Всероссийское СМИ
«Академия педагогических идей «НОВАЦИЯ»
Свидетельство о регистрации ЭЛ №ФС 77-62011 от 05.06.2015 г.
(выдано Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций)

Сайт: akademnova.ru
e-mail: akademnova@mail.ru

Ахметшин Р.Р., Журавлев М.П. Использование языка Python при разработке парсера // Академия педагогических идей «Новация». – 2017. – № 02 (февраль). – АРТ 11-эл. – 0,2 п. л. – URL: http://akademnova.ru/page/875548

РУБРИКА: ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАНИЕ

УДК 004

Ахметшин Рамазан Рафикович
Магистр, 2 курс, факультет дизайна и программной инженерии
ФГБОУ ВО «Казанский Национальный
Исследовательский Технологический Университет»,
г. Казань, Республика Татарстан, Российская Федерация
e-mail: 19badr@gmail.com

Журавлёв Михаил Петрович
Магистр, 2 курс, факультет дизайна и программной инженерии
ФГБОУ ВО «Казанский Национальный
Исследовательский Технологический Университет»,
г. Казань, Республика Татарстан, Российская Федерация
e-mail: mihuna@yandex.ru

ИСПОЛЬЗОВАНИЕ ЯЗЫКА PYTHON ПРИ РАЗРАБОТКЕ ПАРСЕРА

Аннотация: В статье рассматривается применение языка Python для автоматизированного синтаксического анализа больших структурированных объёмов информации.

Ключевые слова: python, парсинг, программирование, синтаксический анализ, условные операторы, оптимизация кода, высокопроизводительные вычисления.

Всероссийское СМИ
«Академия педагогических идей «НОВАЦИЯ»
Свидетельство о регистрации ЭЛ №ФС 77-62011 от 05.06.2015 г.
(выдано Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций)

Сайт: akademnova.ru
e-mail: akademnova@mail.ru

Akhmetshin Ramazan Rafikovich
Master, 2nd year, Design and Software Engineering Faculty
FSEI HE «Kazan National Research Technological University»
Kazan, Republic of Tatarstan, Russian Federation
e-mail: 19badr@gmail.com

Zhuravlyev Mihail Petrovich
Master, 2nd year, Design and Software Engineering Faculty
FSEI HE «Kazan National Research Technological University»
Kazan, Republic of Tatarstan, Russian Federation
e-mail: mihuna@yandex.ru

USE OF THE PYTHON PROGRAMMING LANGUAGE IN THE PARSER DEVELOPMENT

Abstract: The article describes the using of Python for automated parsing large amounts of structured data.

Keywords: programming, parsing, conditional statements, code optimization, high-performance computing.

Парсинг (по-русски «синтаксический анализ») — задача, результатом решения которой является осмысленный разбор и преобразование в осмысленные единицы блока информации, описанного на некотором фиксированном языке, будь то язык программирования, язык разметки, язык структурированных запросов и т.п. Типичная последовательность этапов решения задачи выглядит примерно так:

1. **Описание языка**. При ясности получаемого результата устанавливаются как правила построения предложений в языке, так и правила определения валидных слов.

Всероссийское СМИ
«Академия педагогических идей «НОВАЦИЯ»
Свидетельство о регистрации ЭЛ №ФС 77-62011 от 05.06.2015 г.
(выдано Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций)
Сайт: akademnova.ru
e-mail: akademnova@mail.ru

2. **Разбиение ввода на токены**. Пишется лексический анализатор, который позволит разбить вводную строку (файл, блок информации и т.п.) на валидные слова.

3. **Построение синтаксического древа**. Этот этап определяет проверку работы, проведенной на предыдущем этапе. Обычно, эта задача решается с помощью метода рекурсивного спуска, с помощью которого строится валидное синтаксическое древо.

4. **Выполнение операции над информацией.** После построения синтаксического древа информация готова к компиляции, сборке, переводу, отображению и т.п.

Хоть эта область программирования описана внушительным количеством литературы, имеются некоторые проблемы на практике, которые идут вразрез с теорией.

Проиллюстрируем работу парсера на выводе JSON:

```
root ::= value
value ::= string | number | object | array | 'true' | 'false' | 'null'

array ::= '[' ']' | '[' comma-separated-values ']'
comma-separated-values ::= value | value ',' comma-separated-values

object ::= '{' '}' | '{' comma-separated-keyvalues '}'
comma-separated-keyvalues ::= keyvalue | keyvalue ',' comma-separated-keyvalues
keyvalue ::= string ':' value
```

Здесь нет правил для `string` и `number` — они, вместе со всеми строками в кавычках могут быть использованы, как токены.

Нашей задачей не является написание полноценного токенайзера, так как не стоит задача разбора конкретного блока информации, поэтому будем описывать только по мере необходимости:

Всероссийское СМИ
«Академия педагогических идей «НОВАЦИЯ»
Свидетельство о регистрации ЭЛ №ФС 77-62011 от 05.06.2015 г.
(выдано Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций)
Сайт: akademnova.ru
e-mail: akademnova@mail.ru

```
import re
number_regex = re.compile(r"(-?(?:0|[1-9]\d*)(?:\.\d+)?(?:[eE][+-]?\d+)?)\s*(.*)",
re.DOTALL)
def parse_number(src):
    match = number_regex.match(src)
    if match is not None:
        number, src = match.groups()
        return eval(number), src

string_regex = re.compile(r"('(?:[^\\']|\\['\\/bfnrt]|\\u[0-9a-fA-F]{4})*?')\s*(.*)",
re.DOTALL)
def parse_string(src):
    match = string_regex.match(src)
    if match is not None:
        string, src = match.groups()
        return eval(string), src
```

Для всего остального напишем одну функцию, которая создаст вспомогательные парсеры:

```
def parse_word(word, value=None):
    l = len(word)
    def result(src);
        if src.startswith(word):
            return value, src[l:].lstrip()
    result.__name__ = "parse_%s" % word  # для отладки
    return result

parse_true = parse_word("true", True)
parse_false = parse_word("false", False)
parse_null = parse_word("null", None)
```

Итого, по какому принципу мы строим наши функции:

1. Принятие строки.
2. Возвращение true/false.

Всероссийское СМИ
«Академия педагогических идей «НОВАЦИЯ»
Свидетельство о регистрации ЭЛ №ФС 77-62011 от 05.06.2015 г.
(выдано Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций)
Сайт: akademnova.ru
e-mail: akademnova@mail.ru

3. Удаление всех пробелов (парсится не python).

После построения токенайзера пишется, непосредственно, парсер.

Если заменить `return` на `yield!`, то функции будут возвращать генераторы — пустые, если парсинг не удался, и ровно с одним элементом, если удался. Другими словами, таким образом меняется весь принцип программирования и код теперь будет писаться в таком стиле:

```
number_regex = re.compile(r"(-?(?:0|[1-9]\d*)(?:\.\d+)?(?:[eE][+-]?\d+)?)\s*(.*)",
re.DOTALL)

def parse_number(src):
    match = number_regex.match(src)
    if match is not None:
        number, src = match.groups()
        yield eval(number), src
    # если управление дошло до сюда без yield, числа обнаружить не удалось.
    # ну что же, пустой генератор тоже генератор.

string_regex = re.compile(r"('(?:[^\\']|\\['\\/bfnrt]|\\u[0-9a-fA-F]{4})*?')\s*(.*)",
re.DOTALL)

def parse_string(src):
    match = string_regex.match(src)
    if match is not None:
        string, src = match.groups()
        yield eval(string), src

def parse_word(word, value=None):
    l = len(word)
    def result(src):
        if src.startswith(word):
            yield value, src[l:].rstrip()
    result.__name__ = "parse_%s" % word
    return result  # здесь возвращаем функцию-парсер, не yield'им
```

Всероссийское СМИ
«Академия педагогических идей «НОВАЦИЯ»
Свидетельство о регистрации ЭЛ №ФС 77-62011 от 05.06.2015 г.
(выдано Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций)
Сайт: akademnova.ru
e-mail: akademnova@mail.ru

```
parse_true = parse_word("true", True)
parse_false = parse_word("false", False)
parse_null = parse_word("null", None)
```

Таким образом, воплощается ситуация, при которой каждая опция может занимать лишь одну строку. С помощью функции `intertools` мы объединим код таким образом, что каждая функция сможет исполниться только после исполнения предыдущей функции:

```
from itertools import chain

def parse_value(src):
    for match in chain(
        parse_string(src),
        parse_number(src),
        parse_array(src),
        parse_object(src),
        parse_true(src),
        parse_false(src),
        parse_null(src),
    ):
        yield match
        return
```

При этом эффективность остаётся на прежнем уровне — каждая функция начнёт выполняться (проверять регулярное выражение) только тогда, когда предыдущая не даст результата. `Return` гарантирует, что лишняя работа не будет выполнена, если где-то в середине списка парсинг удался (был возвращен «true»)

В таком ключе, функция `parse_array` будет выглядеть так:

```
parse_left_square_bracket = parse_word("[")
```

Всероссийское СМИ
«Академия педагогических идей «НОВАЦИЯ»
Свидетельство о регистрации ЭЛ №ФС 77-62011 от 05.06.2015 г.
(выдано Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций)
Сайт: akademnova.ru
e-mail: akademnova@mail.ru

```
parse_right_square_bracket = parse_word("]")

def parse_array(src):
    # здесь потребуется временная переменная tsrc, иначе даже при неудачном
    # парсинге пустого массива открывающая скобка может быть "съедена"
    for _, tsrc in parse_left_square_bracket(src):
        for _, tsrc in parse_right_square_bracket(tsrc):
            # если мы здесь, то нам удалось последовательно распарсить '[' и ']'
            yield [], tsrc
            return
    for _, src in parse_left_square_bracket(src):
        for items, src in parse_comma_separated_values(src):
            for _, src in parse_right_square_bracket(src):
                yield items, src
    # если управление дошло до сюда без yield, то парсинг массива не удался
```

На данном этапе разрабатываемая функция описана без конструкции if-else, что, в определенной степени, является оптимизацией кода. Допишем рекурсивную функцию для полной автоматизации:

```
def sequence(*funcs):
    if len(funcs) == 0:
        def result(src):
            yield (), src
        return result
    def result(src):
        for arg1, src in funcs[0](src):
            for others, src in sequence(*funcs[1:])(src):
                yield (arg1,) + others, src  # конкатенируем содержательные результаты
    return result
```

С использованием этого кода функция перепишется в виде:

```
parse_left_square_bracket = parse_word("[")
parse_right_square_bracket = parse_word("]")
parse_empty_array = sequence(parse_left_square_bracket, parse_right_square_bracket)
```

Всероссийское СМИ
«Академия педагогических идей «НОВАЦИЯ»
Свидетельство о регистрации ЭЛ №ФС 77-62011 от 05.06.2015 г.
(выдано Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций)
Сайт: akademnova.ru
e-mail: akademnova@mail.ru

```
def parse_array(src):
    for _, src in parse_empty_array(src):  # увы, эта функция недостаточно умна, чтобы вернуть []
        yield [], src
        return  # этот return нужен, чтобы не пойти радостно парсить
                # дальше в конструкции вида {} {"a": 1}

    for (_, items, _), src in sequence(
        parse_left_square_bracket,
        parse_comma_separated_values,
        parse_right_square_bracket,
    )(src):
        yield items, src
    # если управление дошло сюда без yield, то парсинг массива не удался
```

Останется только дописать функцию `parse_comma_separated_values`, объект интерфейса и отправить код интерпретатор.

```
parse_comma = parse_word(",")

def parse_comma_separated_values(src):
    for (value, _, values), src in sequence(
        parse_value,
        parse_comma,
        parse_comma_separated_values  # я говорил, что не могу в рекурсию без if? я соврал
    )(src):
        yield [value] + values, src
        return

    for value, src in parse_value(src):
        yield [value], src
```

Стоит заметить, что такое решение не приведет к бесконечной

Всероссийское СМИ
«Академия педагогических идей «НОВАЦИЯ»
Свидетельство о регистрации ЭЛ №ФС 77-62011 от 05.06.2015 г.
(выдано Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций)
Сайт: akademnova.ru
e-mail: akademnova@mail.ru

рекурсии, так как `parse_comma` не найдёт очередной запятой, и до последующей `parse_comma_separated_values` выполнение уже не дойдёт.

Пример вычисления представлен ниже:

```
>>> import my_json
>>> my_json.parse("null")
>>> my_json.parse("true")
True
>>> my_json.parse("false")
False
>>> my_json.parse("0.31415926E1")
3.1415926
>>> my_json.parse("[1, true, '1']")
[1, True, '1']
>>> my_json.parse("{}")
{}
>>> my_json.parse("{'a': 1, 'b': null}")
{'a': 1, 'b': None}
```

В данной статье рассмотрены далеко не все возможные проблемы и пути решения возникающих перед программистом задач при парсинге, однако, представлен весьма элегантный способ решения рядовой задачи: токенайзер строился, по сути, в процессе решения поставленной задачи — написания парсера, а так же максимально рационально использованы вычислительные мощности и время работы программиста.

Список использованной литературы:

1. Доусон М. Программируем на Python. – СПб.: Питер, 2014. – 416 с.
2. Лутц М. Изучаем Python, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 1280 с.
3. Лутц М. Программирование на Python, том I, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 992 с.

Всероссийское СМИ
«Академия педагогических идей «НОВАЦИЯ»
Свидетельство о регистрации ЭЛ №ФС 77-62011 от 05.06.2015 г.
(выдано Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций)

Сайт: akademnova.ru
e-mail: akademnova@mail.ru

4. Лутц М. Программирование на Python, том II, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 992 с.
5. Прохоренок Н.А. Python 3 и PyQt. Разработка приложений. – СПб.: БХВ-Петербург, 2012. – 704 с.
6. Пилгрим Марк. Погружение в Python 3 (Dive into Python 3 на русском)
7. Прохоренок Н.А. Самое необходимое. — СПб.: БХВ-Петербург, 2011. — 416 с.
8. Хахаев И.А. Практикум по алгоритмизации и программированию на Python. – М.: Альт Линукс, 2010. — 126 с. (Библиотека ALT Linux).
9. Чаплыгин А.Н. Учимся программировать вместе с питоном.
10. Briggs J. R. — Python for Kids — 2012
11. Allen Downey – ThinkPython+Kart[Python_3.2]

Дата поступления в редакцию: 27.02.2017 г.
Опубликовано: 28.02.2017 г.