1F-03

pyparsing を用いたプログラミング言語の構文解析処理教育

黒瀬　浩†

金沢工業大学工学部†

1. はじめに

近年，Web スクレイピングやビックデータからのデータ抽出が盛んに行われている．

著者は，プログラミング言語ソースファイルの字句解析，構文解析処理を中心としたコンパイラの授業科目を汎用的なデータ解析も行えるように pyparsing を用いた教育を行った．学習者は，Python 言語と構造的なテキストデータ処理が学習でき，さまざまな構造の文字列解析への応用が可能となる．一方，コンパイラコンパイラ等のツールを用いないため，言語に特化した処理サンプルの不足や，コンパイラの詳細処理について学習者の理解度に差が生じる課題も存在した．

2. コンパイラ処理の学習

大学学部生に対する専門教育としてコンパイラの処理を学ぶ授業を行っている大学がある．多くの大学では yacc[1)]や ANTLR[2)]を用いてコンパイラ処理の教育を行っている．

近年，HTML，XML，ログファイル，組込み機器との通信などが盛んになり，汎用的な構文解析処理が求められてることから汎用的なパーサを学ぶことは応用分野の拡大が期待できる．筆者は，pyparsing[3)]を用いてプログラミング言語のコンパイラ，インタプリタ処理の教材を作成し大学学部生に対する教育を行った．

3. 教育内容

3.1 教育概要

教育の目標は，プログラミング言語間の比較からプログラミング言語の処理構造とデータ構造の理解を深め，プログラミング言語の記述から実行コード生成までの処理を講義と演習から修得することである．受講学生は，C 言語，Java，SQL を学習済みである．

授業は 90 分 16 回で各回の予習・復習に 90 から 180 分の自習を前提とする．概要を表 1 に示す．第 1 回から第 3 回までは前提知識の教育である．第 1 回で python 言語のインストールと対話型モードでの動作確認を行う．第 2 回で python 言語の制御とデータ構造の基本を他プログラミング言語と比較して演習する．第 3 回で，テキストファイル，HTML ファイルの文字列抽出を学習する．HTML パーサは，BeatifulSoup4[4)]を用いる．HTML の日本語の語句抽出方法を形態素解析と n グラムで比較する．

表 1　授業概要

回	概要
1	授業概要, Python の概要と演習
2	プログラミング言語比較, python 演習
3	文字列処理: grep, HTML パーサ, 形態素解析, N グラム
4	汎用的記述: BNF, 正規表現, 構文図, pyparsing
5	字句解析
6-7	構文解析, DSL
8	前半復習
9-10	式の処理, 式の AST 作成
11	ブロック処理, AST 作成, 意味解析
12	代入文を処理するインタプリタ演習
13	実行に必要な処理, 実行環境, 最適化
14	後半復習
15-16	達成度確認

第 4 回から第 12 回はコンパイラやインタプリタの処理を pyparsing による演習等サンプルプログラムを用いて学習する．C 言語サブセットのソースファイルを字句解析する．字句解析ではトークンとトークン種別のリストを出力する．pyparsing ではマッチしたトークンに名前をつけることができるのでこれをトーク種別として DSL(Domain Specific Language)として用いる．字句解析の出力を構文解析し，実行順を管理できるように AST(抽象構文木)を生成する．代入文または条件式は，さらに式の AST を生成する．ブロックについては単純化のために構文解析時に再帰呼出による処理を行わずブロックレベルを調査する．文ごとに python 辞書でブロックレベル，親の文, 子の文のリストを保持する．意味解析では，演算子に対応する python の関数を呼び出すように高階関数によるマッピングを行う．第 12 回では，今までの学習を総合的に確認するため，変数，定数，式を処理するインタプリタサンプルの動作確認を行いシンボルテーブルに

Education of syntactic analysis processing of programming languages using pyparsing

†Hiroshi KUROSE, College of Engineering, Kanazawa Institute of Technology

ついて学ぶ.

第 13 回，および第 14 回は主に講義が中心で，変数のスコープ，ヒープ, 再帰呼び出しなど実行時に必要となる処理を学習する.

3.2 教材

作成したサンプルプログラムは, python 言語の機能学習の他に以下がある.

1) python の文字列メソッド各種
2) python 正規表現を仕様した検索
3) mecab[5]を用いた日本語語句抽出
4) python の文字列スライスによる n グラム処理
5) pyparsing による字句解析
6) pyparsing による構文解析
7) 操車場(デルタ線)アルゴリズムによる RPN 生成
8) 優先順位による多重リスト生成
9) operatorPrecedence による優先順位リスト生成
10) 多重リストから優先度による括弧表記の変換
11) 中置・前置・後置記法の相互変換
12) 深さ優先探索による式の AST 処理
13) ブロックレベル調査
14) 深さ優先探索による構文の AST 処理
15) 括弧の対応チェック
16) AST から Graphviz[6]の dot スクリプト生成
17) スタックマシン
18) 演算子と python 関数のマッピング
19) 変数，定数，式を処理するインタプリタ

数式の項数，優先順位，および，結合方向を学ぶために演算子リストから 2 項演算の優先順位を生成する python プログラムと, pyparsing の operatorPrecedence メソッドとの比較を行った.

構文図の生成にあたって atom[7]エディタプラグイン regex-railroad-diagram[8]やブラウザで動作する描画ソフトウェア railroad-diagram[9]を用いた.

3.3 字句解析の例

文字列，整数，実数を識別し出現する行および桁，トークン長，トークン，および，トークン種別をリストに格納する例をリスト 1 に示す. ここでトークン種別は setResultsName メソッドで指定した名前である．サンプルプトグラムのルール定義を追加していくことで C 言語サブセットの字句解析ができる．字句解析で出力されたトークン種別をさらに解析することで文や式の解析が行える.

4. 結果

python 言語を今回学ぶ学生でも，サンプルプログラムを拡張していくことで式，変数の処理を行うインタプリタの動作を理解できる教材が用意できた.

課題は，2 つある．1 つめはアルゴリズムの動作や，予想した抽出ができない場合の試行に学習者ごとに要する時間に差が大きいため，演習科目に比べて学習者の理解が得られないことである．2 つめは，内容が多岐にわたるため，サンプルプログラムを前提に学習者が確認および拡張を演習で行う場合が多く，コンパイラの処理アルゴリズムや各種解析方法を検討する時間が少ないことである.

リスト 1: pyparsing による字句解析

```
#!/usr/bin/env python3
''' tokenizing sample '''
from pyparsing import *
tokens = list()
string = quotedString
integer= Word( nums )
real   = Combine(integer+'.'+integer)
_s=string .setResultsName('文字列')
_i=integer.setResultsName('整数')
_r=real   .setResultsName('実数')
rule = _s ^ _r ^ _i
s = '''a=1.0+5; b=5*"abc"'''
for i in rule.scanString(s):
   tokens.append( [
     lineno(i[1], s),
     col(i[1], s),
     i[2]-i[1],      # length
     i[0][0],        # token
     [x for x in i[0].asDict()][0]
                     # token type
    ] )
```

5. おわりに

python 言語のパッケージ pyparsing を用いてコンパイラ，インタプリタの構文解析および実行のしくみを学習する教育の一例を紹介した.

参考文献

[1] Stephen C. Johnson, 'Yacc: Yet Another Compiler-Compiler, ' http://dinosaur.compilertools.net/yacc/ index.html
[2] Terence Parr, 'ANTLR (ANother Tool for Language Recognition),' http://www.antlr.org/
[3] Pyparsing, http://pyparsing.wikispaces.com/
[4] Beautiful Soup, https://www.crummy.com/software/BeautifulSoup/
[5] Mecab: Yet Another Poar-of-Speech and Morphological Analyzer, http://taku910.github.io/mecab/
[6] Graphviz – Graph visualization Software, http://www.graphviz.org/
[7] atom A hackable text editor, https://atom.io/
[8] regex-railroad-diagram, https://atom.io/packages/regex-railroad-diagram
[9] tabatkins, 'railroad-diagrams,' https://github.com/tabatkins/railroad-diagrams