Университет ИТМО, факультет программной инженерии и компьютерной техники
Двухнедельная отчётная работа по «Информатике»: аннотация к статье
Дата прошедшей лекции: _25.10.2022_ Номер прошедшей лекции: _4_ Дата сдачи: _08.11.2022_

Выполнил(а) Коломиец Н. С., № группы *P3108*, оценка _____
Фамилия И.О. студента — не заполнять

Название статьи/главы книги/видеолекции
Генерация PEG-парсера

ФИО автора статьи (или e-mail)	**Дата публикации (не старше 2019 года)**	**Размер статьи (от 400 слов)**
TyVik	"21"октября 2019 г.	__1к+__

Прямая полная ссылка на источник или сокращённая ссылка (bit.ly, tr.im и т.п.)
https://habr.com/ru/post/471864/

Теги, ключевые слова или словосочетания

Python, Программирование, Алгоритмы

Перечень фактов, упомянутых в статье
1. Простой способ описать мета-грамматику — использовать только встроенные типы данных: правая часть правила — список списков элементов, каждый из которых может быть просто строкой
2. Для правил используется простой класс Rule, и вся грамматика представляет собой список таких объектов
3. item() возвращает строку, alternative() возвращает список строк, а переменная alts внутри rule() собирает список списков строк. Затем метод rule() объединяет имя правила (строку) и преобразует его в объект Rule.
4. Если применить этот алгоритм к нашей игрушечной грамматике, то метод grammar() вернёт список правил
5. Декоратор @memoize: введён, чтобы перейти к другой теме: использование мемоизации, чтобы сделать сгенерированный парсер достаточно быстрым.

Позитивные следствия и/или достоинства описанной в статье технологии (минимум три пункта)
1. Можно избавиться от вложенных словарей, используя (pos, func, args) в качестве ключа**.**
2. При последующих вызовах того же метода синтаксического анализа с теми же аргументами в той же позиции входных данных мы будем брать их из кэша. Для этого достаточно всего лишь перевести указатель на позицию ввода с помощью self.reset() и посмотреть в кэш.
3. Внутренние словари добавляются по мере необходимости; их ключи состоят из метода и его аргументов.

Негативные следствия и/или недостатки описанной в статье технологии (минимум три пункта)
1. Большинство вызовов будут негативными. В этом случае возвращаемое значение равно None, а позиция ввода не изменяется
2. В Python принято реализовывать кэш в локальной переменной в функции memoize(). В нашем случае так не получится: как я выяснил в самом конце отладки, каждый экземпляр Parser обязан иметь свой собственный кэш
3. Работает только на «игрушечной» грамматике