Lab 03 Specification – Developing New PL's
50 points

(TEAM Based Lab -) Due by: 09/20/2021 2:00 PM

Lab Goals

- Practice implementing new Programming Languages using Sly toolkit.
- Learning to work as a team so as to sharpen the team development skills before the course project that is done in a few months.

Learning Assignment

If not done previously, it is strongly recommended to read all of the relevant "GitHub Guides", available at the following website:
`https://guides.github.com/`
that explains how to use many of the features that GitHub provides. This reading assignment is useful to understand how to use both GitHub and GitHub Classroom. To do well on this assignment, it is also recommended to do the reading assignment from the section of the course textbook outlined below:

- **PLP chapter 01, section 1.4, 1.5**

Assignment Details

Now that we have discussed some basics of Programming Languages, we are ready to explore development of a new PL. In this lab, we are going to develop new PL using Sly and work as a Team.

In this lab, students will practice developing syntactic and semantic components of new PL using the Sly toolkit. This is a good practice to retain the knowledge from the class discussions so far.

Students are not expected to achieve technical mastery in any single language in this course. Instead, we are trying to learn different languages in the context of studing the principles of programming languages. In other words, our goal is to put the principles of programming languages at the forefront and not achieving language mastery.

At any duration during and/or after the lab, students are recommended to team up with the Professor and the TL(s) to clarify if there is any confusion related to the lab and/or class materials. The Professor proofread the document more than once, if there is an error in the document, it will be much appreciated if you can communicate that to the Professor. The class will be then informed as soon as possible regarding the error in the document. Additionally, it is highly recommended that students will reach out to the Professor in advance of the lab submission with any questions. Waiting till the last minute will minimize the student's chances to get proper assistance from the Professor and the Technical Leader(s).

Students are recommended to get started with this part in the laboratory session, by discussing ideas and clarifying with the Professor and the Technical Leader(s). Late submission is accepted for the part(s) in this section, based on the late policy outlined in the course syllabus.

It is required for all students to follow the honor code. Some important points from the class honor code are outlined below for your reference:

1. Students are not allowed to share code files and/or other implementation details. It is acceptable to have a healthy discussion with your peers. However, this discussion should be limited to sharing ideas only.

2. Submitting a copy of the other's program(s) and technical reports is strictly not allowed. Although this lab is a Team based, we are limited to collaborate within our Team. Please note that all work done during lab sessions will be an opportunity for students to learn, practice, and master the materials taught in this course. By doing the work, students maximize the learning and increase the chances to do well in other assessments such as skill tests, exams, etc ···

Preliminary Steps

It is important that you can set up Docker and GitHub to complete the rest of the lab. Please follow the guidelines below to complete the preliminary steps.

1. **[Docker Setup.]** At this point, I expect the MAC, Linux, and Windows Pro users, to have this step completed based on our previous class discussions. For those who had not completed this step, the documentation below should provide more details regarding the download and installation setup.

   - Get Docker setup completed on your laptops:
   - Docker Mac Setup:

     `https:/docs.docker.com/docker-for-mac/install/`
   - Docker Ubuntu Setup

     `https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-18-04`
   - Docker Windows Setup:

     `https:/docs.docker.com/docker-for-windows/install/`
   - If the setup goes correctly as desired, you should be able to get started and validate the Docker version and run the hello world docker container using the following commands:

     docker –version

     docker run hello-world
   - There are some more documentation for Docker get started to test your installation in the link provided below:

     `https:/docs.docker.com/docker-for-mac/`

     `https:/docs.docker.com/docker-for-windows/`

2. **[Loading Docker Container.]** There are two steps in loading the container, namely:

   - Build the container
   - Connect and Run the container

   Build the container: So to build the container. the following steps should be performed.

   (a) First, accept the lab URL provided in Slack. After downloading the lab folder from the GitHub classroom, navigate to the cmpsc201-fall-21-lab03 directory using terminal (Mac/Ubuntu) or Command Prompt/Docker quick start terminal (windows).

   (b) Build the docker image using the following command:

   docker build -t cs201lab03 .

   Please note, you are required to have the period in the command above.

   (c) Note: In the command above, cs201lab03 is the user-provided image name. This could be random. But it is recommended to use the same name to easily follow the rest of this document. Additionally, it is required to be inside the `cmpsc201-fall-21-lab03` directory to run the build command. If you are not inside the `cmpsc201-fall-21-lab03` directory, you may receive an error message.

   (d) Upon successful build, it is recommended to verify the correctness of image creation by using the following command:

   docker image ls

   (e) The image named ”cs201lab03” should be listed as one of the outputs from the command above.

   Connect and Run the container: So to create and run the container. the following steps should be performed.

   (a) Run the docker container based on the image created in the previous steps using the following command:

   Mac/Ubuntu:

   ```
   docker run --rm -v $(pwd)/src:/root -it cs201lab03
   ```

   Windows:

   ```
   docker run --rm -v "%cd%/src":/root -it cs201lab03
   ```

   Ignore if you receive a bash command not found error while connecting to the docker container. The command above simply takes us into the docker container so that we can start to execute the code.

   (b) To run the above command, it is required to be inside the `cmpsc201-fall-21-lab03` directory. And, please note, you will log in to the container after entering the above command.

   (c) After creating the container, the run command above creates a mount between the host machine and the container with a shared folder space. So, any files placed inside the host mount directory can be easily accessible inside the container mount directory and vice versa.

   (d) Run the Hello World Python program using

   python3 hello.py

3. **[GitHub Setup.]** Take a look at the detailed documentation for getting started with GitHub, which is available at: `https://docs.github.com/en/get-started`

   You are required to know the procedure to git clone, git pull, git add, git commit, and git push to access the lab specification folder and to submit your lab for grading purposes. If there is an issue with your GitHub setup please discuss it with your Technical Leader(s) and/or the Professor.

Developing new PL’s (50 points)

For this assignment, we will be using a group assignment functionality of GitHub Classroom. For group assignments only one person will be creating the team on GitHub Classroom while the other team members will join that team. Please form a team consisting of two or three members, assign one person to be the designated team manager.

The selected team manager should go into the #announcement channel in our Slack team and find the announcement that provides a link for it. Copy this link and paste it into your web browser. Now, you should accept the laboratory assignment and create a new team with a unique and descriptive team name (under ”Or Create a new team”). Now the other members of the team can click on the assignment link in the #announcement channel and select their team from the list under ”Join an Existing Team”. When other team members join their group in GitHub Classroom, a team is created in our GitHub organization. Every team member will be able to push and pull to their team’s repository.

Team Work:
Each member of the class will be arranged into a team at the beginning of the lab session. Class members are invited to select their own teams consisting of two or three individuals.

Your team should use GitHub and its features (e.g., issue tracker, pull requests, commit log, and code review request) to complete all of the tasks referenced in the previous section. Since multiple approaches may support the effective completion of the required documents, this assignment does not dictate team organization or communication strategies.

Creating a Programming Language:
For this laboratory assignment, your task is to design a specification for a very simple toy programming language and implement a scanner and a parser for this language in the file named `toy-pl.py` in the src directory. You are encouraged to be creative in your design of a language, however, it should satisfy certain minimum requirements outlined below.

Grammar:
Once you have an idea for the type of language to construct, the first step should be to define its grammar. Please construct grammar rules in a BNF format that generates your language in the appropriate section of the **Technical-Report** file.

Your grammar should be a CFG grammar that starts with a start terminal and provides rules for things such as statements, expressions, and derivations of all non-terminals. Ps examples in Lesson-4 slides.

Scanner:
After you identified the features that your source language will support through the design of its grammar, you should implement a scanner to produce a token stream for your language. You can use [SLY] `https://github.com/dabeaz/sly` to make the implementation of the scanner easier, or alternatively, you can also use its predecessor [PLY] `https://www.dabeaz.com/ply/`.

Your scanner will transform the source file into a series of meaningful tokens containing information that will be used by the parser. You should begin by making a list of all items your scanner needs to handle based on the features of your source language.

At the minimum, your scanner should support the following:

1. Skip over white space (if appropriate for your source language),
2. Recognize all keywords, and return the correct token,
3. Recognize literals and return the correct token,
4. Recognize identifiers and return the correct token,
5. Recognize at least three data types (for example, int, float and a String),
6. Recognize at least one conditional statement, please look at an example of conditional statement use in **example.py**,
7. Recognize new lines,
8. Recognize comments,
9. Report lexical errors.

It is recommended that you add token types one at a time and test after each addition. For any items that are not explicitly defined in your source language documentation, you can make any assumptions as appropriate.

Parser:
Once you can generate a token stream for various inputs appropriate for your language, you will expand the front-end of your compiler by implementing a parser for it. Please note that at this stage it is likely that you will need to expand and/or modify parts of your scanner to get your parsing to work correctly - this is okay and expected! Any conflicts can be resolved in a multitude of ways (rewriting the productions, setting precedence, etc.) and you are free to take whatever approach appeals to you. All you need to ensure is that you end up with grammar that is free of conflicts and errors.

To implement a parser you need to add the rules for each of your language's grammar features. Remember that syntax analysis is only responsible for verifying that the sequence of tokens forms a valid sentence given the definition of the grammar. Given that some grammars can be somewhat loose, some nonsensical constructions will parse correctly since you are not doing any of the work to verify semantic validity (type checking, declare before use, etc.)

Test Cases:
To ensure that your scanner and your parser operate correctly, you are to create at least ten input examples to use in testing. These inputs should contain Strings that represent different combinations of what your language allows and should contain an equal number of inputs that run without errors and those that produce some lexical and parsing errors. The output from the input test cases can be printed to the terminal or the file. You need to add the ten input examples in the **Technical-Report** file so it can be used to test your code during the grading phase.

Part 02 - Honor Code

Make sure to **Sign** the following statement in the `honor-code.txt` file in your repository. To sign your name, simply replace Student Name with your name. The lab work will not be graded unless the honor code file is signed by you. I expect that all team members sign their name.

This work is mine unless otherwise cited - Student Name

Part 03 - Reflection

Add a Reflection to the repository by modifying the `reflection` file in the lab repository. List out the biggest learning points and any challenges that you have encountered during this lab. I expect that all team members individually add their reflection in the reflection file.

PS next page ...

Submission Details

For this assignment, please submit the following to your GitHub repository by using the link shared to you by the Professor:

1. Modified source code in `toy-pl.py` file.
2. A document containing the technical report in the file named `Technical-Report` that provides the list of grammar and test cases for testing your new language.
3. A document containing the reflection of the lab in the file named `reflection`.
4. A signed honor code file, named `Honorcode`.
5. To reiterate, it is highly important, for you to meet the honor code standards provided by the college. The honor code policy can be accessed through the course syllabus.

Grading Rubric

1. There will be full points awarded for the lab if all the requirements in the lab specification are correctly implemented. Partial credits may be awarded if deemed appropriate.
2. Failure to upload the lab assignment code to your GitHub repository will lead to receiving no points given for the lab submission. In this case, there is no solid base to grade the work.
3. There will be no partial credit awarded if your code doesn't compile correctly. It is highly recommended to validate if the correct version of the code is being submitted before the due date and make sure to follow the honor code policy described in the syllabus. If it is a late submission, then it is the student's responsibility to let the professor know about it after the final submission in GitHub. In this way, an updated version of the student's submission will be used for grading. If the student did not communicate about the late submission, then automatically, the most updated version before the submission deadline will be used for grading purposes. If the student had not submitted any code, then, in this case, there are no points awarded to the student.
4. If a student needs any clarification on their lab grade, it is strongly recommended to talk to the Professor. The lab grade may be changed if deemed appropriate.