LAPORAN TUGAS BESAR
IF2124 TEORI BAHASA FORMAL DAN OTOMATA
COMPILER BAHASA PYTHON

Oleh

Kelompok 10

Ahmad Romy Zahran	13520009
Aji Andika Falah	13520012
Primanda Adyatma Hafiz	13520022

Daftar Isi

BAB 1
TEORI DASAR

1.1 Teori Dasar FA

Finite Automata atau biasa disebut dengan FA adalah mesin yang terdiri atas beberapa state yang terhitung jumlahnya dan pada setiap state terdapat fungsi transisi yang menerima sebuah input tertentu serta pindah ke state lain sesuai dengan input tersebut. FA adalah model yang baik untuk komputer yang memiliki jumlah memori yang terbatas. Contoh penerapan FA dalam kehidupan sehari-hari yaitu pada pintu otomatis dan lift. Suatu string akan diterima oleh FA jika dan hanya jika saat string tersebut habis diproses FA berada pada posisi final state.

Dalam program Compiler Python yang kami buat, kami menggunakan variabel pada python untuk mencatat state dari FA setiap kali state berubah. Nantinya ketika program masuk ke final state (input telah habis) ataupun state buangan maka akan langsung direturn apakah string tersebut diterima atau tidak berdasarkan apakah FA berada pada final state atau state buangan.

1.2 Teori Dasar CFG

CFG, yaitu singkatan dari Context Free-Grammar, adalah tata bahasa formal yang aturan produksinya dalam bentuk

$A \rightarrow B$

dengan A adalah simbol yang memproduksi, sebuah variabel non-terminal, dan B adalah hasil produksi. B bisa dalam bentuk variabel non-terminal, terminal, maupun string kosong, dan jumlahnya boleh lebih dari satu. Definisi formal dari sebuah CFG G adalah

$G = (V, \sum, R, S)$

dengan V adalah himpunan variabel non-terminal, $\sum$ adalah himpunan terminal, R adalah himpunan relasi (himpunan produksi), dan S adalah *start symbol*.

Contoh dari CFG adalah

G = ({S, A}, {a, b}, R, S), dimana R adalah

$S \rightarrow Aab \mid abAS$

$A \rightarrow ba \mid ab \mid aA$

1.3 Teori Dasar CNF

Chomsky Normal Form, atau disingkat CNF, adalah sebuah CFG yang semua aturan produksinya dalam bentuk

$A \rightarrow BC$ atau

$A \rightarrow a$

dimana A, B, dan C adalah variabel non-terminal, dan a adalah terminal. Sebuah CFG dapat dikonversi menjadi CNF dengan langkah-langkah berikut:

1. Membuat produksi baru S0 → S dengan S adalah start symbol baru.
2. Menghapus null, unit, dan usless production.
3. Menghapus terminal pada RHS jika ada bersamaan dengan non-terminal atau terminal. Jika terdapat A → aB, maka diubah menjadi A →CB dan C → a.
4. Menghapus RHS yang memiliki lebih dari 2 non-terminal. Jika terdapat S → ABC, maka diubah menjadi S → DC dan D → AB.

1.4 Teori Dasar CYK

Cocke-Younger Kasami (CYK) adalah algoritma untuk mengetes suatu string termasuk dalam sebuah Context-free Language (CFL) L atau tidak. Sebelumnya, grammar G harus dalam bentuk CNF. String w dengan panjang n diproses dengan meninjau setiap substring dari panjang terpendek dan menyocokkan nya dengan aturan produksi yang ada. Hasil pencocokan itu disimpan dalam tabel triangular sehingga bisa disebut table-filling algorithm juga dengan $X_{i,j} = variabel\ yang\ cocok\ dengan\ substring\ a_i..a_j$. Variabel yang cocok dengan substring yang lebih panjang diperoleh dari dua substring yang lebih pendek karena grammar dalam bentuk CNF atau

$$X_{i,j} \rightarrow X_{i,k}X_{k,j}, \text{ dengan } i \leq k < j$$

Hal ini sesuai juga dengan konsep dynamic programming. Kompleksitas algoritma CYK bila mengabaikan pengali konstan dari ukuran CNF adalah $O(N^3)$ dengan N panjang string masukan karena ada orde ukuran tabel N^2 dan setiap sel maksimal ada N kemungkinan pencocokan .

BAB 2
SOLUSI PROGRAM

2.1 Solusi FA

Pada program Compiler Python yang dibuat, FA digunakan untuk memproses data input dari file. Nantinya setiap karakter yang berasal dari input file akan dicocokkan menggunakan FA menjadi variabel non terminal sehingga selanjutnya dapat langsung diolah dengan menggunakan CFG. Solusi FA yang kami buat adalah sebagai berikut:

Variabel Terminal	Solusi NFA
variable	State awal akan mengecek apakah karakter awal merupakan huruf besar, huruf kecil, atau underscore. Jika tidak memenuhi maka program akan pindah ke state buangan. Jika memenuhi akan pindah state dan terus dilakukan pengecekan apakah setiap karakter merupakan huruf besar, huruf kecil, undescore, atau angka. Jika pada salah satu karakter kondisi tersebut tidak terpenuhi maka program akan pindah ke state buangan. Jika terpenuhi maka pogram tetap berada di state yang sama dan sekaligus menjadi final state.
number	State awal akan mengecek apakah karakter terdepan merupakan karakter angka. Jika tidak dipenuhi maka program akan pindah ke state buangan. Jika terus terpenuhi maka program akan tetap berada di state yang sama yang sekaligus menjadi final state.
string	State awal akan mengecek apakah karakter pertama adalah ‘. Jika terpenuhi maka program akan masuk ke state baru dan terus melakukan pengecekan apakah karakter selanjutnya adalah ‘. Jika karakter selanjutnya adalah ‘ maka program akan masuk ke final state. Jika tidak maka akan dilakukan pengecekan apakah karakter pertama adalah “. Jika terpenuhi maka program akan masuk ke state baru dan terus melakukan pengecekan apakah karakter selanjutnya adalah “. Jika tidak ditemukan “ maka program akan masuk ke final state. Jika tidak maka program akan masuk ke state buangan.
comment	State awal akan mengecek apakah karakter awalan dari masukan adalah #. Jika tidak terpenuhi program akan pindah ke state buangan. Jika terpenuhi program akan terus melakukan pengecekan apakah ditemukan karakter \n atau akhir dari masukan. Jika ditemukan maka program akan langsung masuk ke final state.
multiline comment	State awal akan mengecek apakah 3 karakter pertama adalah ‘. Jika terpenuhi maka program akan masuk ke state baru dan terus melakukan pengecekan apakah 3 karakter selanjutnya adalah ‘. Jika 3 karakter selanjutnya adalah ‘ maka program akan masuk ke final state. Jika tidak maka akan dilakukan pengecekan apakah 3 karakter pertama adalah “. Jika terpenuhi maka program akan masuk ke state baru dan terus melakukan pengecekan apakah 3 karakter selanjutnya adalah “. Jika 3

	karakter selanjutnya adalah “ maka program akan masuk ke final state. Jika tidak ditemukan maka program akan masuk ke state buangan.
newline	State awal akan mengecek apakah string merupakan \n. Jika terpenuhi maka akan masuk ke final state. Jika tidak maka akan masuk ke state buangan

2.2 Solusi CFG

CFG yang dibuat terdapat dalam file cfg.txt dan nantinya CFG akan diolah oleh program untuk dikonversi menjadi CNF. Hasil dari CFG yang kami buat adalah sebagai berikut:

Terminal Variable (T)

True	False	variable	class	is	return	None
continue	pass	break	for	def	from	import
while	and	or	not	with	as	if
elif	else	range	print	open	in	raise
string	number	newline	comment	+	-	*
/	=	(	)	<	>	%
:	‘	“	,	.	#	[
]	&	|	^	~	!	

Non Terminal Variable (V)

S	VAR	VARARR	VAROPS
FORLOOP	RANGE	WHILELOOP	EXPRESSION
IMPORT	RETURN	WITH	DEF
FUNCTION	PRINT	INBRACKET	OPERATOR
OPERATORBOOL	COMMENT	BOOL	IF
ELIF	ELSE	INBRACKET	STRING
CLASS	METHOD	OPERATORCOMPARISON	INPUTFUNC
RAISE	NEWLINE	INBRACKETEXPRESSION	VARARR
MERGEVAROPS	ONELINERIF	CONTROLLOOP	DECLAREARR
MULTICONTROLOOP	NUMBER	FORLOOPCONTROL	
WHILELOOPCONTROL		FORLOOPMULTICONTROL	
DEFMULTIRETURN		WHILELOOPMULTICONTROL	
MULTIRETURN		INDEXINGARR	
INSQUAREBRACKET		MULTILINECOMMENT	

Production (P)

Production	Hasil
S	S NEWLINE S | S NEWLINE | NEWLINE S | S COMMENT | VARLEFT = VAROPS | VARLEFT + = VAROPS | VARLEFT - = VAROPS | VARLEFT * = VAROPS | VARLEFT ^ = VAROPS | VARLEFT &

	= VAROPS | VARLEFT > > = VAROPS | VARLEFT < < = VAROPS | VARLEFT / / = VAROPS | VARLEFT / = VAROPS | VARLEFT % = VAROPS | VARLEFT * * = VAROPS | FORLOOP | FORLOOPCONTROL | FORLOOPMULTICONTROL | WHILELOOP | WHILELOOPCONTROL | WHILELOOPMULTICONTROL | IF | ELIF | ELSE | IMPORT | WITH | DEF | DEFRETURN | DEFMULTIRETURN | FUNCTION | METHOD | PRINT | COMMENT | MULTILINECOMMENT | RAISE | ONELINERIF | CLASS
VAR	variable
VARLEFT	VAR | VARARR | VARDOT
VARDOT	VAR | VAR . VARDOT
VARARR	variable [INDEXINGARR] | VARARR [INDEXINGARR]
NEWLINE	NEWLINE newline | newline
BOOL	True | False | not BOOL
OPERATOR	+ | - | * | / | * * | / / | % | & | ^ | < < | > > | |
OPERATORBOOL	and | or
OPERATORCOMPARISON	< | > | < = | > = | = = | is | in | not in | is not | !=
NUMBER	NUMBER -> number | - number
VAROPS	- VAROPS | VAR | VARARR | NUMBER | STRING | BOOL | FUNCTION | METHOD | DECLAREARR | VAROPS OPERATOR VAROPS | MERGEEXPRESSION | VAR (METHOD) | VAR (METHOD INSQUAREBRACKET) | ~ VAROPS | INBRACKET | None
INBRACKET	(INBRACKET) | (VAROPS) | - INBRACKET | (- INBRACKET)
MERGEVAROPS	VAROPS | INBRACKET
RANGE	range INBRACKET
FORLOOP	for VAR in RANGE : | for VAR in VAR :
FORLOOPCONTROL	FORLOOP S CONTROLLOOP | FORLOOP NEWLINE CONTROLLOOP
FORLOOPMULTICONTROL	FORLOOP MULTICONTROLLOOP
EXPRESSION	BOOL | VAR | VARARR | VAROPS OPERATOR VAROPS | VAROPS OPERATORCOMPARISON VAROPS | VAROPS OPERATORBOOL VAROPS | not EXPRESSION
INBRACKETEXPRESSION	(EXPRESSION) | (INBRACKETEXPRESSION) | not INBRACKETEXPRESSION
MERGEEXPRESSION	EXPRESSION | INBRACKETEXPRESSION | ~ MERGEEXPRESSION
WHILELOOP	while MERGEEXPRESSION :
WHILELOOPCONTROL	WHILELOOP S NEWLINE CONTROLLOOP | WHILELOOP NEWLINE CONTROLLOOP

WHILELOOPMULTICONTROL	WHILELOOP MULTICONTROLLOOP
IMPORT	from VARDOT import VARDOT | from VARDOT import VARDOT as VAR | from VARDOT import * | from . VAR import VAR | from . VAR import VARDOT as VAR | from . VAR import * | import VARDOT as VAR | import VARDOT
RETURN	return MERGEVAROPS | return MERGEEXPRESSION | return
STRING	string | STRING string
WITH	with open (INPUTFUNC) as VAR :
INPUTFUNC	VAROPS | INPUTFUNC , INPUTFUNC
FUNCTION	VAR ( ) | VAR (INPUTFUNC)
DEF	def FUNCTION :
DEFRETURN	DEF S RETURN | DEF NEWLINE RETURN
DEFMULTIRETURN	DEF NEWLINE MULTIRETURN
PRINT	print INBRACKET
IF	if MERGEEXPRESSION :
ELIF	elif MERGEEXPRESSION :
ELSE	else :
ONELINERIF	VAR = VAROPS if MERGEEXPRESSION else VAROPS
CLASS	class VAR :
METHOD	VAR . FUNCTION | VAR . VAR | VAR . VARARR | FUNCTION . FUNCTION
COMMENT	comment
MULTILINECOMMENT	multilinecomment
RAISE	raise INBRACKET
CONTROLLOOP	pass | break | continue
MULTICONTROLLOOP	S CONTROLLOOP | CONTROLLOOP | MULTICONTROLLOOP NEWLINE MULTICONTROLLOOP
MULTIRETURN	S newline RETURN | S NEWLINE RETURN | RETURN | MULTIRETURN NEWLINE MULTIRETURN
INDEXINGARR	: | VAROPS : VAROPS | VAROPS : | : VAROPS | VAROPS
INSQUAREBRACKET	[INSQUAREBRACKET , INSQUAREBRACKET] | [INPUTFUNC]
DECLAREARR	[ ] | [VAROPS for VAR in RANGE] | [VAROPS for VAR in VAR] | [DECLAREARR] | [DECLAREARR for VAR in RANGE] | [DECLAREARR for VAR in VAR] | INSQUAREBRACKET

BAB 3
IMPLEMENTASI DAN PENGUJIAN

3.1 Implementasi Program

File main.py

File main.py menerima masukan path dari file yang akan dilakukan pengecekan syntax melalui command line pada saat proses compile. Setelah itu program akan memanggil fungsi parser yang terdapat pada lexer.py kemudian hasilnya akan dimasukkan ke fungsi cyk yang diimport dari cyk.py. Selain itu juga dilakukan pengecekan conditional dengan memanggil fungsi validasi conditional yang terdapat pada lexer. Jika cyk dan conditional valid maka akan diberikan output Accepted, jika tidak akan diberikan output Syntax Error .

File lexer.py

File lexer.py berisi fungsi untuk parsing teks dari input file serta FA untuk pengecekan variabel, string, angka, dan comment. Fungsi dan prosedur yang digunakan dalam program ini adalah :

1. conditional_validation
   Fungsi ini akan membaca seluruh if, elif, else yang terdapat pada file input kemudian jika terdapat kondisi prefix else lebih besar dibandingkan dari prefix if atau elif muncul sebelum if maka fungsi akan mengembalikan False, selain kasus tersebut maka fungsi akan mengembalikan True.
2. is_variable
   Fungsi ini merupakan implementasi FA untuk pengecekan variabel. Jika masukan merupakan variabel maka akan dikembalikan True dan jika tidak dikembalikan False.
3. is_number
   Fungsi ini merupakan implementasi FA untuk pengecekan angka. Jika masukan merupakan angka maka akan dikembalikan True dan jika tidak dikembalikan False.
4. is_string
   Fungsi ini merupakan implementasi FA untuk pengecekan string. Jika masukan merupakan string maka akan dikembalikan indeks dari tanda kutip terakhir dan jika tidak dikembalikan 0.
5. is_newline
   Fungsi ini merupakan implementasi FA untuk pengecekan karakter \n. Jika masukan merupakan \n maka akan dikembalikan True dan jika tidak dikembalikan False.
6. is_comment
   Fungsi ini merupakan implementasi FA untuk pengecekan comment. Jika masukan merupakan comment maka akan dikembalikan indeks sebelum karakter \n atau jika comment berada pada akhir teks maka akan dikembalikan indeks dari karakter terakhir. Jika bukan merupakan comment maka akan dikembalikan 0.
7. is_multilinecomment
   Fungsi ini merupakan implementasi FA untuk pengecekan multiline comment. Jika masukan merupakan multiline comment maka akan dikembalikan indeks sebelum karakter terakhir dari tanda kutip penutup dan jika bukan merupakan multiline comment maka akan dikembalikan 0.

File cyk.py

File cyk.py berisi implementasi algoritma CYK dalam bentuk fungsi. Terdapat 3 fungsi, yaitu:

1. LoadCNF
   Menerima path file txt berisi aturan produksi CNF dan mengembalikan dictionary dengan format {$Body_1$: [$Head_{1,1}$, $Head_{1,2}$, ...], ...}. Prosesnya dengan memisahkan Head dan Body/Bodies dengan delimiter ' -> ', memisahkan Bodies dengan delimiter ' | ', memasukan pasangan Head dan Body sesuai format. Bila Body berupa konkatenasi 2 variabel, misal AB, maka Body disimpan sebagai string 'A B'.
2. cyk
   Menerima tokens dan chomskyDict, tokens berupa list token hasil parsing kode sesuai terminal dan implementasi FA, chomskyDict berupa dictionary aturan produksi CNF dengan format seperti output fungsi LoadCNF. Fungsi ini mengembalikan tabel triangular cykTable dengan isi sel $X_{i,j}$ berupa set bila tidak kosong dan berupa list bila kosong. Digunakan set agar variabel hasil pencocokan tidak berulang. Proses pengisian tabel dilakukan sesuai teori dasar, dengan pencarian Head yang cocok menggunakan dictionary chomskyDict sehingga pencarian cepat. cykTable[i][j] bersesuaian dengan substring [j..j+i] dengan aturan indeks mulai dari 0.
3. verdict
   Menerima cykTable dan mengembalikan True jika dan hanya jika start symbol S ada di X_{1n}.

File CFG2CNF.py

File CFG2CNF.py adalah program python yang merubah bentuk Context Free-Grammar (CNF) menjadi Chomsky Normal Form (CNF). Program akan mengonversi CFG dalam file cfg.txt menjadi CNF dan disimpan di file cnf.txt. Fungsi dan prosedur yang digunakan dalam program ini adalah:

1. loadAll
   Prosedur ini membaca CFG (terminal, variable, dan rules) dalam file cfg.txt dan mengubahnya menjadi array of tuple yang dapat diolah. Misal sebuah cfg.txt berisi rules
   S → A B | a b
   A → b
   B → a
   maka prosedur ini akan merubahnya menjadi
   [('S',['A','B']), ('S',['a','b']), ('A',['b']), ('B',['a'])]

2. first
   Prosedur ini menambahkan variabel S0 ke variable dan S0 → S ke rules yang telah dibaca oleh loadAll.

3. moreThanTwo
   fungsi ini mengecek jika ada rules yang memiliki hasil lebih dari 2 tanpa melihat bentuknya (terminal atau variabel). Jika ada, akan menghasilkan True, jika tidak, akan menghasilkan False. Contohnya A → aAb akan menghasilkan True, dan B→ bB akan menghasilkan False.

4. termVar

termVar adalah fungsi yang menghasilkan True jika ada rules yang menghasilkan terminal dan variabel, atau terminal dan terminal. Jika tidak, akan menghasilkan False.

5. unit
   Fungsi unit adalah fungsi yang menghasilkan True jika ada unit Production, dan False jika tidak.

6. delRightMoreThanTwo
   Prosedur ini adalah prosedur yang menghapus rules yang menghasilkan lebih dari 2 dan menggantikan 2 pertama menjadi variabel, dan sisanya menjadi variabel berbeda. Jika menghasilkan 3, hanya 2 pertama yang dirubah. Misalkan ada rules
   S → a A B B b | a b | A | B
   A → c
   B → d

   Rules tersebut akan diganti menjadi
   S → V1 V2 | a b | A | B
   A → c
   B → d
   V1 → a A
   V2 → B B b

   dan akan dirubah lagi menjadi
   S → V1 V2 | a b | A | B
   A → c
   B → d
   V1 → a A
   V2 → V3 b
   V3 → B B

7. delTermVar
   Prosedur ini akan merubah rules yang menghasilkan variabel terminal atau terminal terminal menjadi variabel variabel. Contohnya rules hasil dari delRightMoreThanTwo diatas akan dirubah menjadi
   S → V1 V2 | V4 V5 | A | B
   A → c
   B → d
   V1 → V4 A
   V2 → V3 V5
   V3 → B B
   V4 → a
   V5 → b

8. delUnit
   Prosedur delUnit adalah prosedur yang menghaspus unit production. Jika rules hasil dari delTermVar diatas dimasukkan ke delUnit, hasilnya akan menjadi
   S → V1 V2 | V4 V5 | c | d
   A → c

B → d
V1 → V4 A
V2 → V3 V5
V3 → B B
V4 → a
V5 → b

9. writeRules
   Prosedur ini adalah prosedur yang menerima CNF dari hasil pengolahan, dan menuliskannya pada cnf.txt

3.2 Pengujian Program

Percabangan (if, elif, else)

Kasus percabangan diterima :

```
x=2
if x>3:
    if (x>5):
        x+=1
    elif x>2:
        x+=2
    else:
        print(x+x**2)
else:
    x=1
```

Hasil :

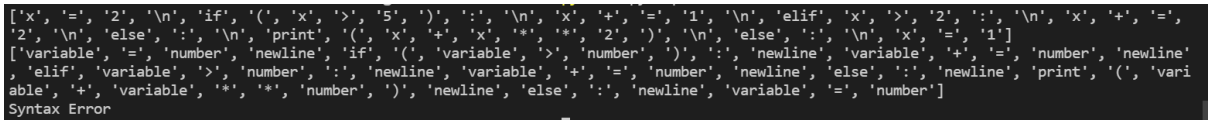
```
['x', '=', '2', '\n', 'if', 'x', '>', '3', ':', '\n', 'if', '(', 'x', '>', '5', ')', ':', '\n', 'x', '+', '=', '1', '\n', 'elif', 'x', '>',
 '2', ':', '\n', 'x', '+', '=', '2', '\n', 'else', ':', '\n', 'print', '(', 'x', '+', 'x', '*', '*', '2', ')', '\n', 'else', ':', '\n', 'x'
, '=', '1']
['variable', '=', 'number', 'newline', 'if', 'variable', '>', 'number', ':', 'newline', 'if', '(', 'variable', '>', 'number', ')', ':', 'ne
wline', 'variable', '+', '=', 'number', 'newline', 'elif', 'variable', '>', 'number', ':', 'newline', 'variable', '+', '=', 'number', 'newl
ine', 'else', ':', 'newline', 'print', '(', 'variable', '+', 'variable', '*', '*', 'number', ')', 'newline', 'else', ':', 'newline', 'varia
ble', '=', 'number']
Accepted
```

Percabangan berhasil ditokenisasi dan diterima karena percabangannya valid dan tidak ada kesalahan syntax.

Kasus percabangan tidak diterima :

```
x=2
if (x>5):
    x+=1
    elif x>2:
        x+=2
    else:
        print(x+x**2)
else:
    x=1
```

Hasil :



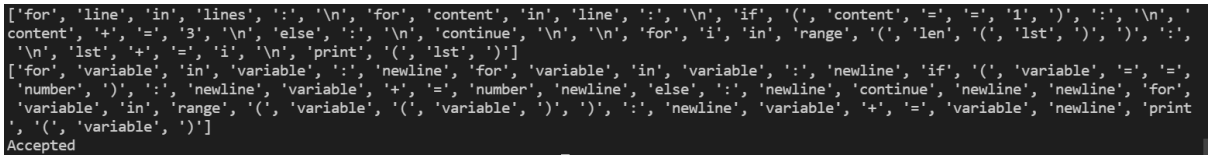
Program di atas tidak diterima karena terjadi syntax error disebabkan jumlah else melebihi jumlah if

For Loop

Kasus for loop yang diterima :

```
for line in lines:
    for content in line:
        if (content==1):
            content+=3
        else:
            continue

for i in range(len(lst)):
    lst+=i
    print(lst)
```

Hasil :



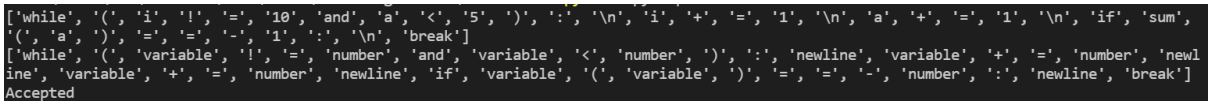
For loop berhasil diterima karena struktur dari for loop valid disebabkan memuat variabel iterator, range iterasi, dan tanda titik dua.

While Loop

Kasus while loop yang diterima :

```
while (i!=10 and a<5):
    i+=1
    a+=1
    if sum(a)==-1:
        break
```

Hasil :



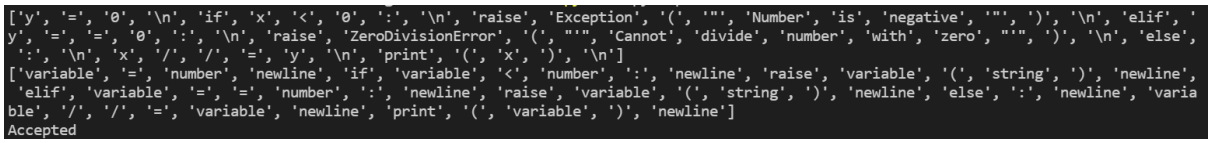
While loop berhasil diterima karena struktur dari while loop lengkap yaitu terdiri dari kata while, ekspresi, dan tanda titik dua.

Raise

Kasus raise yang diterima :

```
y = 0
if x < 0:
    raise Exception("Number is negative")
elif y == 0:
    raise ZeroDivisionError('Cannot divide number with zero')
else:
    x//=y
    print(x)
```

Hasil :



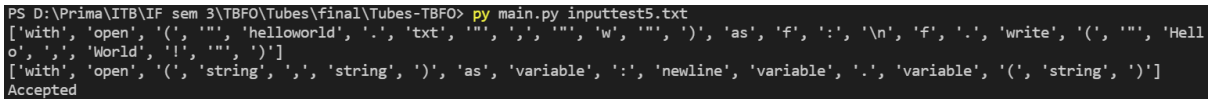
Raise berhasil diterima karena struktur dari raise lengkap yaitu memuat kata raise dan fungsi yang menyatakan error.

With

Kasus with yang diterima :

```
with open("helloworld.txt", "w") as f:
    f.write("Hello, World!")
```

Hasil :



With dapat diterima oleh program dan ditokenisasi dengan baik oleh lexer

Comment

Kasus program yang memiliki comment dan multiline comment pada comment.py

```
# This is a comment
''' This is
a multiline
comment '''
```

Hasil:

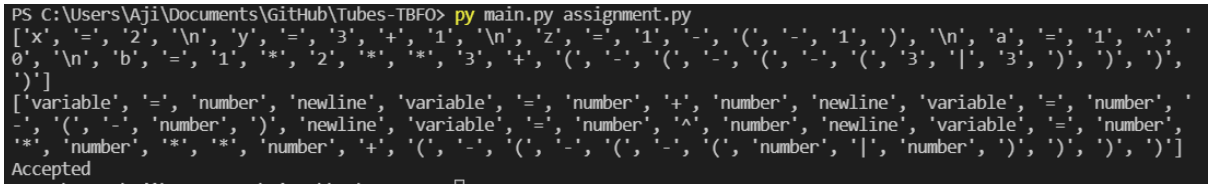
Hasil dari program tersebut “Accepted” karena untuk comment, dideteksi newline, sedangkan untuk multiline comment, terdeteksi tiga apostrof (‘) diawal multilline comment dan terdeteksi tiga apostrof lagi di akhir multiline comment.

Variable Assignment

Kasus program yang memiliki variable assignment pada assignment.py

```
x = 2
y = 3 + 1
z = 1 - (-1)
a = 1 ^ 0
b = 1 * 2 ** 3 + (-(-(-(3 | 3))))
```

Hasil:



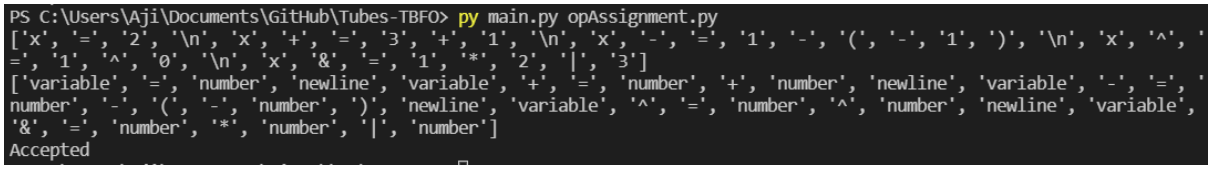
Hasilnya diterima dengan assignment yang memiliki operator dan negatif didalam kurung.

Variable Operator Assignment

Kasus program yang memiliki operator assignment pada opAssignment.py

```
x = 2
x += 3 + 1
x -= 1 - (-1)
x ^= 1 ^ 0
x &= 1 * 2 | 3
```

Hasil:



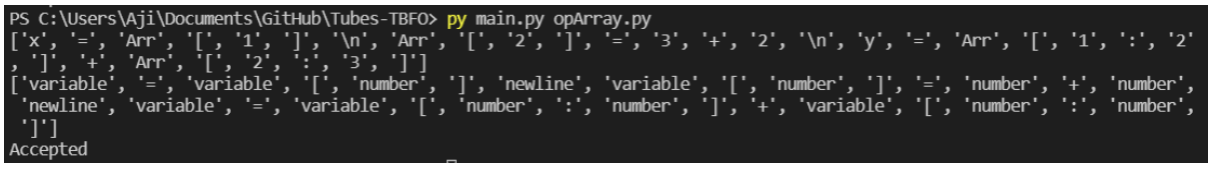
Hasilnya diterima dengan operator assignment arithmetic dan bitwise.

Operasi dan Indexing Array

Kasus program dengan operasi dan indexing array pada file opArray.py

```
x = Arr[1]
Arr[2] = 3 + 2
y = Arr[1:2] + Arr[2:3]
```

Hasil:



Hasil program dengan operasi dan indexing array diterima.

Deklarasi Array

Kasus program dengan deklarasi array pada file array.py

```
w = []
x = [0 for i in range(2)]
y = [x for j in range(20)]
z = [['z' for a in range(3)] for b in range(5)]
v = [[] for c in x]
```

Hasil:

```
PS C:\Users\Aji\Documents\GitHub\Tubes-TBFO> py main.py array.py
['w', '=', '[', ']', '\n', 'x', '=', '[', '0', 'for', 'i', 'in', 'range', '(', '2', ')', ']', '\n', 'y', '=', '[', 'x
', 'for', 'j', 'in', 'range', '(', '20', ')', ']', '\n', 'z', '=', '[', '[', "'", 'z', "'", 'for', 'a', 'in', 'range'
, '(', '3', ')', ']', 'for', 'b', 'in', 'range', '(', '5', ')', ']', '\n', 'v', '=', '[', '[', ']', 'for', 'c', 'in',
 'x', ']']
['variable', '=', '[', ']', 'newline', 'variable', '=', '[', 'number', 'for', 'variable', 'in', 'range', '(', 'number
', ')', ']', 'newline', 'variable', '=', '[', 'variable', 'for', 'variable', 'in', 'range', '(', 'number', ')', ']',
'newline', 'variable', '=', '[', '[', 'string', 'for', 'variable', 'in', 'range', '(', 'number', ')', ']', 'for', 'va
riable', 'in', 'range', '(', 'number', ')', ']', 'newline', 'variable', '=', '[', '[', ']', 'for', 'variable', 'in',
'variable', ']']
Accepted
```

Hasil deklarasi array diterima, dari array kosong, array of array, juga menggunakan range() dan in.

Deklarasi Fungsi

Kasus fungsi dengan banyak return (multi return) dalam def1.py

```
def fungsi(arg1, arg2, arg3):
    if(arg2<0):
        return None
    res = arg1
    for i in range(20):
        if(res>arg3):
            break
        else:
            res += arg2
    return arg1

ans = fungsi(1,2,3)
print(ans)
```

Hasil:

```
PS D:\Python parser\Tubes-TBFO> python main.py def1.py
['def', 'fungsi', '(', 'arg1', ',', 'arg2', ',', 'arg3', ')', ':', '\n', 'if', '(', 'arg2', '<', '0', ')', ':', '\n', 'return', 'None', '\n', 'res', '=', 'arg1'
, '\n', 'for', 'i', 'in', 'range', '(', '20', ')', ':', '\n', 'if', '(', 'res', '>', 'arg3', ')', ':', '\n', 'break', '\n', 'else', ':', '\n', 'res', '+', '=',
'arg2', '\n', 'return', 'arg1', '\n', '\n', 'ans', '=', 'fungsi', '(', '1', ',', '2', ',', '3', ')', '\n', 'print', '(', 'ans', ')']
['def', 'variable', '(', 'variable', ',', 'variable', ',', 'variable', ')', ':', 'newline', 'if', '(', 'variable', '<', 'number', ')', ':', 'newline', 'return',
 'None', 'newline', 'variable', '=', 'variable', 'newline', 'for', 'variable', 'in', 'range', '(', 'number', ')', ':', 'newline', 'if', '(', 'variable', '>', 'v
ariable', ')', ':', 'newline', 'break', 'newline', 'else', ':', 'newline', 'variable', '+', '=', 'variable', 'newline', 'return', 'variable', 'newline', 'newlin
e', 'variable', '=', 'variable', '(', 'number', ',', 'number', ',', 'number', ')', 'newline', 'print', '(', 'variable', ')']
Accepted
```

Def dengan banyak return berhasil diterima dan ditokenisasi oleh lexer dengan baik.

Kasus return tanpa def dalam def2.py:

```
a = input()
return a
```

Hasil:

```
PS D:\Python parser\Tubes-TBFO> python main.py def2.py
['a', '=', 'input', '(', ')', '\n', 'return', 'a']
['variable', '=', 'variable', '(', ')', 'newline', 'return', 'variable']
Syntax Error
PS D:\Python parser\Tubes-TBFO>
```

Return tanpa def menghasilkan syntax error.

Kasus def dalam def dalam def3.py:

```
def fungsi1(arg1, arg2):
    def fungsi2(arg3):
        return 5*arg3
    arg1 = fungsi2(arg1)
    arg2 = fungsi2(arg2)
    return (arg1-arg2)
```

Hasil:

```
PS D:\Python parser\Tubes-TBFO> python main.py def3.py
['def', 'fungsi1', '(', 'arg1', ',', 'arg2', ')', ':', '\n', 'def', 'fungsi2', '(', 'arg3', ')', ':', '\n', 'return', '5', '*', 'arg3', '\n', 'arg1', '=', 'fung
si2', '(', 'arg1', ')', '\n', 'arg2', '=', 'fungsi2', '(', 'arg2', ')', '\n', 'return', '(', 'arg1', '-', 'arg2', ')']
['def', 'variable', '(', 'variable', ',', 'variable', ')', ':', 'newline', 'def', 'variable', '(', 'variable', ')', ':', 'newline', 'return', 'number', '*', 'va
riable', 'newline', 'variable', '=', 'variable', '(', 'variable', ')', 'newline', 'variable', '=', 'variable', '(', 'variable', ')', 'newline', 'return', '(', '
variable', '-', 'variable', ')']
Accepted
PS D:\Python parser\Tubes-TBFO>
```

Def dalam def diterima dengan baik.

Penggunaan Fungsi dan Method

Kasus method:

```
dog1 = animal()
dog1.sound()
print(dog1.species)
print(dog1.age)
```

Hasil:

```
PS D:\Python parser\Tubes-TBFO> python main.py test2.py
['dog1', '=', 'animal', '(', ')', '\n', 'dog1', '.', 'sound', '(', ')', '\n', 'print', '(', 'dog1', '.', 'species', ')', '\n', 'print', '(', 'dog1', '.', 'age',
')']
['variable', '=', 'variable', '(', ')', 'newline', 'variable', '.', 'variable', '(', ')', 'newline', 'print', '(', 'variable', '.', 'variable', ')', 'newline',
'print', '(', 'variable', '.', 'variable', ')']
Accepted
PS D:\Python parser\Tubes-TBFO>
```

Metode diterima dengan baik.

Import

Kasus import:

```
import matplotlib.pyplot as plt
import os
from numpy import *
from sys import path
plt.show()
```

Hasil:

```
PS D:\Python parser\Tubes-TBFO> python main.py test.py
['import', 'matplotlib', '.', 'pyplot', 'as', 'plt', '\n', 'import', 'os', '\n', 'from', 'numpy', 'import', '*', '\n', 'from', 'sys', 'import', 'path', '\n', 'p
lt', '.', 'show', '(', ')']
['import', 'variable', '.', 'variable', 'as', 'variable', 'newline', 'import', 'variable', 'newline', 'from', 'variable', 'import', '*', 'newline', 'from', 'var
iable', 'import', 'variable', 'newline', 'variable', '.', 'variable', '(', ')']
Accepted
PS D:\Python parser\Tubes-TBFO>
```

Semua cara import diterima dengan baik.

Class

Kasus kelas:

```
class animal:
    def __init__(self, species, name, age):
        self.species = species
        self.name = name
        self.age = age
        return a

    species = 'apa'
    name = 'apa'
    age = 10

hwn1 = animal('anjing','heli',1)
```

Hasil:

```
PS D:\Python parser\Tubes-TBFO> python main.py test.py
['class', 'animal', ':', '\n', 'def', '__init__', '(', 'self', ',', 'species', ',', 'name', ',', 'age', ')', ':', '\n', 'self', '.', 'species', '=', 'species',
'\n', 'self', '.', 'name', '=', 'name', '\n', 'self', '.', 'age', '=', 'age', '\n', 'return', 'a', '\n', '\n', 'species', '=', "'", 'apa', "'", '\n', 'name', '=
', "'", 'apa', "'", '\n', 'age', '=', '10', '\n', '\n', 'hwn1', '=', 'animal', '(', "'", 'anjing', "'", ',', "'", 'heli', "'", ',', '1', ')']
['class', 'variable', ':', 'newline', 'def', 'variable', '(', 'variable', ',', 'variable', ',', 'variable', ',', 'variable', ')', ':', 'newline', 'variable', '.
', 'variable', '=', 'variable', 'newline', 'variable', '.', 'variable', '=', 'variable', 'newline', 'variable', '.', 'variable', '=', 'variable', 'newline', 're
turn', 'variable', 'newline', 'newline', 'variable', '=', 'string', 'newline', 'variable', '=', 'string', 'newline', 'variable', '=', 'number', 'newline', 'newl
ine', 'variable', '=', 'variable', '(', 'string', ',', 'string', ',', 'number', ')']
Accepted
PS D:\Python parser\Tubes-TBFO>
```

Class diterima dan ditokenisasi dengan baik.

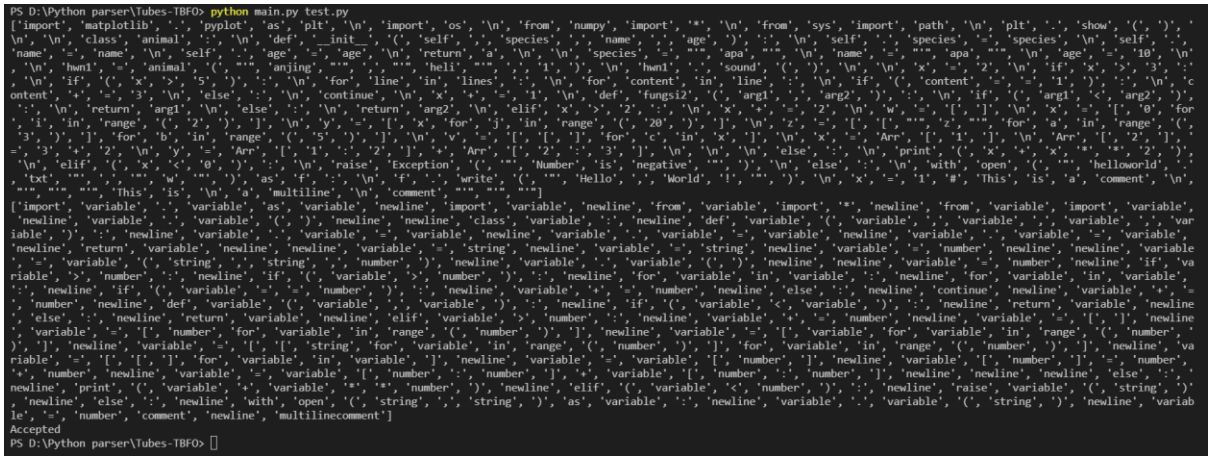
Gabungan Keseluruhan

Gabungan semua aspek:

```
import matplotlib.pyplot as plt
import os
from numpy import *
from sys import path
plt.show()

class animal:
    def __init__(self, species, name, age):
        self.species = species
        self.name = name
        self.age = age
        return a

    species = 'apa'
    name = 'apa'
    age = 10

hwn1 = animal('anjing','heli',1)
hwn1.sound()
```

```
x=2
if x>3:
    if (x>5):
        for line in lines:
            for content in line:
                if (content==1):
                    content+=3
                else:
                    continue
        x+=1
        def fungsi2(arg1,arg2):
            if(arg1<arg2):
                return arg1
            else:
                return arg2
    elif x>2:
        x+=2
        w = []
        x = [0 for i in range(2)]
        y = [x for j in range(20)]
        z = [['z' for a in range(3)] for b in range(5)]
        v = [[] for c in x]
        x = Arr[1]
        Arr[2] = 3 + 2
        y = Arr[1:2] + Arr[2:3]


    else:
        print(x+x**2)
elif(x<0):
    raise Exception("Number is negative")
else:
    with open("helloworld.txt", "w") as f:
        f.write("Hello, World!")
    x=1 # This is a comment
    ''' This is
    a multiline
    comment '''
```

Hasil:



Semua tetap diterima dan ditokenisasi dengan baik setelah disatukan dan dicampur.

LAMPIRAN

Link ke Repository Github

https://github.com/primahafiz/Tubes-TBFO

Daftar Pembagian Tugas

Nama	Pembagian Tugas
Ahmad Romy Zahran (13520009)	Implementasi algoritma CYK pada cyk.py. Membuat laporan bab 1 bagian CYK, bab 3 bagian cyk.py, dan pengujian.
Aji Andika Falah (13520012)	Membuat CFG2CNF.py dan mengecek jika hasilnya benar, membuat laporan bab 1 bagian CFG dan CNF, bab 3 bagian CFG2CNF.py, dan pengujian.
Primanda Adyatma Hafiz (13520022)	Membuat parser dan FA pada lexer.py dan membuat CFG, mebuat laporan bab 1 bagian FA, bab 2, bab 3 bagian main.py dan lexer.py, dan pengujian.