

Laporan Tugas Besar
IF2124 Teori Bahasa Formal dan Otomata



Compiler Python

Disusun Oleh:

13520112 Fernaldy
13520150 Haidar Ihzaulhaq
13520163 Frederik Imanuel Louis

Sekolah Tinggi Elektro dan Informatika
Institut Teknologi Bandung
2021

Daftar Isi

Daftar Isi	2
Bab I Teori Dasar	3
1.1 Finite Automata.....	3
1.2 Context Free Grammar.....	3
1.3 Syntax Python	3
Bab II Hasil Finite Automata dan Context-Free Grammar.....	5
1.1 Finite Automata.....	5
1.2 Context Free Grammar.....	5
BAB III Implementasi dan Pengujian.....	9
Lampiran	19

Bab I

Teori Dasar

1.1 Finite Automata

Finite Automata adalah sebuah model mesin yang sering dipakai untuk menggambarkan kerja mesin yang ada saat ini. Bahasa yang diterima oleh FA disebut bahasa reguler. Kerja FA adalah dengan menerima input simbol-simbol lalu berpindah dari suatu state ke state lainnya. FA dapat dibagi menjadi dua yaitu Deterministic Finite Automata (DFA) dan Nondeterministic Finite Automata (NFA). DFA adalah FA yang hanya mencapai satu state tertentu ketika menerima serangkaian input. Notasi yang dipakai untuk menggambarkan DFA adalah $A = (Q, \Sigma, \delta, q_0, F)$ dengan Q adalah himpunan terhingga states, Σ adalah himpunan terhingga simbol, δ adalah fungsi transisi, q_0 adalah state awal, dan F adalah state final. Berbeda dari DFA yang hanya mampu mencapai satu state, NFA adalah FA yang mampu mencapai beberapa state ketika menerima serangkaian input. Notasi yang dipakai untuk menggambarkan suatu NFA sama seperti DFA yaitu $A = (Q, \Sigma, \delta, q_0, F)$. Perbedaannya terletak pada fungsi transisinya yaitu fungsi transisi pada DFA memetakan pasangan state dan simbol ke tepat satu state, sedangkan fungsi transisi pada NFA memetakan pasangan state dan simbol ke nol atau lebih state.

1.2 Context Free Grammar

Context Free Grammar (CFG) berisi serangkaian aturan produksi yang mendefinisikan bahasa yang bisa diterima. Bahasa yang diterima CFG disebut Context Free Language (CFL). Komponen yang mendeskripsikan suatu CFG adalah $G = (V, T, P, S)$ dengan V adalah himpunan terhingga variabel, T adalah himpunan terhingga terminal, P adalah himpunan terhingga aturan produksi, dan S adalah simbol awal. Kerja CFG adalah dengan menurunkan variabel-variabel mulai dari simbol awal mengikuti aturan produksi hingga memperoleh string yang diterima oleh CFG.

1.3 Syntax Python

Bahasa pemrograman Python adalah bahasa interpreted, berarti bahasa dieksekusi oleh interpreter Python. Penulisan bahasa Python bersifat case sensitive, dimana penggunaan huruf kapital dan huruf kecil berpengaruh pada keterbacaan kode. Bahasa Python juga sangat memperhatikan indentasi. Blok Kode di Python hanya ditentukan berdasarkan indentasinya, sehingga kode dengan indentasi yang tidak tepat akan memiliki control flow yang salah, atau bahkan tidak dapat dibaca interpreter. Tipe data variabel di Python juga termasuk loosely-typed, yang berarti tipe data tiap variabel akan ditentukan saat operasi dilakukan pada variabel tersebut (sesuai kebutuhan). Beberapa operator aritmetik di Python yaitu + (tambah), - (kurang), * (kali), / (bagi), % (modulus), // (floor), ** (pangkat), & (bitwise and), | (bitwise or), ^ (bitwise xor), >> (binary shift right), dan << (binary shift left). Kemudian, assignment di Python dilakukan dengan =. Beberapa operator

assignment yaitu +=, -=, /=, *=, %=, //=, **=, &=, |=, ^=, >>=, dan <<=. Kemudian, beberapa operasi perbandingan di Python antara lain == (sama dengan), != (tidak sama dengan), > (lebih besar dari), < (lebih kecil dari), >= (lebih besar dari atau sama dengan), dan <= (lebih kecil dari atau sama dengan). Beberapa operator logika di Python adalah and (kedua statement benar), or (salah satu statement benar), dan not (kebalikan dari statement). Selain itu, beberapa reserved words di Python adalah sebagai berikut:

Tabel 1. Reserved Words di Bahasa Python

Keywords in Python programming language				
False	await	else	import	pass
None	break	except	in	raise
True	class	finally	is	return
and	continue	for	lambda	try
as	def	from	nonlocal	while
assert	del	global	not	with
async	elif	if	or	yield

Bab II

Hasil Finite Automata dan Context-Free Grammar

1.1 Finite Automata

Finite Automata digunakan untuk mengecek nama variabel yang valid. Nama variabel harus dimulai dengan huruf alfabet (kapital atau kecil) atau dengan underscore (_). Kemudian, karakter selanjutnya haruslah huruf alfabet (kapital atau kecil), underscore, atau angka. Finite Automata tersebut didefinisikan dengan:

- $FA = \{ Q, C, T, S, Q_f \}$
- $Q = \{ S, F \}$
- $C = \{ a, b, \dots, z, A, B, \dots, Z, _, 0, 1, 2, \dots, 9 \}$
- $Q_f = \{ F \}$
- $T = \{ (S, \{ a, b, \dots, z, A, B, \dots, Z, _ \}, F), (F, \{ a, b, \dots, z, A, B, \dots, Z, _, 0, 1, 2, \dots, 9 \}, F) \}$

Tabel transisi dapat lebih jelas dilihat sebagai berikut:

Tabel 2. Tabel Transisi Finite Automata

	$A - Z$	$a - z$	$0 - 9$	$_$
$\rightarrow S$	F	F	\emptyset	\bar{F}
$* F$	F	F	F	F

1.2 Context Free Grammar

Pada pembuatan Context Free Grammar, semua transisi CFG sudah diubah menjadi Chomsky Normal Form untuk mempermudah proses coding. Berikut hasil pembuatan CNF:

1. Ekspresi

```
// Ekspresi, E1E2 -> x=y=z=literal
E -> E1E2 | E37E38 | E40E38 | VE19 | E44E45
// X=Y=Z= (E1)
E1-> E1E1 | VE3
E3-> =
// Literal (E2) E2E4-> binary, E6E7-> () E9E10->[a,b,c] E9E13-> list c. E18E2-> unary VE19-
> functions E6E20-> tuples E23E24-> sets E23E28-> dicts E31E2-> double ops
E2-> E2E4 | E6E7 | E9E10 | E9E13 | E18E2 | VE19 | E6E20 | E23E24 | E23E28 | E31E2 | V1V2
| V10V11 (str) | V14V15 (char) | V4V5 | integer | V3V7 (float) | VV17 (array) | True | False |
None | A-Z | a-z | _
E4-> E5E2
E5-> + | - | * | / | and | or | is | in | > | < | % | & | | ^//binary operators
E6-> (
E7-> E2E8
```

```

E8-> )
E9-> [
E10-> E2E11 | ]
E11-> E12E25 | ]
E12-> ,
E25-> E2E11 | ]
E13-> E2E14
E14-> EF E15
E15-> VE16
E16-> E17E35
E35->E2E36
E36-> ]
E17-> in
E18-> Not | + | - | ~ //unary operators
E19-> E6E20
E20-> E2E21 | )
E21-> E12E22 | )
E22-> E2E21 | )
E23-> {
E24-> E2E26 | }
E26-> E12E27 | }
E27-> E2E26 | }
E28-> E2E29 | }
E29-> E12E30 | }
E30-> E2E29 | }
EF-> for
// **, ==, >=, <=, //, <<, >> handling
E31-> E2E32
E32-> E33E33 | E34E3
E33-> * | = | / | < | >
E34-> > | <
E37-> VE39
E38-> E3E2
E39-> binary
E40-> VE41
E41-> E42E42
E42-> * | / | < | >
//Method/structs
E43-> E44E45
E44-> E44E44 | VE46
E45-> V1V2 | VV17 | A-Z | a-z | _ | VE19
E46-> .

```

2. Raise

R-> raiseVE19

3. Comments

C -> #{anything}

4. Tipe Data

```
// Integers
V3 -> {integer} | {nonzero}V4
V4-> V4V4 | integer

// Float
V6-> V3.V4

//Str
V9-> “V10” | “”
V10-> {anything} | V10V10

//Chr
V13-> ‘V14’ | ‘’
V14-> {anything} | V14V14
```

5. If

```
I -> ifE2: | IC
I1 -> elifE2: | I1C
I2 -> else: | I2C
I3 -> E | pass | I3C
I4 -> E | pass | I4C | O # if block inside def
I5 -> E | pass | break | continue | I5C # if block inside loop
```

6. For

```
F -> forVinE2: | FC
F1 -> E | pass | break | continue | W1C
F2 -> E | pass | break | continue | W1C | O # for block inside def
```

7. While

W -> whileE2: | WC

W1 -> E pass break continue W1C
W2 -> E pass break continue W2C O # while block inside def

8. With

M -> withVE19asV: withVE43asV: MC

9. Import

L -> fromVimportVasV importVasV importV fromVimportV
--

10. Def

D -> defV(D1): DC
D1 -> {blank} VD2 V
D2 -> , ,V D2D2

11. Return

O -> return returnE2 OC

12. Class

P -> classV: PC

13. Array

V16-> V2V17
V17-> E9V18
V18-> E2E36

BAB III

Implementasi dan Pengujian

1.1 Spesifikasi Teknis Program

Dalam program ini, kami menggunakan beberapa fungsi dan prosedur untuk membuat interpreter bahasa Python ini, berikut adalah fungsi dan prosedur yang digunakan:

1. `main()`

`main()` merupakan fungsi program utama tempat berjalannya program. Pada `main()`, akan dipanggil fungsi `blockParse(w)` untuk menjalankan pengecekan. Apabila `blockParse(w)` mengembalikan nilai `True`, maka artinya kode yang dicek tidak memiliki kesalahan, sedangkan bila mengembalikan nilai `false`, maka akan ditunjukkan letak kesalahannya. Fungsi `main()` juga memanggil fungsi `readFile(filename)` untuk membaca file yang akan dicek.

2. `blockParse(w)`

Fungsi ini digunakan untuk membaca tiap baris kode dan mengecek kebenarannya dengan memanggil fungsi `exprParse(w)`. Baris kode yang dibaca juga terbagi menjadi apakah dia merupakan sebuah block atau bukan, jika sebuah block, maka akan diperiksa sesuai dengan blocknya, jika tidak, maka akan dicek tiap barisnya. Kemudian `blockParse(w)` akan mengembalikan nilai `True` jika semua kode tertulis dengan benar dan `False` jika ada penulisan yang salah dan akan menyimpan dimana letak kesalahannya.

3. `readFile(filename)`

Fungsi `readFile()` adalah fungsi yang digunakan untuk membaca file yang akan diperiksa.

4. `exprParse(w)`

Fungsi ini digunakan untuk memeriksa tiap baris kode yang diberikan. Algoritma yang digunakan pada fungsi ini adalah CYK-Algorithm yang diimplementasikan ke dalam bahasa Python.

5. `lineToList(s)`

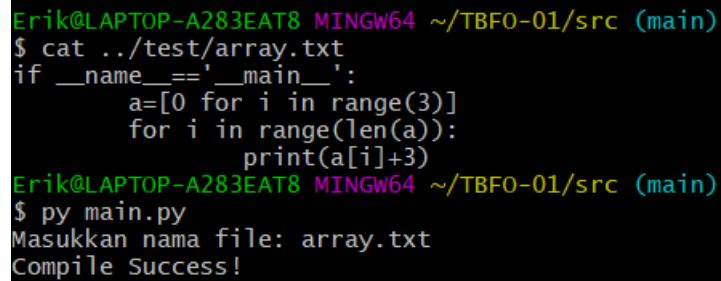
Fungsi ini mengembalikan sebuah list yang berisi kata-kata pada satu baris kode yang diberikan. Kata-kata pada sebuah baris kode akan dipisah sesuai dengan ketentuan, ada yang masuk dalam kategori `reserved`, `special`, atau yang lain sehingga terbentuklah sebuah list yang berisi kata-kata dalam satu baris untuk kemudian dicek kebenarannya.

Selain menggunakan fungsi dan prosedur, pada program kami juga menggunakan beberapa struktur data list yang digunakan, yaitu:

1. Reserved
Reserved list berisi list yang menjadi penanda awal untuk suatu line yang akan diperiksa dengan CYK-Algorithm. List ini berisi kata-kata seperti ["for", "True", "False", "in", "and", "or", "is", "Not", "raise", "while", "pass", "break", "continue", "if", "elif", "else", "import", "from", "as", "return", "class", "def", "with"].
2. Special
Special list berisi list yang akan diabaikan dalam pengubahan baris ke dalam list. List ini berisi ["\n", " ", "\r", "\t"].
3. Uppercase
Uppercase list berisi list huruf kapital dari A sampai Z.
4. Lowercase
Lowercase list berisi list huruf biasa dari a sampai z.
5. Integer
Integer list berisi list integer dari 0 sampai 9.
6. Nonzero
Nonzero list berisi list integer dari 1 sampai 9.
7. Binary
Binary list berisi list operasi yang berlaku untuk binary.
8. Boolop
Boolop list berisi operasi yang berlaku untuk boolean.
9. Unary
Unary list berisi operasi yang berlaku untuk 1 boolean.
10. Boolean
Boolean list berisi tipe tipe boolean, yaitu True, False, dan None.
11. Rules
Rules berisi rules grammar yang digunakan dalam CYK-Algorithm.
12. Lang
Lang berisi rules yang digunakan dalam blockParse.

1.2 Screenshot Pengujian

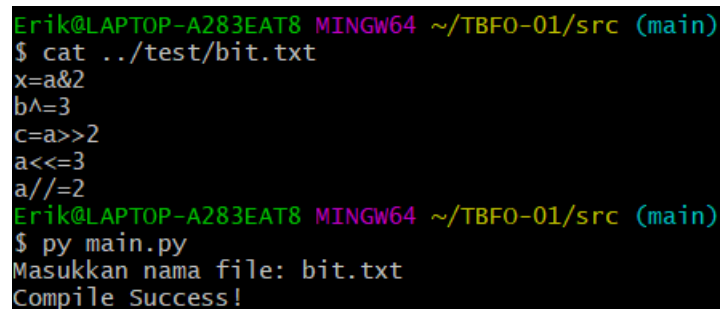
1. Pemrosesan Array

A terminal window with a black background and green text. The prompt is 'Erik@LAPTOP-A283EAT8 MINGW64 ~/TBFO-01/src (main)'. The user enters '\$ cat ../test/array.txt'. The output shows a Python script: 'if __name__=='__main__':', ' a=[0 for i in range(3)]', ' for i in range(len(a)):', ' print(a[i]+3)'. The user then enters '\$ py main.py'. The output is 'Masukkan nama file: array.txt' followed by 'Compile Success!' on the next line.

```
Erik@LAPTOP-A283EAT8 MINGW64 ~/TBFO-01/src (main)
$ cat ../test/array.txt
if __name__=='__main__':
    a=[0 for i in range(3)]
    for i in range(len(a)):
        print(a[i]+3)
Erik@LAPTOP-A283EAT8 MINGW64 ~/TBFO-01/src (main)
$ py main.py
Masukkan nama file: array.txt
Compile Success!
```

Gambar 1. Pengujian Pemrosesan Array

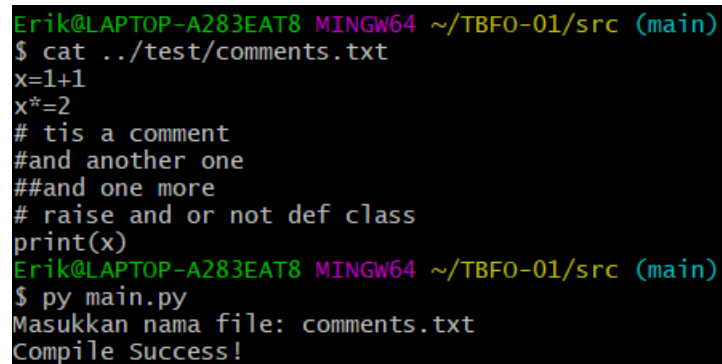
2. Operasi Bitwise

A terminal window with a black background and green text. The prompt is 'Erik@LAPTOP-A283EAT8 MINGW64 ~/TBFO-01/src (main)'. The user enters '\$ cat ../test/bit.txt'. The output shows a Python script: 'x=a&2', 'b^=3', 'c=a>>2', 'a<<=3', 'a/=2'. The user then enters '\$ py main.py'. The output is 'Masukkan nama file: bit.txt' followed by 'Compile Success!' on the next line.

```
Erik@LAPTOP-A283EAT8 MINGW64 ~/TBFO-01/src (main)
$ cat ../test/bit.txt
x=a&2
b^=3
c=a>>2
a<<=3
a/=2
Erik@LAPTOP-A283EAT8 MINGW64 ~/TBFO-01/src (main)
$ py main.py
Masukkan nama file: bit.txt
Compile Success!
```

Gambar 2. Pengujian Operasi Bitwise

3. Comments

A terminal window with a black background and green text. The prompt is 'Erik@LAPTOP-A283EAT8 MINGW64 ~/TBFO-01/src (main)'. The user enters '\$ cat ../test/comments.txt'. The output shows a Python script: 'x=1+1', 'x*=2', '# tis a comment', '#and another one', '##and one more', '# raise and or not def class', 'print(x)'. The user then enters '\$ py main.py'. The output is 'Masukkan nama file: comments.txt' followed by 'Compile Success!' on the next line.

```
Erik@LAPTOP-A283EAT8 MINGW64 ~/TBFO-01/src (main)
$ cat ../test/comments.txt
x=1+1
x*=2
# tis a comment
#and another one
##and one more
# raise and or not def class
print(x)
Erik@LAPTOP-A283EAT8 MINGW64 ~/TBFO-01/src (main)
$ py main.py
Masukkan nama file: comments.txt
Compile Success!
```

Gambar 3. Pengujian Comments

4. Error Message

```
Erik@LAPTOP-A283EAT8 MINGW64 ~/TBFO-01/src (main)
$ cat ../test/error.txt
x=1
x+=1
a=x+3
if 1==2:
    print a+x
a+=x
Erik@LAPTOP-A283EAT8 MINGW64 ~/TBFO-01/src (main)
$ py main.py
Masukkan nama file: error.txt
Error in line 5!
Error in :      print a+x
```

Gambar 4. Pengujian Error Message

5. Penamaan Variabel

```
Erik@LAPTOP-A283EAT8 MINGW64 ~/TBFO-01/src (main)
$ cat ../test/var_1.txt
x=y+z
__x=y2+t3
abcd2=dd+as
Erik@LAPTOP-A283EAT8 MINGW64 ~/TBFO-01/src (main)
$ py main.py
Masukkan nama file: var_1.txt
Compile Success!
```

Gambar 5. Pengujian Penamaan Variabel 1

```
Erik@LAPTOP-A283EAT8 MINGW64 ~/TBFO-01/src (main)
$ cat ../test/var_2.txt
aa=23+x0
2sd=a9o+x
Erik@LAPTOP-A283EAT8 MINGW64 ~/TBFO-01/src (main)
$ py main.py
Masukkan nama file: var_2.txt
Error in line 2!
Error in : 2sd=a9o+x
```

Gambar 6. Pengujian Penamaan Variabel 2

```
Erik@LAPTOP-A283EAT8 MINGW64 ~/TBFO-01/src (main)
$ cat ../test/var_3.txt
a[5] = a[2]+3
_a[3] = _b2[3][4]+4
a23[x][y][z+2*3] = True
Erik@LAPTOP-A283EAT8 MINGW64 ~/TBFO-01/src (main)
$ py main.py
Masukkan nama file: var_3.txt
Compile Success!
```

Gambar 7. Pengujian Penamaan Variabel 3

6. Ekspresi Aritmetik

```
Erik@LAPTOP-A283EAT8 MINGW64 ~/TBFO-01/src (main)
$ cat ../test/expr_1.txt
x = 3+ (2**3 +(x2+ 23 in [i for i in range(20)]) + True - - 1) *xyz
Erik@LAPTOP-A283EAT8 MINGW64 ~/TBFO-01/src (main)
$ py main.py
Masukkan nama file: expr_1.txt
Compile Success!
```

Gambar 8. Pengujian Ekspresi Aritmetik 1

```
Erik@LAPTOP-A283EAT8 MINGW64 ~/TBFO-01/src (main)
$ cat ../test/expr_2.txt
x=y=z((((y))))+(2**(3+5112332)//10)+2^3&7|x[y][w][j][k][l]
Erik@LAPTOP-A283EAT8 MINGW64 ~/TBFO-01/src (main)
$ py main.py
Masukkan nama file: expr_2.txt
Compile Success!
```

Gambar 9. Pengujian Ekspresi Aritmetik 2

7. If

```
if (100000 % 2 == 0):
    print("%")
else:
    print("!")
```

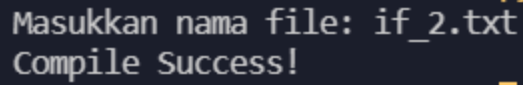
Gambar 10. Pengujian If 1

```
Masukkan nama file: if_1.txt
Compile Success!
```

Gambar 11. Hasil Pengujian If 1

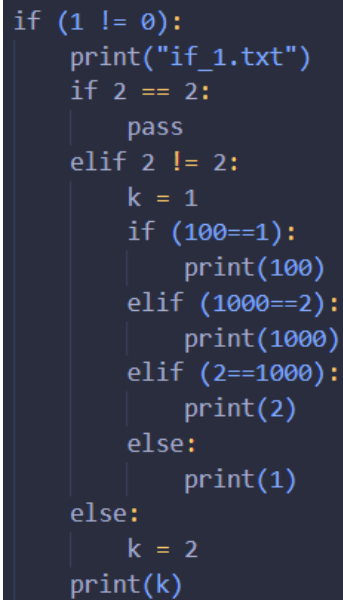
```
if 1==1:
    print(1)
if 1!=2:
    print(2)
elif 1>=2:
    print(3)
elif 1<=2:
    print(4)
elif 3!=3:
    print(5)
else:
    print(6)
```

Gambar 12. Pengujian If 2



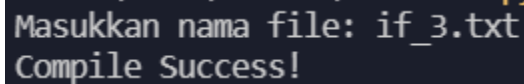
```
Masukkan nama file: if_2.txt
Compile Success!
```

Gambar 13. Hasil Pengujian If 2



```
if (1 != 0):
    print("if_1.txt")
    if 2 == 2:
        pass
    elif 2 != 2:
        k = 1
        if (100==1):
            print(100)
        elif (1000==2):
            print(1000)
        elif (2==1000):
            print(2)
        else:
            print(1)
    else:
        k = 2
print(k)
```

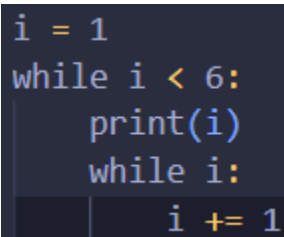
Gambar 14. Pengujian If 3



```
Masukkan nama file: if_3.txt
Compile Success!
```

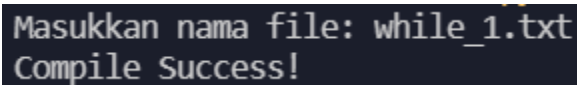
Gambar 15. Hasil Pengujian If 3

8. While



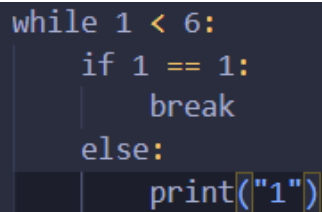
```
i = 1
while i < 6:
    print(i)
    while i:
        i += 1
```

Gambar 16. Pengujian While 1



```
Masukkan nama file: while_1.txt
Compile Success!
```

Gambar 17. Hasil Pengujian While 1



```
while 1 < 6:
    if 1 == 1:
        break
    else:
        print("1")
```

Gambar 18. Pengujian While 2

```
Masukkan nama file: while_2.txt
Compile Success!
```

Gambar 19. Hasil Pengujian While 2

```
while (1==1 and 2>1 and 1<2 and 1!=2 or not(1!=1)):
    while (1==1 and 2>1 and 1<2 and 1!=2 or not(1!=1)):
        while (1==1 and 2>1 and 1<2 and 1!=2 or not(1!=1)):
            while (1==1 and 2>1 and 1<2 and 1!=2 or not(1!=1)):
                while (1==1 and 2>1 and 1<2 and 1!=2 or not(1!=1)):
                    while (1==1 and 2>1 and 1<2 and 1!=2 or not(1!=1)):
                        while (1==1 and 2>1 and 1<2 and 1!=2 or not(1!=1)):
                            print("Hello")
                            continue
                            while (1==1 and 2>1 and 1<2 and 1!=2 or not(1!=1)):
                                break
```

Gambar 20. Pengujian While 3

```
Masukkan nama file: while_3.txt
Compile Success!
```

Gambar 21. Hasil Pengujian While 3

9. For

```
for a in range(len(m)):
    for b in m:
        d = 1
        d = 2
        d = 3
        d = 4
        d = 5
```

Gambar 22. Pengujian For 1

```
Masukkan nama file: for_1.txt
Compile Success!
```

Gambar 23. Hasil Pengujian For 1

```
for a in b:
    while True:
        if not(False):
            break
```

Gambar 24. Pengujian For 2

```
Masukkan nama file: for_2.txt
Compile Success!
```

Gambar 25. Hasil Pengujian For 2

```

for i in a:
    print(i)
    for j in "forloop":
        a = []
        for k in range(10000):
            print(a)
            for l in a:
                if l == "":
                    break
                else:
                    continue
            for x in range(1, 10000, 11):
                print(x+a[0]+55)

```

Gambar 26. Pengujian For 3

```

Masukkan nama file: for_3.txt
Compile Success!

```

Gambar 27. Hasil Pengujian For 3

10. With

```

PS D:\Kuliah\Semester 3\TBFO-01\src> cat ../test/with_1.txt
with open('file_path', 'w') as file:
    file.write('hello world !')
PS D:\Kuliah\Semester 3\TBFO-01\src> py main.py
Masukkan nama file: with_1.txt
Compile Success!

```

Gambar 28. Pengujian with 1

```

PS D:\Kuliah\Semester 3\TBFO-01\src> cat ../test/with_2.txt
with open('file_path', 'w') as file:
    file.write('hello world !')
    x = x+1
    if (x != 1) :
        x = 2
PS D:\Kuliah\Semester 3\TBFO-01\src> py main.py
Masukkan nama file: with_2.txt
Compile Success!

```

Gambar 29. Pengujian with 2

```

PS D:\Kuliah\Semester 3\TBFO-01\src> cat ../test/with_3.txt
message_writer = MessageWriter('hello.txt')
with message_writer.open_file() as my_file:
    my_file.write('hello world')
PS D:\Kuliah\Semester 3\TBFO-01\src> py main.py
Masukkan nama file: with_3.txt
Compile Success!

```

Gambar 30. Pengujian with 3

11. Import/As/From

```
PS D:\Kuliah\Semester 3\TBFO-01\src> cat ../test/import_1.txt
import math as myAlias
PS D:\Kuliah\Semester 3\TBFO-01\src> py main.py
Masukkan nama file: import_1.txt
Compile Success!
```

Gambar 31. Pengujian Import/As/From 1

```
PS D:\Kuliah\Semester 3\TBFO-01\src> cat ../test/import_2.txt
from math import cos as cs
PS D:\Kuliah\Semester 3\TBFO-01\src> py main.py
Masukkan nama file: import_2.txt
Compile Success!
```

Gambar 32. Pengujian Import/As/From 2

```
PS D:\Kuliah\Semester 3\TBFO-01\src> cat ../test/import_3.txt
from math import cos
PS D:\Kuliah\Semester 3\TBFO-01\src> py main.py
Masukkan nama file: import_3.txt
Compile Success!
```

Gambar 33. Pengujian Import/As/From 3

12. Def/Return

```
PS D:\Kuliah\Semester 3\TBFO-01\src> cat ../test/def_1.txt
def a(x) :
    x += 1
    return x+1
PS D:\Kuliah\Semester 3\TBFO-01\src> py main.py
Masukkan nama file: def_1.txt
Compile Success!
```

Gambar 34. Pengujian Def/Return 1

```
PS D:\Kuliah\Semester 3\TBFO-01\src> cat ../test/def_2.txt
def a(x) :
    x += 1
    def tes(m,n) :
        hasil = m+n
        return hasil
    return x+1
PS D:\Kuliah\Semester 3\TBFO-01\src> py main.py
Masukkan nama file: def_2.txt
Compile Success!
```

Gambar 35. Pengujian Def/Return 2

```
PS D:\Kuliah\Semester 3\TBFO-01\src> cat ../test/def_3.txt
def swap(a,b) :
    temp = a
    a = b
    b = temp
    if (a == 1) :
        return b
    else :
        return a
PS D:\Kuliah\Semester 3\TBFO-01\src> py main.py
Masukkan nama file: def_3.txt
Compile Success!
```

Gambar 36. Pengujian Def/Return 3

13. Class

```
PS D:\Kuliah\Semester 3\TBFO-01\src> cat ../test/class_1.txt
class Test :
    tes = 1
PS D:\Kuliah\Semester 3\TBFO-01\src> py main.py
Masukkan nama file: class_1.txt
Compile Success!
```

Gambar 37. Pengujian class 1

```
PS D:\Kuliah\Semester 3\TBFO-01\src> cat ../test/class_2.txt
class Name :
    def keep(a) :
        a = a-1
        if (a == 3) :
            return a
    def throw(a) :
        a = a-1
        if (a == 1) :
            return 4
PS D:\Kuliah\Semester 3\TBFO-01\src> py main.py
Masukkan nama file: class_2.txt
Compile Success!
```

Gambar 38. Pengujian class 2

```
PS D:\Kuliah\Semester 3\TBFO-01\src> cat ../test/class_3.txt
class Employee:
    #Common base class for all employees
    empCount = 0

    def __init__(self, name, salary):
        self.name = name
        self.salary = salary
        Employee.empCount += 1

    def displayCount(self):
        print("Total Employee ",Employee.empCount)

    def displayEmployee(self):
        print("Name : ", self.name, ", salary: ", self.salary)
PS D:\Kuliah\Semester 3\TBFO-01\src> py main.py
Masukkan nama file: class_3.txt
Compile Success!
```

Gambar 39. Pengujian class 3

Lampiran

Repository Github:

<https://github.com/dxt99/TBFO-01>

Pembagian Tugas:

NIM	Nama	Pembagian Tugas
13520112	Fernaldy	CFG dan CNF untuk If, For, dan While Implementasi If, For, dan While
13520150	Haidar Ihzaulhaq	CFG dan CNF untuk with, import from as, def return, class. Implementasi with, import from as, def return, class.
13520163	Frederik Imanuel Louis	CFG dan CNF Expression, Variables, dan Raise, FA Variables Implementasi Expression, Variables, dan Raise