CSCI 3415 – Principles of Programming Languages
Fall 2021 – Dr. Doug Williams
Program 1 – Sentence Derivation (Python)
Due Friday, September 24, 2021

In this programming assignment you will create a Python program that reads in a grammar from an input file and then uses that grammar to print leftmost derivations (if they exist) of sentences read from standard input. For example, the grammar from Example 3.1 (p. 116 in the text) would be in a text file as:

```
<program> -> begin <stmt_list> end
<stmt_list> -> <stmt>
             | <stmt> ; <stmt_list>
<stmt> -> <var> = <expression>
<var> -> A | B | C
<expression> -> <var> + <var>
              | <var> - <var>
              | <var>
```

Then, given the input "`begin A = B + C ; B = C end`", your program would output the following leftmost derivation.

```
 1: <program> -> begin <stmt_list> end
 2:           -> begin <stmt> ; <stmt_list> end
 3:           -> begin <var> = <expression> ; <stmt_list> end
 4:           -> begin A = <expression> ; <stmt_list> end
 5:           -> begin A = <var> + <var> ; <stmt_list> end
 6:           -> begin A = B + <var> ; <stmt_list> end
 7:           -> begin A = B + C ; <stmt_list> end
 8:           -> begin A = B + C ; <stmt> end
 9:           -> begin A = B + C ; <var> = <expression> end
10:           -> begin A = B + C ; B = <expression> end
11:           -> begin A = B + C ; B = <var> end
12:           -> begin A = B + C ; B = C end
```

Part 1 – Python

The official web site for the Python programming language is at https://www.python.org/. From there you can download the latest version for recent versions of Windows and Mac OS X. Major Linux distributions include Python. Please use a recent version of Python 3 (the current version is 3.9.7 and you will need 3.6+). Do not use Python 2.

There are alternative ways to get Python that do a better job of maintaining packages for complex Python environments. The most common one is Anaconda (https://www.anaconda.com/), which includes many packages and applications for data sciences. (If you are serious about learning and using Python this is a good way to go. But it is overkill for this programming assignment.)

Part 2 – Grammar File

The grammar file contains the rules for the grammar in Backus-Naur Form (sometimes written as Backus Normal Form), or BNF. Each line of the file contains zero or more rules. Blank lines, which contain zero rules, are ignored. All the elements of the rule(s) are separated by whitespace to make parsing easy.

An example of a typical rule is

```
<program> -> begin <stmt_list> end
```

The left-hand side of the rule must be a single nonterminal, which is `<program>` in this case, the symbol `->` separates the left-hand and right-hand sides of the rule, and the right-hand side is a nonempty sequence of terminals and nonterminals.

The symbol `|` separates alternative rules. For example, the line

```
<var> -> A | B | C
```

specifies three alternative rules.

There is a shortcut notation that allows alternatives to be written on separate lines where subsequent lines after the first begin with the symbol |. In this case, the left-hand side is the same as the previous line(s). For example, the lines:

```
<stmt_list> -> <stmt>
             | <stmt> ; <stmt_list>
```

specify two rules.

Therefore, the grammar above is equivalent to

```
<program> -> begin <stmt_list> end
<stmt_list> -> <stmt>
<stmt_list> -> <stmt> ; <stmt_list>
<stmt> -> <var> = <expression>
<var> -> A
<var> -> B
<var> -> C
<expression> -> <var> + <var>
<expression> -> <var> - <var>
<expression> -> <var>
```

Part 3 – Specifics

Your Python program should accept a command-line argument specifying the grammar file. Therefore, your program will be started using a command like:

```
>python derive.py example_3.1.txt
```

where `derive.py` is the name of the file containing your Python code and `example_3.1.txt` is the name of the grammar file. Your program must be able to parse the example grammar files on Canvas.

After parsing the grammar file and printing the parsed grammar, your program will accept sentences from standard input and then print the leftmost derivation for each, if they exist.

An example execution of the program using Example 3.1 from the text might look like:

```
> python derive.py example_3.1.txt
Reading grammar from example_3.1.txt
<program> -> begin <stmt_list> end
<stmt_list> -> <stmt>
<stmt_list> -> <stmt> ; <stmt_list>
<stmt> -> <var> = <expression>
<var> -> A
<var> -> B
<var> -> C
<expression> -> <var> + <var>
<expression> -> <var> - <var>
<expression> -> <var>
---
Enter a sentence:
begin A = B + C ; B = C end
Sentence:
begin A = B + C ; B = C end
Derivation:
   1: <program> -> begin <stmt_list> end
   2:           -> begin <stmt> ; <stmt_list> end
   3:           -> begin <var> = <expression> ; <stmt_list> end
   4:           -> begin A = <expression> ; <stmt_list> end
   5:           -> begin A = <var> + <var> ; <stmt_list> end
   6:           -> begin A = B + <var> ; <stmt_list> end
   7:           -> begin A = B + C ; <stmt_list> end
   8:           -> begin A = B + C ; <stmt> end
   9:           -> begin A = B + C ; <var> = <expression> end
  10:           -> begin A = B + C ; B = <expression> end
  11:           -> begin A = B + C ; B = <var> end
  12:           -> begin A = B + C ; B = C end
---
Enter a sentence:
begin A = B - C end
Sentence:
begin A = B - C end
Derivation:
   1: <program> -> begin <stmt_list> end
   2:           -> begin <stmt> end
   3:           -> begin <var> = <expression> end
   4:           -> begin A = <expression> end
   5:           -> begin A = <var> - <var> end
   6:           -> begin A = B - <var> end
   7:           -> begin A = B - C end
```

```
---
Enter a sentence:
begin A = A + B + C end
Sentence:
begin A = A + B + C end
Derivation:
No derivation found
---
Enter a sentence:
^Z
```

I put a file derive-hint.py on Canvas that implements the first part of this assignment – it reads the grammar file and prints the parsed grammar. You may use this code as is or modify it as needed.

Part 4 – Pylint

Your code should follow the Python coding standards as defined in PEP 8 – Style Guide for Python Code. The best was to ensure that is to run your code through Pylint.

Pylint is a tool that checks for errors in Python code, tries to enforce a coding standard and looks for code smells. It can also look for certain type errors, it can recommend suggestions about how particular blocks can be refactored and can offer you details about the code's complexity.

Pylint can be downloaded from its web site at https://www.pylint.org/.

The file derive-hint.py has (intentionally) not been scrubbed of warnings from Pylint although it is free of (known) errors. When run through Pylint, it produces the following output.

```
> pylint derive-hint.py
************* Module derive-hint
derive-hint.py:15:0: C0303: Trailing whitespace (trailing-whitespace)
derive-hint.py:18:0: C0303: Trailing whitespace (trailing-whitespace)
derive-hint.py:25:0: C0303: Trailing whitespace (trailing-whitespace)
derive-hint.py:37:0: C0303: Trailing whitespace (trailing-whitespace)
derive-hint.py:52:0: C0303: Trailing whitespace (trailing-whitespace)
derive-hint.py:62:0: C0303: Trailing whitespace (trailing-whitespace)
derive-hint.py:64:0: C0303: Trailing whitespace (trailing-whitespace)
derive-hint.py:67:0: C0303: Trailing whitespace (trailing-whitespace)
derive-hint.py:75:0: C0304: Final newline missing (missing-final-newline)
derive-hint.py:1:0: C0103: Module name "derive-hint" doesn't conform to
snake_case naming style (invalid-name)
derive-hint.py:1:0: C0114: Missing module docstring (missing-module-
docstring)
derive-hint.py:6:0: C0103: Argument name "x" doesn't conform to snake_case
naming style (invalid-name)
derive-hint.py:6:0: C0116: Missing function or method docstring (missing-
function-docstring)
derive-hint.py:10:0: C0103: Argument name "x" doesn't conform to snake_case
naming style (invalid-name)
derive-hint.py:10:0: C0116: Missing function or method docstring (missing-
function-docstring)
```

```
derive-hint.py:14:0: C0116: Missing function or method docstring (missing-
function-docstring)
derive-hint.py:38:27: C0103: Variable name "fp" doesn't conform to snake_case
naming style (invalid-name)
derive-hint.py:56:0: C0116: Missing function or method docstring (missing-
function-docstring)
derive-hint.py:61:0: C0116: Missing function or method docstring (missing-
function-docstring)

-----------------------------------

Your code has been rated at 6.42/10
```

The trailing-whitespace warnings are caused by spaces on blank lines caused by Notepad++ automatic indentation. The invalid-name warnings are caused by short variable names – arguments in this case. The style code only allows short (1 or 2 character) names for indexes and lambda variables. Finally, the missing-docstring warnings are, obviously, because I didn't include documentation strings (docstrings) for the module or top-level procedures.

You will need to run your code through Pylint with no errors or warnings. Note this is a common requirement in industry. In fact, many source code control tools can be configured to automatically run such tools prior to code being checked in.

Part 5 – Report

In addition to the source code for your project, you must turn in a report on the program. This report must include the following sections:

1. Abstract – a short overview of the report
2. Problem Statement – basically the information in this document
3. Approach – how you approached the problem including design, algorithms, etc.
4. Results – results of running the code on sample inputs and the Pylint results
5. Conclusions – believe it or not, you learned things in doing this program

Do not use raw screen shots for the results section – they can be very difficult to read sometimes. Cut and paste the text of program runs using a fixed width font – like Courier New. (This is what I did in this document.)

The report must be in PDF format and submitted with the Python source code on Canvas.

For test cases in the report use the sentences in problems 3-6 and 3-7 from the Problem Set at the end of Chapter 3, which use the grammars from Example 3.2 and Example 3.4, respectively. The files for these grammars are posted on Canvas.