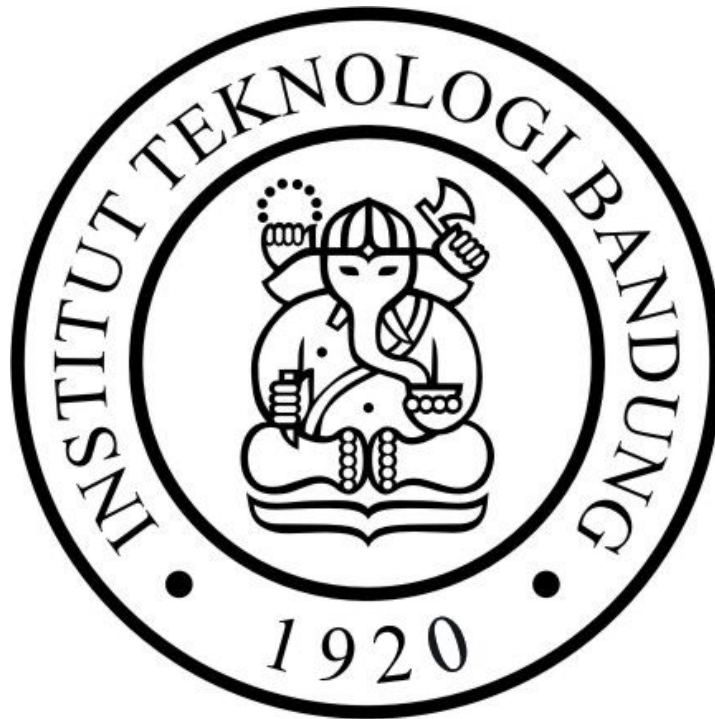


LAPORAN TUGAS BESAR II
IF-2124 : TEORI BAHASA FORMAL DAN OTOMATA
COMPILER BAHASA PYTHON



Disusun oleh :

Jun Ho Choi Hedyatmo	13518044
Naufal Prima Yoriko	13518146
Stefanus Gusega Gunawan	13518149

DAFTAR ISI

DAFTAR ISI	1
BAB I	2
Bahasa Python	2
Context-Free Grammar (CFG)	2
Chomsky Normal Form (CNF)	3
Algoritma Cocke-Younger-Kasami (CYK)	3
BAB II	5
Deskripsi Persoalan	5
Pemecahan Persoalan	5
BAB III	16
Spesifikasi Teknis Program	16
1.1 Grammar	16
1.2 Parser	17
1.3 CYK	17
1.4 CYK	18
Capture Layar	18
BAB IV	22

BAB I

TEORI DASAR

I. Bahasa Python

Python merupakan salah satu bahasa pemrograman yang terbilang sangat populer saat ini. Banyak aplikasi yang dikembangkan menggunakan bahasa pemrograman ini. Hal itu dikarenakan sintaks yang digunakan pada Python mudah dimengerti manusia dan memiliki *library-library* yang fiturnya amat banyak. Dengan sintaksis yang mudah, maka Python menjadi bahasa yang mudah dipelajari oleh pemula.

Python merupakan bahasa pemrograman tingkat tinggi yang diracik oleh Guido van Rossum. Python banyak digunakan untuk membuat berbagai macam program, seperti program CLI, program GUI (desktop), aplikasi *mobile*, pranala, IoT, *game*, program untuk *hacking*, dan sebagainya. Sampai sekarang, Python sudah mencapai versi 3.0+, yang mana memiliki fitur yang beragam dan mudah digunakan oleh umat manusia.

II. *Context-Free Grammar* (CFG)

Dalam mendefinisikan sebuah susunan bahasa, diperlukan sebuah aturan yang membuat struktur bahasa itu dapat dikenali. Hal ini dapat dilakukan dengan membuat sebuah *grammar*. Metode untuk mendefinisikan *grammar* juga sangat beragam, di antaranya adalah CFG, CNF, 2NF, 2LF, dan lain-lain. Biasanya, metode-metode pendefinisian *grammar* digunakan pada pendefinisian *grammar* bahasa pemrograman.

Context-Free Grammar adalah tata bahasa/*grammar* formal yang digunakan untuk memproduksi dan menggeneralisasi semua *string* yang mungkin dari suatu bahasa yang diberikan. Sebuah *context-free grammar* G dapat didefinisikan menjadi *4-tuple*, sebagai berikut.

$$G = (V, T, P, S)$$

G mendefinisikan *grammar*. T mendefinisikan sebuah himpunan terbatas dari simbol-simbol terminal. V mendefinisikan sebuah himpunan terbatas dari simbol-simbol

non-terminal. P mendefinisikan sebuah himpunan dari fungsi produksi yang berguna untuk menurunkan pembuktian *grammar*. S mendefinisikan *start symbol*.

CFG perlu disederhanakan dengan tujuan untuk melakukan pembatasan sehingga tidak menghasilkan pohon penurunan yang memiliki kerumitan yang tak perlu atau aturan produksi tak berarti. Metode penyederhanaan itu disebut dengan *Chomsky Normal Form* (CNF).

III. *Chomsky Normal Form* (CNF)

Bentuk Normal Chomsky atau *Chomsky Normal Form* (CNF) merupakan salah satu bentuk *grammar* yang sangat mudah di-*parsing* dan salah satu bentuk normal yang sangat berguna untuk *Context-Free Grammar* (CFG). CNF dapat dibentuk dari langkah-langkah penyederhanaan, yaitu penghilangan produksi *useless*, unit, dan epsilon (*string* kosong). Aturan *rule* dalam bentuk CNF adalah ruas kanan dari tiap *rule* hanya terdiri atas dua variabel atau satu terminal.

Dengan ekspresi yang lebih sederhana, tiap ekspresi pada CNF hanyalah berbentuk :

$$A \rightarrow BC \text{ atau } A \rightarrow a$$

Dengan A, B, dan C adalah simbol non-terminal, sedangkan a adalah simbol terminal.

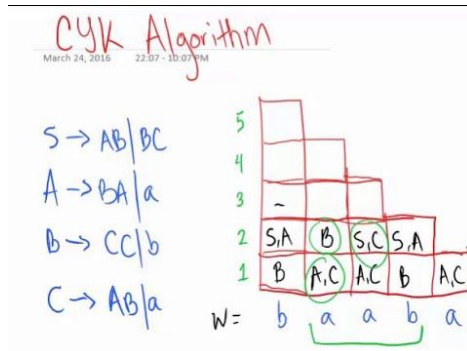
IV. Algoritma Cocke-Younger-Kasami (CYK)

Untuk mengenali sebuah pola dari *input string* yang dimasukkan oleh pengguna, dibutuhkan algoritma yang mampu mengecek apakah *input string* tersebut dapat didefinisikan oleh bahasa tersebut. Ada banyak metode untuk mengecek kebenaran suatu *input* tersebut. Salah satunya adalah dengan menggunakan algoritma Cocke-Younger-Kasami atau biasa dikenal dengan algoritma CYK.

Algoritma CYK adalah algoritma untuk menentukan apakah suatu untaian *string* dapat diterima oleh suatu tata bahasa bebas-konteks atau yang biasa disebut CFG yang sudah dinormalkan dan disederhanakan dalam bentuk CNF. Lalu, pengecekan dilakukan mulai dari *string* sampai apakah *string* ini dapat diturunkan dari CFG dan CNF. Oleh

karena itu, langkah pertama yang harus dilakukan adalah membuat CFG dari bahasa tersebut dan disederhanakan menjadi bentuk CNF.

Berikut ini adalah contoh pengisian tabel pada CYK :



Gambar 1 : Contoh pengisian tabel CYK

Sumber : <https://images.app.goo.gl/rhWwHcX4BRvDUGQv7>

BAB II

ANALISIS PERSOALAN

I. Deskripsi Persoalan

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 multi-paradigma 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*.

II. Pemecahan Persoalan

Pada persoalan ini, guna melakukan compile pada bahasa python, digunakan salah satu algoritma yang cukup mudah dan sederhana untuk diimplementasikan, yakni CYK. CYK sendiri secara umum memiliki performa dalam melakukan compiling yang cukup baik, yakni kompleksitasnya sekitar $O(n^3)$.

Dalam menerapkan CYK sendiri perlu dilakukan beberapa tahapan supaya kode yang telah dibuat dapat di-*compile*. Tahapan utama tersebut antara lain adalah *Parsing*, *Converting* CFG ke CNF, kemudian baru proses CYK itu sendiri guna menguji apakah suatu kode dapat di-*compile* atau tidak.

Tahap pertama, yaitu menentukan CFG dari bahasa Python. Kami diberikan tugas untuk mengimplementasikan beberapa *command* pada Python. Maka dari itu, kami perlu mencari bagaimanakah *command* tersebut diimplementasikan pada bahasa Python. Karena, yang dibuat pertama kali adalah CFG, maka penulisan *rule* masih banyak dan masih kurang rapi. Oleh karena itu, dibutuhkannya *converter* dari CFG ke CNF.

Akan tetapi, sebelum dikonversi ke CNF, terlebih dahulu dilakukan *parsing*. *Parsing* adalah suatu proses untuk memecah-mecah suatu *input string* menjadi beberapa bagian, yang kemudian akan diproses oleh CYK. Untuk *parsing*, kami menggunakan bahasa Python, karena kami menganggap bahasa Python adalah bahasa yang memiliki tingkat yang tinggi dan bisa melakukan proses-proses *parsing* dengan cepat dan memiliki kompleksitas program yang relatif kecil.

Parsing yang kami gunakan adalah membuat *input-input string* pada kode Python menjadi sebuah *array of tokens*. Hal itu akan lebih memudahkan pemrosesan algoritma CYK dan dengan cara tersebut akan menjadi lebih mudah dalam mencari kesalahan di *line* mana. Lalu, beralih ke konversi CFG ke CNF, yang mana kelompok kami tidak menggunakan *script* dalam bahasa Python, namun mengubahnya secara manual. Lalu, untuk CYK, kami menggunakan *script* juga dalam bahasa Python yang memanfaatkan konsep *vector* (dalam C++), yang mana merupakan konsep *map and dictionary* pada Python.

Berikut ini adalah *grammar* CFG dari kode Python yang sudah kami buat :

```
STARTSYMBOL -> XPRINT | XFOR | XIF | XIMP | ASSIGNMENT | XCLASS |  
XDEF | XELIF | XELSE | XRET | XWHILE | XWITH | BREAK | CONT | PASS |  
RAISE;  
ANGKA -> num;  
FALSE -> false;
```

NONE -> none;
TRUE -> true;
AND -> and;
AS -> as;
BREAK -> break;
CLASS -> class;
CONT -> continue;
DEF -> def;
ELIF -> elif;
ELSE -> else;
FOR -> for;
FROM -> from;
IF -> if;
IMP -> import;
IN -> in;
IS -> is;
NOT -> not;
OR -> or;
PASS -> pass;
RAISE -> raise;
RET -> return;
WHILE -> while;
WITH -> with;
RANGE -> range;
PRINT -> print;
RELASI -> op;
OPM -> opm;
BUKASIKU -> [
TUTUPIKU ->];

KURUNGBUKA -> (
 KURUNGTUTUP ->)
 KURAWALBUKA -> {
 KURAWALTUTUP -> }
 TITIKDUA -> ::
 VAR -> var;
 KOMA -> ,;
 PETIKSATU -> '
 PETIKDUA -> "
 ASSIGN -> assign;
 STR -> str;
 XARG -> VAR | STR | ANGKA | XARG KOMA VAR | XARG KOMA STR | XARG
 KOMA ANGKA | FUNC
 XARGD -> VAR | XARG KOMA VAR;
 XPRINT -> PRINT | PRINT XARG;
 XFOR -> FOR VAR IN RANGE KURUNGBUKA XARG KURUNGTUTUP
 TITIKDUA | FOR VAR IN VAR TITIKDUA | FOR VAR NOT IN RANGE
 KURUNGBUKA XARG KURUNGTUTUP TITIKDUA | FOR VAR NOT IN VAR
 TITIKDUA;
 XIF -> IF REL TITIKDUA;
 XIMP -> IMP VAR | IMP VAR AS VAR | FROM VAR IMP VAR;
 FUNC -> VAR KURUNGBUKA KURUNGTUTUP | VAR KURUNGBUKA XARG
 KURUNGTUTUP;
 XTRUEFALSE -> TRUE OR FALSE | TRUE AND FALSE | TRUE OR TRUE | TRUE
 AND TRUE | FALSE AND FALSE | FALSE OR FALSE | FALSE AND TRUE | FALSE
 OR TRUE;
 REL -> VAREXPR IN VAR | VAREXPR IN STR | VAREXPR IN FUNC | ANGKA
 RELASI ANGKA | VAR RELASI VAR | TRUE | FALSE | VAR RELASI ANGKA |

REL OR REL | REL AND REL | NOT REL | XTRUEFALSE | ANGKA | STR | VAR |
XIS | KURUNGBUKA REL KURUNGTUTUP;
MEXPR -> ANGKA | ANGKA OPM ANGKA | MEXPR OPM MEXPR |
KURUNGBUKA MEXPR KURUNGTUTUP;
SEXPR -> STR | SEXPR * ANGKA | SEXPR + SEXPR;
ARRSATU -> BUKASIKU MEXPR FOR VAR IN RANGE KURUNGBUKA XARG
KURUNGTUTUP TUTUPSIKU | BUKASIKU SEXPR FOR VAR IN RANGE
KURUNGBUKA XARG KURUNGTUTUP TUTUPSIKU | BUKASIKU ARRSATU
FOR VAR IN RANGE KURUNGBUKA XARG KURUNGTUTUP TUTUPSIKU;
ARRDUA -> MEXPR | SEXPR | ARRDUA KOMA ARRDUA;
ARREXPR -> BUKASIKU TUTUPSIKU | ARRSATU | BUKASIKU ARRDUA
TUTUPSIKU;
ASSIGNMENT -> VAR ASSIGN MEXPR | VAR KOMA ASSIGNMENT KOMA
MEXPR | VAR ASSIGN VAR | VAR KOMA ASSIGNMENT KOMA VAR | VAR
ASSIGN SEXPR | VAR KOMA ASSIGNMENT KOMA SEXPR | VAR ASSIGN
FUNC | VAR KOMA ASSIGNMENT KOMA FUNC | VAR ASSIGN ARREXPR |
VAR KOMA ASSIGNMENT KOMA ARREXPR;
XCLASS -> CLASS VAR KURUNGBUKA XARGD KURUNGTUTUP TITIKDUA;
XDEF -> DEF VAR KURUNGBUKA XARGD KURUNGTUTUP TITIKDUA;
XELIF -> ELIF REL TITIKDUA;
XELSE -> ELSE TITIKDUA;
XIS -> TRUE IS TRUE | FALSE IS TRUE | TRUE IS FALSE | FALSE IS FALSE |
ANGKA IS ANGKA | KURUNGBUKA KURUNGTUTUP IS KURUNGBUKA
KURUNGTUTUP | BUKASIKU TUTUPSIKU IS BUKASIKU TUTUPSIKU |
KURAWALBUKA KURAWALTUTUP IS KURAWALBUKA KURAWALTUTUP |
STR IS STR | VAR IS VAR | KURUNGBUKA XIS KURUNGTUTUP;
XRET -> RET MEXPR | RET VAR | RET SEXPR;
XWHILE -> WHILE REL TITIKDUA;
XWITH -> WITH FUNC AS VAR TITIKDUA;

VAREXPR -> var | VAR OPM VAREXPR | VAR OPM MEXPR | KURUNGBUKA
VAREXPR KURUNGTUTUP

Lalu, berikut CNF yang telah diproses oleh *script* dengan sedikit perubahan oleh kami :

ANGKA -> num

FALSE -> false

NONE -> none

TRUE -> true

AND -> and

AS -> as

BREAK -> break

CLASS -> class

CONT -> continue

DEF -> def

ELIF -> elif

ELSE -> else

FOR -> for

FROM -> from

IF -> if

IMP -> import

IN -> in

IS -> is

NOT -> not

OR -> or

PASS -> pass

RAISE -> raise

RET -> return

WHILE -> while

WITH -> with

RANGE -> range
 PRINT -> print
 RELASI -> op
 OPM -> opm
 BUKASIKU -> [
 TUTUPSIKU ->]
 KURUNGBUKA -> (
 KURUNGTUTUP ->)
 KURAWALBUKA -> {
 KURAWALTUTUP -> }
 TITIKDUA -> :
 VAR -> var | VAR I0
 KOMA -> ,
 PETIKSATU -> '
 PETIKDUA -> "
 ASSIGN -> assign
 STR -> str
 XARG -> XARG A1 | XARG B1 | XARG C1 | XARG AZ | var | str | num | VAR K1 |
 VAR L1
 A1 -> KOMA VAR
 B1 -> KOMA STR
 C1 -> KOMA ANGKA
 AZ -> KOMA FUNC
 XARGD -> XARG D1 | var
 D1 -> KOMA VAR
 XPRINT -> PRINT XARG | print
 TG -> KURUNGBUKA KURUNGTUTUP
 TF -> KURUNGBUKA RT
 RT -> XARG KURUNGTUTUP

XFOR -> FOR E1 | FOR F1
E1 -> VAR E2 | VAR E7
E2 -> IN E3
E3 -> RANGE E4
E4 -> KURUNGBUKA E5
E5 -> XARG E6
E6 -> KURUNGTUTUP TITIKDUA
E7 -> NOT E2
F1 -> VAR F2 | VAR F4
F2 -> IN F3
F3 -> VAR TITIKDUA
F4 -> NOT F2
XIF -> IF G1
G1 -> REL TITIKDUA
XELIF -> ELIF H1
XELSE -> ELSE TITIKDUA
XIMP -> IMP VAR | IMP I1 | FROM J1
I1 -> VAR I2
I2 -> AS VAR
J1 -> VAR J2
J2 -> IMP VAR
FUNC -> VAR K1 | VAR L1
K1 -> KURUNGBUKA KURUNGTUTUP
L1 -> KURUNGBUKA L2
L2 -> XARG KURUNGTUTUP
AM -> KURUNGBUKA AN
AN -> SEXPR KURUNGTUTUP
AQ -> VAR AO
AO -> KURUNGBUKA AP

AP -> KURUNGTUTUP TITIKDUA
 XDEF -> DEF M1
 M1 -> VAR M2
 M2 -> KURUNGBUKA M3
 M3 -> XARGD M4
 M4 -> KURUNGTUTUP TITIKDUA
 XRET -> RET MEXPR | RET VAR | RET SEXPR
 XWHILE -> WHILE N1
 N1 -> REL TITIKDUA
 XCLASS -> CLASS O1
 O1 -> VAR O2
 O2 -> KURUNGBUKA O3
 O3 -> XARGD O4
 O4 -> KURUNGTUTUP TITIKDUA
 XWITH -> WITH P1
 P1 -> FUNC P2
 P2 -> AS P3
 P3 -> VAR TITIKDUA
 MEXPR -> ANGKA Q1 | MEXPR R1 | KURUNGBUKA S1 | num
 Q1 -> OPM ANGKA
 R1 -> OPM MEXPR
 S1 -> MEXPR KURUNGTUTUP
 Z -> *
 SEXPR -> SEXPR T1 | SEXPR U1 | str
 T1 -> Z ANGKA
 Y -> opm
 VAREXPR -> var | VAR AG | VAR AI | KURUNGBUKA AH
 AG -> OPM VAREXPR
 AH -> VAREXPR KURUNGTUTUP

AI -> OPM MEXPR

U1 -> Y SEXPR

W1 -> ASSIGN MEXPR

X1 -> ASSIGN SEXPR

AJ -> ASSIGN VAREXPR

AK -> ASSIGN AL

AL -> BUKASIKU TUTUPSIKU

AR -> ASSIGN FUNC

REL -> true | false | num | str | var | NOT REL | NOT AD | MEXPR AA | MEXPR AB |
MEXPR AC | SEXPR AB | SEXPR AA | SEXPR AC | VAREXPR AC | VAREXPR AB |
VAREXPR AA | VAR BZ | ANGKA BZ | STR BZ | BA BZ | BE BZ | BH BZ | FUNC
BZ | VAR BY | ANGKA BY | STR BY | BA BY | BE BY | BH BY | FUNC BY | REL
AZ | REL AY | FUNC FY | ANGKA FY | BA FY | BE FY | BH FY | STR FY
FY -> RELASI ANGKA | RELASI STR | RELASI BA | RELASI BH | RELASI BE |
RELASI FUNC | FY FY

AZ -> OR REL

AY -> AND REL

AA -> RELASI MEXPR

AB -> RELASI SEXPR

AC -> RELASI VAREXPR

AD -> KURUNGBUKA AE

AE -> REL AF | REL KURUNGTUTUP

AF -> KURUNGTUTUP TITIKDUA

BC -> ASSIGN BA

BA -> BUKASIKU BB

BB -> XARG TUTUPSIKU

BD -> ASSIGN BE

BE -> KURAWALBUKA BF

BF -> XARG KURAWALTUTUP

BG -> ASSIGN BH

BH -> KURUNGBUKA BI

BI -> XARG KURUNGTUTUP

BY -> NOT BZ

BZ -> IN VAR | IN STR | IN ANGKA | IN BA | IN BE | IN BH | IN FUNC

I0 -> BUKASIKU I2 | I0 I0

I2 -> ANGKA TUTUPSIKU | VAR TUTUPSIKU | STR TUTUPSIKU

STARTSYMBOL -> PRINT VAREXPR | PRINT TF | PRINT AM | PRINT TG | PRINT
XARG | FOR E1 | FOR F1 | IF AD | IF G1 | IMP VAR | IMP I1 | FROM J1 | VAR W1 |
VAR X1 | VAR AK | VAR VAREXPR | VAR AJ | VAR AR | VAR BC | VAR BD |
VAR BG | CLASS O1 | DEF M1 | DEF AQ | VAR K1 | VAR L1 | ELIF G1 | ELIF AD |
ELSE TITIKDUA | RET MEXPR | RET VAR | RET SEXPR | WHILE N1 | WITH P1 |
break | continue | pass | raise | print

Untuk CFG dan CNF yang telah kami buat, mungkin ada terjadi ketidakcocokan, dikarenakan kami mengkonversi sendiri dari CFG ke CNF yang mana bisa menimbulkan ketidaktelitian. Dan, dalam proses *debugging* CNF, kami juga tidak menuliskan beberapa *rule* kembali ke dalam bentuk CFG lagi. Sehingga, mungkin saja jika ingin mencocokkan CFG dengan CNF, terjadi ketidakcocokan. Namun, yang kami gunakan untuk pembuktian dengan CYK adalah dengan menggunakan *file* cnf.txt.

BAB III

IMPLEMENTASI DAN PENGUJIAN

I. Spesifikasi Teknis Program

Untuk penjelasan mengenai teknis program, pertama-tama kami hendak menjelaskan struktur utama apa yang terdapat dalam program kami. Secara umum, program kami memiliki bagian *Parser*, CYK, dan juga *grammar*. Dalam merealisasikan fungsi-fungsi compiler python tersebut, kami menggunakan bahasa python untuk membuat program ini. Untuk lebih jelasnya akan diuraikan pada subbagian ini.

1.1 Grammar

Grammar ini bukan bagian dari program python, tapi merupakan database yang memuat aturan dalam meng-*construct* bahasa python yang diterima. *Grammar* di sini ditulis dalam format CFG (*context free grammar*), yang nanti akan diolah menjadi bentuk CNF oleh *converter* untuk kemudian digunakan secara nyata/riil oleh program untuk memvalidasi kode melalui CYK. Berikut adalah *screenshot* dari *file grammar.txt*

```
1 STARTSYMBOL -> XPRINT | XFOR | XIF | XIMP | ASSIGNMENT | XCLASS | XDEF | XELIF | XELSE | XRET | XWHILE | XWITH | BREAK | CO
2 ANGKA -> num;
3 FALSE -> false;
4 NONE -> none;
5 TRUE -> true;
6 AND -> and;
7 AS -> as;
8 BREAK -> break;
9 CLASS -> class;
10 CONT -> continue;
11 DEF -> def;
12 ELIF -> elif;
13 ELSE -> else;
14 FOR -> for;
15 FROM -> from;
16 IF -> if;
17 IMP -> import;
18 IN -> in;
19 IS -> is;
20 NOT -> not;
21 OR -> or;
22 PASS -> pass;
23 RAISE -> raise;
24 RET -> return;
25 WHILE -> while;
26 WITH -> with;
27 RANGE -> range;
28 PRINT -> print;
29 RELASI -> op;
30 OPM -> opm;
31 BUKASIKU -> [;
32 TUTUPIKU -> ];
33 KUTUBUNDA -> ;
```

Gambar : Potongan layar dari *grammar.txt*

1.2 Parser

Di bagian *parser* ini, program kami akan membaca *file*, yakni *file grammar* dan kode. *Parser* berfungsi untuk mengubah info-info yang terdapat pada kedua *file* tersebut menjadi bentuk yang dapat dimengerti program dan dapat diolah lebih lanjut.

No	Fungsi/Prosedur	Header fungsi/prosedur
1	<pre>def lexer(text_code, token_list): return tokenized</pre>	Sebuah fungsi untuk tokenisasi suatu kode pada <i>text_code</i> . Mengembalikan hasil-hasil tokenisasi dari kode, sesuai dengan <i>regex</i> yang telah didefinisikan sebelumnya pada program.
2	<pre>def parseGrammar(filename): return rules</pre>	Fungsi yang digunakan untuk <i>parsing file grammar</i> ke dalam bentuk <i>dictionary of list</i> .

1.3 CYK

Untuk CYK, berikut adalah tabel fungsi dan/atau prosedur yang kami gunakan.

No	Fungsi/Prosedur	Header fungsi/prosedur
1	<pre>def readCNF(filepath): return grammarLeft, grammarRight</pre>	Fungsi yang digunakan untuk membaca <i>file</i> yang berisi <i>grammar</i> CNF.
2	<pre>def isTerminal(a): return (a[1] == "" and not(a[0].isupper()))</pre>	Fungsi yang digunakan untuk memvalidasi apakah <i>a</i> adalah sebuah terminal.
3	<pre>def CYK(ln, grammarLeft, grammarRight, dp, inp): return dp</pre>	Fungsi yang digunakan untuk mengembalikan tabel CYK.

4	<pre>def isValid(dp, ln): return (not (dp[ln][1] == [])) and ("STARTSYMBOL" in dp[ln][1])</pre>	Fungsi yang digunakan untuk mengecek apakah di <i>top of table</i> atau <i>top of tree</i> terdiri atas STARTSYMBOL
5	<pre>def printTree(dp, ln):</pre>	I.S. : dp adalah tabel CYK dan ln adalah panjang <i>array of tokens</i> , semuanya terdefinisi F.S. :Tabel CYK tercetak

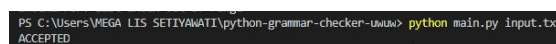
1.4 CYK

Untuk antarmuka dari program yang kami buat, kami membuatnya sesuai spesifikasi yaitu berbasis *Command Line*, yang mana kami menjalankan program kami pada *Command Prompt*.

Teknis program utamanya adalah kami menghubungkan program dengan *file* cnf.txt terlebih dahulu guna menambah *dictionary of grammar*. Lalu, program akan meminta *file* apakah yang akan diuji. *File* yang diuji haruslah memiliki ekstensi .txt. Dan juga ingat, kami hanya membuat *compiler* ini untuk mengecek bahasa Python saja. Setelah itu, program akan menampilkan apakah *file* tersebut lulus *compile* atau tidak. Jika lulus, maka program akan mencetak “ACCEPTED” ke layar, jika tidak, maka program akan mencetak "Syntax Error on line : " + str(Er) . Sehingga, si pengguna bisa tahu kesalahan penulisan kode Python-nya terdapat pada *line* seberapa.

II. Capture Layar

Untuk bagian *capture* layar kami akan memberikan beberapa contoh tampilan dari program yang sudah kami buat. Pertama, adalah cara menggunakan *compiler* ini dengan cara jadikan satu *file* program dengan *file* input. Misal di sini, *file* programnya adalah main.py, dan *file* input “input.txt”



```
PS C:\Users\MEGA LIS SETIYAWATI\python-grammar-checker-uuu> python main.py input.txt
ACCEPTED
```

Gambar 3 : Menguji *input.txt*

Sumber : Dokumen pribadi

Dan juga, berikut ini adalah hasil *capture* layar dari 5 *file input.txt* yang sudah kami ujikan:

1. *Testcase pertama*

```
from s.tr import x

s = 5

def func(x, y):
    for x in y:
        if(x>5):
            x += 7
        else:
            x -= 3
    return x
```

ACCEPTED

2. *Testcase kedua*

```

from s.tr import x
import m as y

if(x > 5):
    print("helloo ")

else:
    print("whiyyyy")
    a = 7

```

ACCEPTED

3. Testcase ketiga

```

#comment bisa

#hahahaha

while(i < 5):
    print(i)

```

ACCEPTED

4. Testcase keempat

```

elif(ggg):
    print("hello")

```

Syntax Error on line : 1
you cant use elif without if

5. Testcase kelima

```
for i in range(5):  
    print(555)  
    return i
```

```
Python 3.6.4 Shell  
Syntax Error on line : 3  
you can't use return! you didn't define any function!
```

BAB IV

REFERENSI

Dari Buku

Dari Internet

Binus.ac.id. (2018, 20 Desember). *Penyederhanaan Context-Free Grammar*. Diakses pada 26 November 2019, dari <https://socs.binus.ac.id/2018/12/20/penyederhanaan-context-free-grammar/>

PetaniKode.com. (2018, 15 September). *Belajar Pemrograman Python: Pengenalan Dasar Python dan Persiapan Awal*. Diakses pada 25 November 2019, dari <https://www.petanikode.com/python-linux/>

Unila.ac.id. (2016, 29 Juni). *Push Down Automata (Ekivalensi PDA dan CFG serta deterministic PDA)*. Diakses pada 27 November 2019, dari <http://web.if.unila.ac.id/resalinaoktaria/2016/06/29/push-down-automata-ekivalensi-pda-dan-cfg-serta-deterministic-pda/>

Wordpress.com. (2011, 23 Juni). *Bentuk Normal Chomsky*. Diakses pada 27 November 2019, dari <https://fairuzelsaid.wordpress.com/2011/06/23/bentuk-normal-chomsky/>