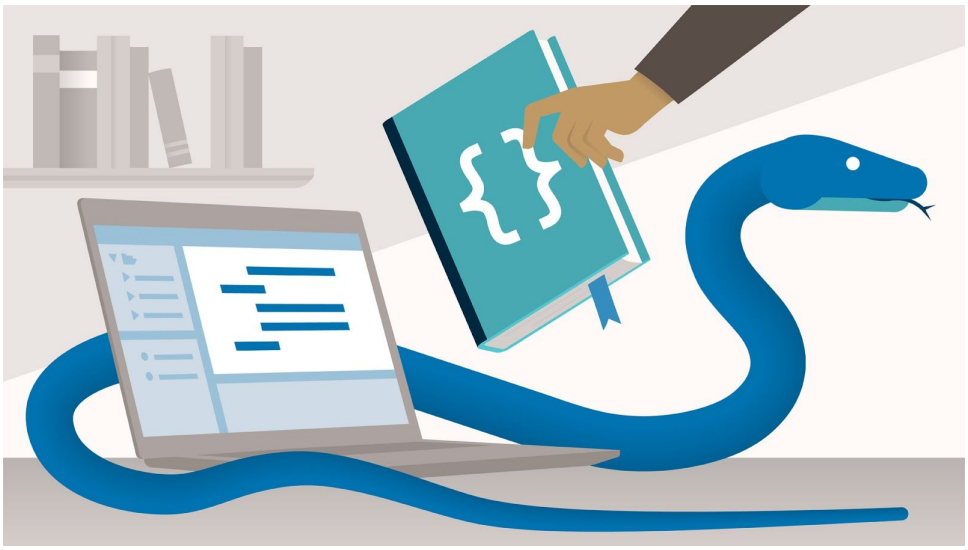
Tugas Besar 2 IF 2124 Teori Bahasa Formal dan Otomata
Compiler Bahasa Python

Tanggal Rilis : **Rabu, 13 November 2019**
Tanggal Pengumpulan : **Rabu, 27 November 2019 pukul 11:59**



Sumber: https://cdn.lynda.com/course/786416/786416-636886711478372555-16x9.jpg

Deskripsi Permasalahan

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*.

Pada tugas besar ini, implementasikanlah ***compiler* untuk Python** untuk *statement-statement* dan sintaks-sintaks bawaan Python dengan menggunakan algoritma **CYK (Cocke-Younger-Kasami)**. Algoritma CYK harus menggunakan *grammar* CNF (Chomsky Normal Form) sebagai *grammar* masukannya. Oleh karena itu, 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	

Hal-hal ini tidak perlu kalian masukkan ke dalam *grammar* atau diimplementasikan:

1. Arti semantik dari *input* (mis. walaupun kelas `Foo` belum pernah didefinisikan, `bar = Foo()` atau `bar.a_method()` diperbolehkan)
2. Arti semantik dari *method* (mis. jumlah parameternya)
3. Regex dalam bentuk apapun, seperti r-string (mis. `r'123'`)
4. F-string (mis. `f'aku suka tbfo'`)
5. *End of statement* dengan menggunakan titik koma (;)
6. *Syntactic sugar*
7. Decorator (@), dan *backslash* (\)
8. Karakter-karakter di luar cakupan ASCII.
9. Tab(INDENT/DEDENT)

Input dan Output

1. Teks input dibaca melalui sebuah file eksternal, dengan nama bebas, dijadikan parameter eksekusi program
2. Keluaran berupa **"Accepted"**, atau **"Syntax Error"**.
3. **(BONUS)** Jika ditolak, tampilkan sebuah output (bebas) yang memudahkan kalian untuk mencari penyebab kesalahan. (5)
4. **(BONUS)** Menerima *type hinting*. (5)

Berikut adalah contoh input-output yang mungkin:

input1.txt (*highlight* biru adalah *type hinting*)

```
def get_rule_category(rule: dict) -> str:
    ''' Get rule category in string. This category is also a key for its corresponding dictionary.
        Input(dict) = rule
        Output(str) = category of the rule

        Example:
        Input   = {'producer': 'N', 'product': ['people']}
        Output  = 'terminal'
    '''
```

```
    rule_product = rule[PRODUCT_KEY]
    if len(rule_product) == 0:
        return EPSILON_RULE_KEY
    elif len(rule_product) == 1:
        if rule_product[0].islower:
            return TERMINAL_RULE_KEY
        else:
            return UNARY_RULE_KEY
    elif len(rule_product) == 2:
        return BINARY_RULE_KEY
    else:
        return N_ARIES_RULE_KEY
```

input2.txt

```
from PIL import Image
import pytesseract
import time
import keyboard
import random

IMAGE_INPUT_NAME = 'input.jpg'
FILE_OUTPUT_NAME = 'result.txt'
ACCEPTED_ASCII_LIST = [x for x in range(32, 127)]

def extract_text_from_image():
    PIPE_ASCII = 124

    time_start = time.perf_counter()
    text = pytesseract.image_to_string(Image.open(IMAGE_INPUT_NAME))
    with open(FILE_OUTPUT_NAME, 'w') as f_out:
        for c in text:
            ascii = ord(c)
            if ascii == PIPE_ASCII:
                f_out.write('I')
            elif ascii in ACCEPTED_ASCII_LIST:
                f_out.write(c)
            else:
                f_out.write(' ')
    time_end = time.perf_counter()
    print('\nDone! Executed in: {:.2f} seconds'.format(time_end - time_start))
```

input3.txt (*highlight* kuning adalah lokasi-lokasi kesalahan)

```
from PIL importasdf Image

-var = 'input.jpg”
ACCEPTED_ASCII_LIST = [x for x in range((32, 127)}

def extract_text_from_image()::
```

```
    PIPE_ASCII = 124

time_start = time.perf_counter()
    text = pytesseract.image_to_string(Image.open(IMAGE_INPUT_NAME))
    with open(FILE_OUTPUT_NAME, 'asdf') as f_out:
        for c in text:
            ascii = ord(c)
            if ascii == PIPE_ASCII:
                f_out.write('I')
            elif ascii in ACCEPTED_ASCII_LIST:
                f_out.write[c]
            else ascii == 1:
                f_out.write(' ')
    time_end = time.perf_counter()
    print('\nDone! Executed in: {:.2f} seconds'.format(time_end - time_start))
```

Contoh 1

main.exe input1.txt

Accepted

Penjelasan:
Terdapat *type hinting* yang adalah bonus

Contoh 2

main.exe input2.txt

Accepted

Contoh 3

main.exe input3.txt

Syntax Error

Penjelasan:
Terdapat kesalahan pada lokasi-lokasi yang telah ditandai.

Lain-lain:

1. Versi Python yang dijadikan referensi adalah **Python 3.7**
2. Dalam pengerjaan tugas ini, belajar dari sumber-sumber lain diperbolehkan, tetapi **persalinan atau kecurangan dalam bentuk apapun sangat dilarang. Pelaku kecurangan akan ditindak tegas**.
3. Tidak diperkenankan menggunakan *library* di luar *library* bawaan bahasa (misalkan yang diperoleh melalui *pip install*, atau *npm -i*). Termasuk *library-library* lain yang memberikan solusi instan untuk algoritma CYK, dan/atau konversi CFG ke CNF.
4. Bahasa yang digunakan **bebas**.

Tips:

1. Dianjurkan untuk menggunakan *script* untuk melakukan konversi CFG ke CNF.
2. Pastikan CFG sudah benar sebelum melakukan konversi ke CNF.
3. Gunakan sesedikit mungkin produksi epsilon untuk mempermudah proses konversi.
4. Ketika melakukan konversi CFG ke CNF, dianjurkan untuk melakukan *binarization* dahulu sebelum melakukan operasi konversi selanjutnya.
5. Gunakan tab (\t) bukan spasi untuk *whitespace*.
6. Jangan lupa *end of statement* cukup newline (\n) saja (titik koma {;} tidak perlu diimplementasikan)
7. Untuk mempermudah visualisasi CYK, dapat melihat referensi berikut.

QnA:

1. *Magic methods* dan *variables* (fungsi atau variabel apapun yang diawali dan diakhiri 2 *underscore* seperti **__init__, __del__, __name__, dll**) digimanain?
   Dianggap seperti method ato variabel biasa
2. Fungsi bawaan seperti len(), range(), digimanain?
   Dianggap seperti method biasa
3. Untuk QnA lebih lanjut dapat ditanyakan pada https://irklab.site/qnatubes2tbfo

Isi laporan:

1. Dasar teori tentang Python, CFG, CNF, dan CYK.
2. Analisis persoalan, berisi:
   a. Grammar CFG
   b. Grammar CNF
3. Implementasi dan pengujian. Bab ini berisi:
   a. Spesifikasi teknis program, termasuk di dalamnya struktur data, fungsi dan prosedur (*header* fungsi dan prosedur saja, tidak perlu *source code*), antarmuka, dan lain-lain yang dianggap perlu.
   b. *Capture* layar yang memperlihatkan contoh dari berbagai kasus yang muncul (**buatlah 5 kemungkinan tipe kasus uji yang mungkin**) dan analisis hasilnya.
4. Referensi

Spesifikasi Tugas

1. Program dibuat berkelompok dengan setiap kelompok terdiri dari **maksimal 3 orang dari kelas yang sama**. Silahkan mengisi *sheet* di pranala berikut: irklab.site/kelompoktubes2tbfo.
2. Terdapat 2 *deliverable* utama dalam tugas besar ini, yaitu laporan, dan program. Kedua *deliverable* tersebut dikumpulkan maksimal hari **Senin, 25 November 2019 pukul 23.59**. Untuk program silakan di-*zip* dengan format : **Tubes2TBFO_NIM1_NIM2_NIM3.zip** ke pranala yang akan diberikan asisten kemudian. Untuk laporan, silahkan dikumpulkan di lab IRK.
3. Segala bentuk pertanyaan dapat dikirimkan ke milis TBFO agar dapat dilihat oleh seluruh peserta mata kuliah TBFO.
4. Demo program akan dilakukan pada tanggal 26-29 November 2019. Mengenai teknis pelaksanaan demo, akan diberitahukan lebih lanjut oleh asisten.

Keterangan Laporan

1. Laporan tidak perlu memakai *cover* mika dan dijilid. Pastikan laporan tidak akan tercecer jika dibaca.

2. Laporan boleh menggunakan kertas daur ulang, boleh bolak-balik, boleh dalam satu halaman kertas terdapat dua halaman tulisan asalkan masih terbaca.

Penilaian

1. Kebenaran program (30%) : program mampu berjalan sesuai dengan spesifikasi yang diberikan.
2. Demo dan tes awal – pemahaman Anda dalam pembuatan program (20%)
3. Laporan (50%), terdiri dari:
    a. Dasar Teori(7.5%)
    b. Analisis Persoalan(25%)
    c. Spesifikasi Teknis Program(10%)
    d. *Capture* Kasus Uji(5%)
    e. Referensi(2.5%)