED_NAME
S_44 -> STRING_TEXT SINGLE_QUOTE
S_45 -> DICT_TEXTED CLOSE_CURLY_BRACKET
S_46 -> MULTIPLY POSITIVE_NUMBER
S_47 -> SINGLE_QUOTE S_343
S_48 -> COND_OPERAND CLOSE_NORMAL_BRACKET
S_49 -> ART_OPERATOR ART_OPERATION
S_50 -> CONSTANT CLOSE_NORMAL_BRACKET
S_51 -> VARIABLE CLOSE_NORMAL_BRACKET
S_52 -> OPEN_NORMAL_BRACKET S_344
S_53 -> COND_OPERATOR COND_OPERATION
S_54 -> OPEN_NORMAL_BRACKET CLOSE_NORMAL_BRACKET
```

```
S_55 -> PLUS STRING_WITH_QUOTES
S_56 -> COND_OPERATION CLOSE_NORMAL_BRACKET
S_57 -> LIST_TEXTED CLOSE_SQUARE_BRACKET
S_58 -> STRING_TEXT DOUBLE_QUOTE
S_59 -> ART_OPERATION CLOSE_NORMAL_BRACKET
S_60 -> DOT METH_TEXTED
S_61 -> DOT METH_TEXTED_NAME
S_62 -> STRING_TEXT SINGLE_QUOTE
S_63 -> MULTIPLY POSITIVE_NUMBER
S_64 -> COND_OPERAND CLOSE_NORMAL_BRACKET
S_65 -> ART_OPERATOR ART_OPERATION
S_66 -> CONSTANT CLOSE_NORMAL_BRACKET
S_67 -> VARIABLE CLOSE_NORMAL_BRACKET
S_68 -> OPEN_NORMAL_BRACKET S_345
S_69 -> OPEN_NORMAL_BRACKET CLOSE_NORMAL_BRACKET
S_70 -> PLUS STRING_WITH_QUOTES
S_71 -> LIST_TEXTED CLOSE_SQUARE_BRACKET
S_72 -> STRING_TEXT DOUBLE_QUOTE
S_73 -> ART_OPERATION CLOSE_NORMAL_BRACKET
S_74 -> DOT METH_TEXTED
S_75 -> DOT METH_TEXTED_NAME
S_76 -> STRING_TEXT SINGLE_QUOTE
S_77 -> MULTIPLY POSITIVE_NUMBER
S_78 -> VARIABLE CLOSE_NORMAL_BRACKET
S_79 -> COMMA FOR_VARIABLE
S_80 -> DOT METH_TEXTED_NAME
S_81 -> VARIABLE CLOSE_NORMAL_BRACKET
S_82 -> OPEN_NORMAL_BRACKET S_346
S_83 -> OPEN_NORMAL_BRACKET CLOSE_NORMAL_BRACKET
S_84 -> PLUS STRING_WITH_QUOTES
S_85 -> LIST_TEXTED CLOSE_SQUARE_BRACKET
S_86 -> STRING_TEXT DOUBLE_QUOTE
S_87 -> DOT METH_TEXTED
S_88 -> DOT METH_TEXTED_NAME
S_89 -> STRING_TEXT SINGLE_QUOTE
S_90 -> DICT_TEXTED CLOSE_CURLY_BRACKET
S_91 -> MULTIPLY POSITIVE_NUMBER
S_92 -> COND_OPERATION COLON
S_93 -> FOR_VARIABLE S_347
S_94 -> COND_OPERATION COLON
S_95 -> FOR_VARIABLE S_348
S_96 -> COND_OPERATION COLON
S_97 -> COMMA CLASS_PARAM
S_98 -> CONTINUE TEXTED
S_99 -> BREAK TEXTED
S_100 -> CONSTANT CLOSE_NORMAL_BRACKET
S_101 -> VARIABLE CLOSE_NORMAL_BRACKET
S_102 -> OPEN_NORMAL_BRACKET S_349
S_103 -> ASS_OPERATOR ASS_OPERAND
S_104 -> OPEN_NORMAL_BRACKET CLOSE_NORMAL_BRACKET
S_105 -> PLUS STRING_WITH_QUOTES
S_106 -> LIST_TEXTED CLOSE_SQUARE_BRACKET
S_107 -> STRING_TEXT DOUBLE_QUOTE
S_108 -> ASS_OPERATOR ASS_OPERAND
S_109 -> DOT METH_TEXTED
S_110 -> DOUBLE_QUOTE S_350
S_111 -> IMP_TEXTED S_351
S_112 -> DOT METH_TEXTED_NAME
S_113 -> STRING_TEXT SINGLE_QUOTE
S_114 -> DICT_TEXTED CLOSE_CURLY_BRACKET
S_115 -> MULTIPLY POSITIVE_NUMBER
S_116 -> SINGLE_QUOTE S_352
```

```
S_117 -> NAME S_353
S_118 -> NAME S_354
S_119 -> CONSTANT CLOSE_NORMAL_BRACKET
S_120 -> VARIABLE CLOSE_NORMAL_BRACKET
S_121 -> OPEN_NORMAL_BRACKET S_355
S_122 -> ASS_OPERATOR ASS_OPERAND
S_123 -> OPEN_NORMAL_BRACKET CLOSE_NORMAL_BRACKET
S_124 -> PLUS STRING_WITH_QUOTES
S_125 -> LIST_TEXTED CLOSE_SQUARE_BRACKET
S_126 -> STRING_TEXT DOUBLE_QUOTE
S_127 -> BREAK TEXTED_FUNC
S_128 -> ASS_OPERATOR ASS_OPERAND
S_129 -> DOT METH_TEXTED
S_130 -> CONTINUE TEXTED_FUNC
S_131 -> DOUBLE_QUOTE S_356
S_132 -> IMP_TEXTED S_357
S_133 -> DOT METH_TEXTED_NAME
S_134 -> STRING_TEXT SINGLE_QUOTE
S_135 -> DICT_TEXTED CLOSE_CURLY_BRACKET
S_136 -> MULTIPLY POSITIVE_NUMBER
S_137 -> SINGLE_QUOTE S_358
S_138 -> CONSTANT CLOSE_NORMAL_BRACKET
S_139 -> VARIABLE CLOSE_NORMAL_BRACKET
S_140 -> OPEN_NORMAL_BRACKET S_359
S_141 -> OPEN_NORMAL_BRACKET CLOSE_NORMAL_BRACKET
S_142 -> PLUS STRING_WITH_QUOTES
S_143 -> LIST_TEXTED CLOSE_SQUARE_BRACKET
S_144 -> STRING_TEXT DOUBLE_QUOTE
S_145 -> DOT METH_TEXTED
S_146 -> DOT METH_TEXTED_NAME
S_147 -> STRING_TEXT SINGLE_QUOTE
S_148 -> DICT_TEXTED CLOSE_CURLY_BRACKET
S_149 -> MULTIPLY POSITIVE_NUMBER
S_150 -> COMMA DEF_FUNC_PARAM
S_151 -> CONSTANT CLOSE_NORMAL_BRACKET
S_152 -> VARIABLE CLOSE_NORMAL_BRACKET
S_153 -> OPEN_NORMAL_BRACKET S_360
S_154 -> ASS_OPERATOR ASS_OPERAND
S_155 -> OPEN_NORMAL_BRACKET CLOSE_NORMAL_BRACKET
S_156 -> PLUS STRING_WITH_QUOTES
S_157 -> LIST_TEXTED CLOSE_SQUARE_BRACKET
S_158 -> STRING_TEXT DOUBLE_QUOTE
S_159 -> BREAK TEXTED_FUNC
S_160 -> ASS_OPERATOR ASS_OPERAND
S_161 -> DOT METH_TEXTED
S_162 -> CONTINUE TEXTED_FUNC
S_163 -> DOUBLE_QUOTE S_361
S_164 -> IMP_TEXTED S_362
S_165 -> DOT METH_TEXTED_NAME
S_166 -> STRING_TEXT SINGLE_QUOTE
S_167 -> DICT_TEXTED CLOSE_CURLY_BRACKET
S_168 -> MULTIPLY POSITIVE_NUMBER
S_169 -> SINGLE_QUOTE S_363
S_170 -> CONSTANT CLOSE_NORMAL_BRACKET
S_171 -> VARIABLE CLOSE_NORMAL_BRACKET
S_172 -> DOT METH_TEXTED_NAME
S_173 -> VARIABLE CLOSE_NORMAL_BRACKET
S_174 -> OPEN_NORMAL_BRACKET S_364
S_175 -> ART_OPERATOR ART_OPERATION
S_176 -> ART_OPERATION CLOSE_NORMAL_BRACKET
S_177 -> DOT METH_TEXTED
S_178 -> OPEN_NORMAL_BRACKET CLOSE_NORMAL_BRACKET
```

```
S_179 -> CONSTANT CLOSE_NORMAL_BRACKET
S_180 -> DOT METH_TEXTED_NAME
S_181 -> IDX S_365
S_182 -> COLON S_366
S_183 -> IDX S_367
S_184 -> COLON S_368
S_185 -> IDX S_369
S_186 -> COLON S_370
S_187 -> COLON S_371
S_188 -> COLON S_372
S_189 -> IDX CLOSE_SQUARE_BRACKET
S_190 -> IDX S_373
S_191 -> COLON OPEN_SQUARE_BRACKET
S_192 -> IDX S_374
S_193 -> IDX S_375
S_194 -> MULTIPLY EQ_SYM
S_195 -> DIVISION EQ_SYM
S_196 -> STRING_TEXT DOUBLE_QUOTE
S_197 -> STRING_TEXT SINGLE_QUOTE
S_198 -> DOUBLE_QUOTE S_376
S_199 -> SINGLE_QUOTE S_377
S_200 -> OPEN_NORMAL_BRACKET S_378
S_201 -> OPEN_NORMAL_BRACKET CLOSE_NORMAL_BRACKET
S_202 -> OPEN_NORMAL_BRACKET S_379
S_203 -> OPEN_NORMAL_BRACKET CLOSE_NORMAL_BRACKET
S_204 -> COND_OPERAND CLOSE_NORMAL_BRACKET
S_205 -> ART_OPERATOR ART_OPERATION
S_206 -> CONSTANT CLOSE_NORMAL_BRACKET
S_207 -> VARIABLE CLOSE_NORMAL_BRACKET
S_208 -> OPEN_NORMAL_BRACKET S_380
S_209 -> COND_OPERATOR COND_OPERATION
S_210 -> COMMA FUNCTION_PARAM
S_211 -> ASS_OPERATOR ASS_OPERAND
S_212 -> OPEN_NORMAL_BRACKET CLOSE_NORMAL_BRACKET
S_213 -> PLUS STRING_WITH_QUOTES
S_214 -> COND_OPERATION CLOSE_NORMAL_BRACKET
S_215 -> LIST_TEXTED CLOSE_SQUARE_BRACKET
S_216 -> STRING_TEXT DOUBLE_QUOTE
S_217 -> ASS_OPERATOR ASS_OPERAND
S_218 -> ART_OPERATION CLOSE_NORMAL_BRACKET
S_219 -> DOT METH_TEXTED
S_220 -> DOT METH_TEXTED_NAME
S_221 -> STRING_TEXT SINGLE_QUOTE
S_222 -> DICT_TEXTED CLOSE_CURLY_BRACKET
S_223 -> MULTIPLY POSITIVE_NUMBER
S_224 -> CONSTANT CLOSE_NORMAL_BRACKET
S_225 -> VARIABLE CLOSE_NORMAL_BRACKET
S_226 -> OPEN_NORMAL_BRACKET S_381
S_227 -> OPEN_NORMAL_BRACKET CLOSE_NORMAL_BRACKET
S_228 -> PLUS STRING_WITH_QUOTES
S_229 -> STRING_TEXT DOUBLE_QUOTE
S_230 -> DOT METH_TEXTED
S_231 -> DOT METH_TEXTED_NAME
S_232 -> STRING_TEXT SINGLE_QUOTE
S_233 -> MULTIPLY POSITIVE_NUMBER
S_234 -> CONSTANT CLOSE_NORMAL_BRACKET
S_235 -> VARIABLE CLOSE_NORMAL_BRACKET
S_236 -> OPEN_NORMAL_BRACKET S_382
S_237 -> OPEN_NORMAL_BRACKET CLOSE_NORMAL_BRACKET
S_238 -> PLUS STRING_WITH_QUOTES
S_239 -> LIST_TEXTED CLOSE_SQUARE_BRACKET
S_240 -> STRING_TEXT DOUBLE_QUOTE
```

```
S_241 -> DOT METH_TEXTED
S_242 -> DOT METH_TEXTED_NAME
S_243 -> STRING_TEXT SINGLE_QUOTE
S_244 -> DICT_TEXTED CLOSE_CURLY_BRACKET
S_245 -> MULTIPLY POSITIVE_NUMBER
S_246 -> CONSTANT CLOSE_NORMAL_BRACKET
S_247 -> VARIABLE CLOSE_NORMAL_BRACKET
S_248 -> OPEN_NORMAL_BRACKET S_383
S_249 -> COMMA LIST_TYPE
S_250 -> OPEN_NORMAL_BRACKET CLOSE_NORMAL_BRACKET
S_251 -> PLUS STRING_WITH_QUOTES
S_252 -> FOR S_384
S_253 -> LIST_TEXTED CLOSE_SQUARE_BRACKET
S_254 -> STRING_TEXT DOUBLE_QUOTE
S_255 -> DOT METH_TEXTED
S_256 -> DOT METH_TEXTED_NAME
S_257 -> STRING_TEXT SINGLE_QUOTE
S_258 -> DICT_TEXTED CLOSE_CURLY_BRACKET
S_259 -> MULTIPLY POSITIVE_NUMBER
S_260 -> COMMA VARIABLE
S_261 -> COMMA VARIABLE
S_262 -> MULTIPLY EQ_SYM
S_263 -> DIVISION EQ_SYM
S_264 -> COND_OPERAND CLOSE_NORMAL_BRACKET
S_265 -> ART_OPERATOR ART_OPERATION
S_266 -> CONSTANT CLOSE_NORMAL_BRACKET
S_267 -> VARIABLE CLOSE_NORMAL_BRACKET
S_268 -> OPEN_NORMAL_BRACKET S_385
S_269 -> COND_OPERATOR COND_OPERATION
S_270 -> OPEN_NORMAL_BRACKET CLOSE_NORMAL_BRACKET
S_271 -> PLUS STRING_WITH_QUOTES
S_272 -> COND_OPERATION CLOSE_NORMAL_BRACKET
S_273 -> LIST_TEXTED CLOSE_SQUARE_BRACKET
S_274 -> STRING_TEXT DOUBLE_QUOTE
S_275 -> ART_OPERATION CLOSE_NORMAL_BRACKET
S_276 -> DOT METH_TEXTED
S_277 -> DOT METH_TEXTED_NAME
S_278 -> STRING_TEXT SINGLE_QUOTE
S_279 -> ASS_OPERAND CLOSE_NORMAL_BRACKET
S_280 -> DICT_TEXTED CLOSE_CURLY_BRACKET
S_281 -> MULTIPLY POSITIVE_NUMBER
S_282 -> VARIABLE CLOSE_NORMAL_BRACKET
S_283 -> OPEN_NORMAL_BRACKET S_386
S_284 -> ART_OPERATOR ART_OPERATION
S_285 -> ART_OPERATION CLOSE_NORMAL_BRACKET
S_286 -> DOT METH_TEXTED
S_287 -> OPEN_NORMAL_BRACKET CLOSE_NORMAL_BRACKET
S_288 -> CONSTANT CLOSE_NORMAL_BRACKET
S_289 -> DOT METH_TEXTED_NAME
S_290 -> COND_OPERAND CLOSE_NORMAL_BRACKET
S_291 -> ART_OPERATOR ART_OPERATION
S_292 -> CONSTANT CLOSE_NORMAL_BRACKET
S_293 -> VARIABLE CLOSE_NORMAL_BRACKET
S_294 -> OPEN_NORMAL_BRACKET S_387
S_295 -> COND_OPERATOR COND_OPERATION
S_296 -> OPEN_NORMAL_BRACKET CLOSE_NORMAL_BRACKET
S_297 -> RAI_BODY CLOSE_NORMAL_BRACKET
S_298 -> PLUS STRING_WITH_QUOTES
S_299 -> COND_OPERATION CLOSE_NORMAL_BRACKET
S_300 -> LIST_TEXTED CLOSE_SQUARE_BRACKET
S_301 -> STRING_TEXT DOUBLE_QUOTE
S_302 -> ART_OPERATION CLOSE_NORMAL_BRACKET
```

```
S_303 -> DOT METH_TEXTED
S_304 -> DOT METH_TEXTED_NAME
S_305 -> STRING_TEXT SINGLE_QUOTE
S_306 -> DICT_TEXTED CLOSE_CURLY_BRACKET
S_307 -> MULTIPLY POSITIVE_NUMBER
S_308 -> DOT IMP_TEXTED
S_309 -> COMMA IMPT_MOD
S_310 -> COMMA AS_BLOCK
S_311 -> AS NAME
S_312 -> OPEN_NORMAL_BRACKET S_388
S_313 -> DOT METH_TEXTED
S_314 -> OPEN_NORMAL_BRACKET CLOSE_NORMAL_BRACKET
S_315 -> DOT METH_TEXTED_NAME
S_316 -> OPEN_NORMAL_BRACKET S_389
S_317 -> DOT METH_TEXTED_FUNC
S_318 -> OPEN_NORMAL_BRACKET CLOSE_NORMAL_BRACKET
S_319 -> OPEN_NORMAL_BRACKET S_390
S_320 -> METH_INIT CLOSE_NORMAL_BRACKET
S_321 -> OPEN_NORMAL_BRACKET CLOSE_NORMAL_BRACKET
S_322 -> COND_OPERATION COLON
S_323 -> COND_OPERATION COLON
S_324 -> FOR_VARIABLE S_391
S_325 -> NAME COLON
S_326 -> NAME S_392
S_327 -> OPEN_NORMAL_BRACKET S_393
S_328 -> DOT METH_TEXTED_FUNC
S_329 -> OPEN_NORMAL_BRACKET CLOSE_NORMAL_BRACKET
S_330 -> COMMA DICT_TYPE
S_331 -> COLON DICT_TYPES
S_332 -> COLON DICT_TYPES
S_333 -> AS NAME
S_334 -> COMMA AS_BLOCK
S_335 -> WITH_ST S_394
S_336 -> FUNCTION_PARAM CLOSE_NORMAL_BRACKET
S_337 -> DOUBLE_QUOTE S_395
S_338 -> IMPORT IMPT_MOD
S_339 -> SINGLE_QUOTE S_396
S_340 -> FUNCTION_PARAM CLOSE_NORMAL_BRACKET
S_341 -> DOUBLE_QUOTE S_397
S_342 -> IMPORT IMPT_MOD
S_343 -> SINGLE_QUOTE S_398
S_344 -> FUNCTION_PARAM CLOSE_NORMAL_BRACKET
S_345 -> FUNCTION_PARAM CLOSE_NORMAL_BRACKET
S_346 -> FUNCTION_PARAM CLOSE_NORMAL_BRACKET
S_347 -> IN S_399
S_348 -> IN S_400
S_349 -> FUNCTION_PARAM CLOSE_NORMAL_BRACKET
S_350 -> DOUBLE_QUOTE S_401
S_351 -> IMPORT IMPT_MOD
S_352 -> SINGLE_QUOTE S_402
S_353 -> OPEN_NORMAL_BRACKET S_403
S_354 -> OPEN_NORMAL_BRACKET S_404
S_355 -> FUNCTION_PARAM CLOSE_NORMAL_BRACKET
S_356 -> DOUBLE_QUOTE S_405
S_357 -> IMPORT IMPT_MOD
S_358 -> SINGLE_QUOTE S_406
S_359 -> FUNCTION_PARAM CLOSE_NORMAL_BRACKET
S_360 -> FUNCTION_PARAM CLOSE_NORMAL_BRACKET
S_361 -> DOUBLE_QUOTE S_407
S_362 -> IMPORT IMPT_MOD
S_363 -> SINGLE_QUOTE S_408
S_364 -> FUNCTION_PARAM CLOSE_NORMAL_BRACKET
```

```
S_365 -> COLON S_409
S_366 -> IDX CLOSE_SQUARE_BRACKET
S_367 -> COLON S_410
S_368 -> COLON CLOSE_SQUARE_BRACKET
S_369 -> COLON S_411
S_370 -> IDX S_412
S_371 -> IDX S_413
S_372 -> COLON S_414
S_373 -> COLON S_415
S_374 -> COLON S_416
S_375 -> COLON CLOSE_SQUARE_BRACKET
S_376 -> DOUBLE_QUOTE S_417
S_377 -> SINGLE_QUOTE S_418
S_378 -> FUNCTION_PARAM CLOSE_NORMAL_BRACKET
S_379 -> FUNCTION_PARAM CLOSE_NORMAL_BRACKET
S_380 -> FUNCTION_PARAM CLOSE_NORMAL_BRACKET
S_381 -> FUNCTION_PARAM CLOSE_NORMAL_BRACKET
S_382 -> FUNCTION_PARAM CLOSE_NORMAL_BRACKET
S_383 -> FUNCTION_PARAM CLOSE_NORMAL_BRACKET
S_384 -> VARIABLE S_419
S_385 -> FUNCTION_PARAM CLOSE_NORMAL_BRACKET
S_386 -> FUNCTION_PARAM CLOSE_NORMAL_BRACKET
S_387 -> FUNCTION_PARAM CLOSE_NORMAL_BRACKET
S_388 -> FUNCTION_PARAM CLOSE_NORMAL_BRACKET
S_389 -> FUNCTION_PARAM CLOSE_NORMAL_BRACKET
S_390 -> FUNCTION_PARAM CLOSE_NORMAL_BRACKET
S_391 -> IN S_420
S_392 -> OPEN_NORMAL_BRACKET S_421
S_393 -> FUNCTION_PARAM CLOSE_NORMAL_BRACKET
S_394 -> AS S_422
S_395 -> STRING_TEXT S_423
S_396 -> STRING_TEXT S_424
S_397 -> STRING_TEXT S_425
S_398 -> STRING_TEXT S_426
S_399 -> ITERABLE COLON
S_400 -> ITERABLE COLON
S_401 -> STRING_TEXT S_427
S_402 -> STRING_TEXT S_428
S_403 -> DEF_FUNC_PARAM S_429
S_404 -> CLOSE_NORMAL_BRACKET COLON
S_405 -> STRING_TEXT S_430
S_406 -> STRING_TEXT S_431
S_407 -> STRING_TEXT S_432
S_408 -> STRING_TEXT S_433
S_409 -> IDX S_434
S_410 -> IDX S_435
S_411 -> COLON CLOSE_SQUARE_BRACKET
S_412 -> COLON S_436
S_413 -> COLON CLOSE_SQUARE_BRACKET
S_414 -> IDX CLOSE_SQUARE_BRACKET
S_415 -> IDX CLOSE_SQUARE_BRACKET
S_416 -> COLON S_437
S_417 -> STRING_TEXT S_438
S_418 -> STRING_TEXT S_439
S_419 -> IN FUNCTION
S_420 -> ITERABLE COLON
S_421 -> CLASS_PARAM S_440
S_422 -> VARIABLE COLON
S_423 -> DOUBLE_QUOTE S_441
S_424 -> SINGLE_QUOTE S_442
S_425 -> DOUBLE_QUOTE S_443
S_426 -> SINGLE_QUOTE S_444
```

```
S_427 -> DOUBLE_QUOTE S_445
S_428 -> SINGLE_QUOTE S_446
S_429 -> CLOSE_NORMAL_BRACKET COLON
S_430 -> DOUBLE_QUOTE S_447
S_431 -> SINGLE_QUOTE S_448
S_432 -> DOUBLE_QUOTE S_449
S_433 -> SINGLE_QUOTE S_450
S_434 -> COLON CLOSE_SQUARE_BRACKET
S_435 -> COLON S_451
S_436 -> IDX CLOSE_SQUARE_BRACKET
S_437 -> IDX CLOSE_SQUARE_BRACKET
S_438 -> DOUBLE_QUOTE S_452
S_439 -> SINGLE_QUOTE S_453
S_440 -> CLOSE_NORMAL_BRACKET COLON
S_441 -> DOUBLE_QUOTE DOUBLE_QUOTE
S_442 -> SINGLE_QUOTE SINGLE_QUOTE
S_443 -> DOUBLE_QUOTE DOUBLE_QUOTE
S_444 -> SINGLE_QUOTE SINGLE_QUOTE
S_445 -> DOUBLE_QUOTE DOUBLE_QUOTE
S_446 -> SINGLE_QUOTE SINGLE_QUOTE
S_447 -> DOUBLE_QUOTE DOUBLE_QUOTE
S_448 -> SINGLE_QUOTE SINGLE_QUOTE
S_449 -> DOUBLE_QUOTE DOUBLE_QUOTE
S_450 -> SINGLE_QUOTE SINGLE_QUOTE
S_451 -> IDX CLOSE_SQUARE_BRACKET
S_452 -> DOUBLE_QUOTE DOUBLE_QUOTE
S_453 -> SINGLE_QUOTE SINGLE_QUOTE
```

BAB IV
Spesifikasi Teknis Program

4.1. Alur Umum Program

Program terdiri dari 2 file utama yaitu file cfgutil.py dan file linter.py. Kedua file tersebut merupakan starting point dari program ini. File cfgutil.py digunakan untuk melakukan konversi grammar yang akan dijadikan konfigurasi file. File linter.py digunakan untuk melakukan proses pemeriksaan grammar dari source code python.

File cfgutil.py terdiri dari 2 utilitas utama, yaitu penerjemahan dan pengkonversian. Pada file ini, Rule cfg dapat diubah dari bentuk terformat menjadi dalam bentuk json ataupun yaml. Berikut ini adalah perintah bantuan dari file ini:

```
usage: cfgutil.py [-h] [--mode {cnf,translate}]
                  [--input-format {formatted,json,yaml}]
                  [--output-format {formatted,json,yaml}]
                  input output

CFG Utility

positional arguments:
  input                 Input file path
  output                Output file path

optional arguments:
  -h, --help            show this help message and exit
  --mode {cnf,translate}, -m {cnf,translate}
                        CFG Util mode
  --input-format {formatted,json,yaml}, -if {formatted,json,yaml}
                        Input format file
  --output-format {formatted,json,yaml}, -of {formatted,json,yaml}
                        Input format file
```

Mode merupakan parameter yang digunakan untuk mengatur fitur yang akan digunakan pada perintah tersebut. Mode terbagi menjadi beberapa bagian, diantaranya sebagai berikut:

- Mode translate : Digunakan untuk mengubah file yang berisi informasi CFG diubah dari satu bentuk ke bentuk file lain. Format dari input dan output dapat diatur dengan menggunakan flag --input-format dan --output-format. Input format dan output format haruslah berbeda.
- Mode cnf : Digunakan untuk mengubah file CFG menjadi dalam bentuk CNF. Format input dan output yang diterima hanyalah dalam bentuk yaml dan json.

File linter.py digunakan untuk memeriksa *syntax* dari sebuah file python. Berikut ini adalah informasi pada file linter.py

```
usage: linter.py [-h] [-c CONFIG] path

Python script grammar checker

positional arguments:
  path                  python script path

optional arguments:
  -h, --help            show this help message and exit
  -c CONFIG, --config CONFIG
                        CNF Config file (JSON format)
```

Pada file ini, terdapat dua argumen yang wajib diisi. Parameter config digunakan untuk memberikan path dari file CNF hasil konversi. Untuk saat ini, hanya didukung file config yang menggunakan format json. Argumen path digunakan untuk memberikan informasi file yang akan diujikan. Lokasi file relative terhadap file linter.py ini disimpan.

Saat file ini dijalankan, akan didapatkan hasil diantaranya "*No syntax error detected*" dan "*Syntax error detected*".

4.2. Module util

Modul ini berisi utilitas yang dapat digunakan pada pemrosesan yang dilakukan pada program. Pada module ini, terdapat sebuah kelas boolmatrix, berada di submodul boolmatrix, yang dapat digunakan untuk menghitung klosur menghantar dari sebuah matriks boolean.

4.3. Module lib

Module lib berisi seluruh program yang berisi implementasi dari CNF, CYK, dan Finite Automata. Pada module ini terdapat 6 module, diantaranya sebagai berikut

Nama Modul	**Deskripsi**
`lib.cnf`	Modul ini berisi implementasi konverter dari CFG menjadi dalam bentuk CNF. Terdapat dua modul di dalam modul cnf, yaitu ● `lib.cnf.cnf` Implementasi konverter CFG menjadi CNF ● `lib.cnf.reductor` Mereduksi CFG menjadi CNF-like CFG. Pada tahap ini, CNF tidak dilakukan penyederhanaan.
`lib.converter`	Modul ini berisi implementasi pengubah bentuk formatted file

	menjadi file json ataupun yaml. Hanya terdapat sebuah modul di dalamnya, yaitu `lib.converter.cfg`. Modul cfg digunakan untuk Melakukan loading file cfg.txt sesuai format dan mengubahnya ke file json atau yaml.
`lib.cyk`	Modul ini berisi implementasi dari algoritma CYK yang digunakan untuk memeriksa syntax dari source code.
`lib.elimination`	Modul ini berisi implementasi pengeliminasian CFG.
`lib.fa`	Modul ini berisi implementasi finite automata yang dapat dibentuk dari Regular Expression.
`lib.cfg`	Modul ini berisi implementasi CFG yang dapat di load dari file JSON ataupun Yaml

BAB V
Capture Kasus Uji

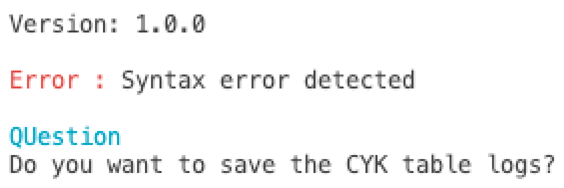
Dalam bagian ini, kami telah melakukan beberapa kasus uji terhadap compiler bahasa python yang telah kami buat. Berikut ini adalah beberapa kasus uji yang telah kami buat.

5.1. Penamaan variable

Kode input :

```
123a = "hallo dunia"
```

Hasil compile :



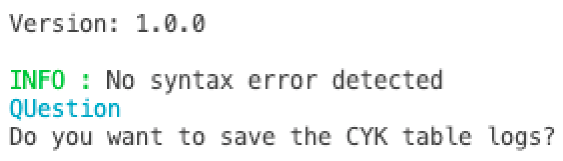
```
Version: 1.0.0

Error : Syntax error detected

QUestion
Do you want to save the CYK table logs?
```

5.2. Assignment Operation

Kode input :

```
X = 2
X += 3
X -= 3
X /= 3
X *= 3
```

Hasil compile :



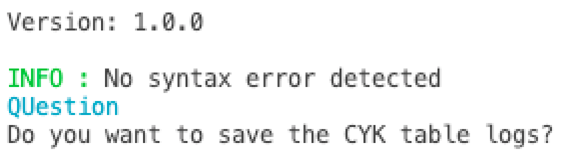
```
Version: 1.0.0

INFO : No syntax error detected
QUestion
Do you want to save the CYK table logs?
```

5.3. Fungsi

Kode input :

```
def do_something(x):
 if x == 0:
   return 0
 elif x + 4 == 1:
   if True:
       return 3
   else:
     ''' This is a sample multiline comment '''
     return 2
 elif x == 32:
   return 4
 else:
   return "Doodoo"
```

Hasil compile :



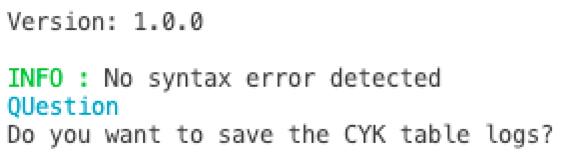
```
Version: 1.0.0

INFO : No syntax error detected
QUestion
Do you want to save the CYK table logs?
```

5.4. Kelas

Kode input :

```
class CFG :
   def __init__ (self) :
       self.x = 1
       return self.x
```

Hasil compile :



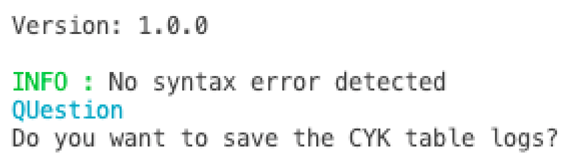
```
Version: 1.0.0

INFO : No syntax error detected
QUestion
Do you want to save the CYK table logs?
```

5.5. For loop dan ‘is’

Kode input :

```
x = 3
for i in range(x) :
   print( x is not i)
```

Hasil compile :



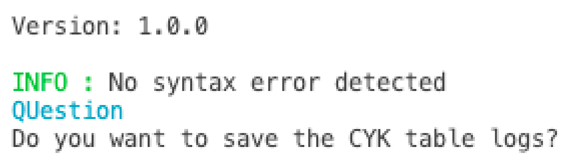
```
Version: 1.0.0

INFO : No syntax error detected
QUestion
Do you want to save the CYK table logs?
```

5.6. Komentar dan ‘raise’

Kode input :

```
x = -1


if x < 0:
 """Komentar"""
 raise Exception("Error")
```

Hasil compile :



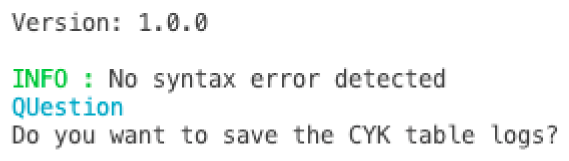
```
Version: 1.0.0

INFO : No syntax error detected
QUestion
Do you want to save the CYK table logs?
```

5.7. Method, ‘with’, dan ‘as’

Kode input :

```
with open('file_path', 'w') as file:
   file.write('hello world !')
```

Hasil compile :



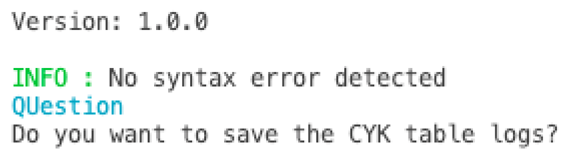
```
Version: 1.0.0

INFO : No syntax error detected
QUestion
Do you want to save the CYK table logs?
```

5.8. Input dan Output sederhana

Kode input :

```
a = input()
b = input()
c = a * b + 3
print(c)
print("tbfo")
```

Hasil compile :



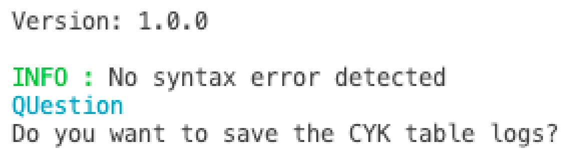
```
Version: 1.0.0

INFO : No syntax error detected
QUestion
Do you want to save the CYK table logs?
```

5.9. Percabangan

Kode input :

```
x = "a"

if x == 0:
 X = 0
elif x + 4 == 1:
 if True:
     x = 3
 else:
   x = 2
elif x == 32:
 x = 4
else:
 ''' This is a sample multiline comment '''
 x = "Doodoo"
```

Hasil compile :



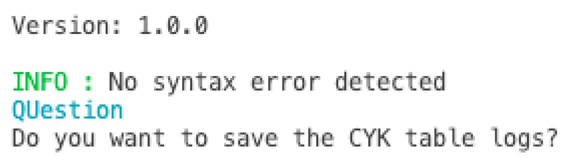
```
Version: 1.0.0

INFO : No syntax error detected
QUestion
Do you want to save the CYK table logs?
```

5.10 Import

Kode input :

```
import argparse
import numpy as np

parser = argparse.ArgumentParser(description="Python")
```

Hasil compile :



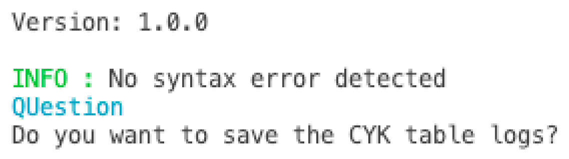
```
Version: 1.0.0

INFO : No syntax error detected
QUestion
Do you want to save the CYK table logs?
```

5.11 Penamaan variable

Kode input :

```
for i in range(x) :
   if i == 0 :
       break
   elif i == 1:
       continue
   else :
       pass
```

Hasil compile :



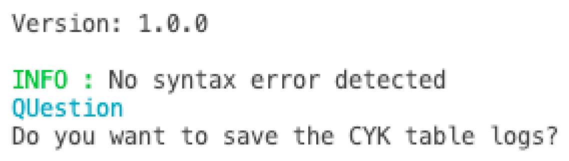
```
Version: 1.0.0

INFO : No syntax error detected
QUestion
Do you want to save the CYK table logs?
```

5.12. While loop, ‘and’, dan ‘None’

Kode input :

```
x = [2,3,4]
i = 2

while True :
   print("1")
   print("2")
   print("3")
   break
 while not False and i > 2:
   if x[1] == None:
     continue
   elif x[1] == 2:
     break
   else:
     pass
```

Hasil compile :



```
Version: 1.0.0

INFO : No syntax error detected
QUestion
Do you want to save the CYK table logs?
```

BAB VI
Kesimpulan

1. Kesimpulan

Tugas besar IF2124 Teori Bahasa Formal & Otomata, Python Sintax Checker yang memeriksa kebenaran sintax pada bahasa python telah berhasil diselesaikan dan diimplementasikan dengan baik sesuai dengan spesifikasi yang ada dan sesuai juga dengan materi-materi yang telah dipelajari di kelas. Hal-hal yang diimplementasikan pada program ini adalah sebagai berikut.

a. CFG

Digunakan sebagai rules dalam mengevaluasi sintax pada program.

b. CFG to CNF

Digunakan untuk melakukan convert dari CFG ke CNF

c. FA

FA yaitu finite automata digunakan untuk mengevaluasi nama variabel

d. CYK

CYK digunakan sebagai algoritma dasar dalam pengerjaan tugas besar ini

Semua materi di atas diimplementasikan dalam tugas besar ini dan membuat tugas besar ini dapat terimplementasikan dengan baik dan sesuai dengan yang diharapkan.

2. Saran

Berikut adalah saran yang dapat penulis berikan dalam tugas besar kali ini:

1. Perbanyak lakukan eksplorasi terhadap tugas yang diberikan.
2. Perbanyak menggali-gali referensi untuk membuat rules grammar yang benar
3. Mempunyai jiwa pembelajar yang kuat dan tangguh dalam mempelajari hal-hal baru.
4. Perbanyak membuat driver untuk melakukan berbagai test case
5. Sering melakukan debug terhadap program agar jika terdapat suatu hal yang error dapat segera diatasi
6. Perbanyak latihan dan jangan lupa istirahat dalam pengerjaan tugas

3. Refleksi

Belajar membuat compiler terhadap suatu bahasa pemograman merupakan suatu hal yang menarik dan menantang bagi penulis. Setelah mengerjakan tugas besar ini, timbul rasa apresiasi dari penulis kepada semua pihak yang terlibat dalam pengerjaan tugas besar ini. Mulai dari pemberian materi hingga pemberian spesifikasi tugas yang diberikan. Selain itu, penulis juga menyadari bahwa dengan mengerjakan tugas besar ini, mungkin saja di kemudian hari penulis tertantang untuk bisa membuat bahasa pemograman sendiri. Hal yang paling penting lagi dalam pengerjaan tugas besar ini adalah kami dapat memperbaiki kinerja kami dalam berkelompok mulai dari pembagian tugas hingga timeline pengerjaan sehingga semuanya dapat selesai sesuai dengan yang diharapkan.

BAB VII
Pembagian Kerja

No.	Nama / NIM	Tugas
1.	Bayu Samudra/13520128	Membuat converter cgf ke cnf
2.	Febryola Kurnia Putri/13520140	Membuat rules cgf
3.	Aloysius Gilang Pramudya / 13520147	Membuat cyk

DAFTAR REFERENSI

https://courses.cs.washington.edu/courses/cse322/08au/lec14.pdf, diakses tanggal 20 November 2021

https://docplayer.info/42226897-Aplikasi-program-dinamis-dalam-algoritma-cocke-younger-kasami-cyk.html, diakses tanggal 19 November 2021

https://data-flair.training/blogs/python-operator/,diakses tanggal 18 November 2021

https://blog.usejournal.com/writing-your-own-programming-language-and-compiler-with-python-a468970ae6df, diakses tanggal 20 November 2021

https://www.programiz.com/python-programming/keyword-list, diakses tanggal 20 November 2021