Programming Assigment: Sebesta Scanner in Python (and Go)

CMPS 450 and CMPS 450G: Dr. Maida

September 12, 2021

1 Assignment

Implement the scanner shown in the Sebesta textbook: *Concepts a Programming Languages (10th Ed)*, on page 172 of Chapter 4. In the textbook the program is written in the C language. There is also a translation into the C++ language on Moodle. The C++ version is easier to understand and use because C++ has true constants and a true string datatype.

Translate the program into **both** the the Python and Go languages. The due date for the Python translation is Wed. Sept. 22. The due date for the Go lang translation is Wed. Sept. 29.

Your programs should accept one line of input on file `front.in` that resides in the same directory as the source code. It should give the same output that the C++ does on the given input.

1.1 Grading Criteria

We will run the program from the command line. If the program runs and gives the correct output, you will get 100%. Later projects are much more difficult than this one, so if you can't get this one running, you will have trouble doing the later projects.

Percentage of final grade: The assignments and projects, altogether, are worth 45% of the final grade. This is a small assignment, so each translation is worth about 5% of the final grade. The larger assignments will be worth about 15% of the grade.

1.2 Due Date and Late Policy:

Turn the source code for each program(s) should reside on one file. For the Python program, the file should be named:`scanner_ULID.py`. Similarly, the source code for the Go language program should be named: `scanner_ULID.go`.

Due date: The due date for the Python translation is Wed. Sept. 22. The due date for the Go lang translation is Wed. Sept. 29. There will be a turn-it-in portal on Moodle for you to turn in each version of the assignment.

Late policy: It the assignment is turned in late, 10% is deducted from the assignment grade. If it is more than one week late, 50% is deducted from the assignment grade.

1.3 Academic honesty:

You must write your own programs. If we see identical copies of the same program from different students, we will suspect academic dishonesty. There are penalties for handing in work that is not your own.

2 Hints for working with Python

Most people install python by going to `docs.anaconda.com` and following the instructions for your platform (mac, windows, linux). You will install the individual edition. This gives you access to a whole data science package but you only need Python and, perhaps, the Spyder development environment. Since your program will only be about 100 lines long, you can use your preferred editor and run the program from the command line.

In the Python version, you may not need to import any packages unless you want to do something extra, like handle command line arguments.

The main difference between the C++ scanner and the Python translation is that Python does not have constants. Also, Python has a very counter-intuitive way of declaring global variables, which you'll have to learn how to use. Finally, Python does not have a `switch/cases` statement. So, if you need this, you will need to use nestedn if-then-else states (called an if-then-else) ladder.

One trick to sort of simulate constants in Python is to use dictionaries (aka hashes). This is shown below for the character class constants.

```
CHAR_CLASS = {
    'LETTER'        :   0,
    'DIGIT'         :   1,
    'UNKNOWN'       :   99,
    'EOF'           :   -1
}
```

3 Hints for working with Go

1. Download and install `go` from `golang.org`.
   (a) On my computer (a mac), go is installed in `/usr/local`.
   (b) My Go workspace was installed as a directory named `go` in my home directoy.
2. Read the instructions on how to write Go code.
   (a) Learn what a Go workspace is.

(b) The file extension for Go programs is `.go`.

(c) For this small, one file program, you can use any editor you want. I used the Visual Studio Code editor that got downloaded with Anaconda Python.

(d) From the terminal, type `go --help` to get documentation on the `go build` command to compile your programs. This command creates a binary/executable that you can run from the command line.

3. Below are the packages I used. In Go terminology, this is a factored import statement. When you write your program, you may find yourself using different packages.

```
import (
    "bufio"
    "fmt"
    "io"
    "io/ioutil"
    "log"
    "strings"
    "unicode"
)
```

I wrote the program by reading some of the Go language tutorials at `golang.org`. Next, I looked at the operations in the C++ program and Googled the information on how to do the same thing in Go. My version of the Go program works with the Unicode character set whereas the C++ program works with the ascii character set. Unicode is a superset of ascii and it wasn't more difficult to use unicode. You can use any character set you want as long as it accepts the same type of input as the C++ version of the scanner.