CS351 Final Project - Python Parser Visualizer

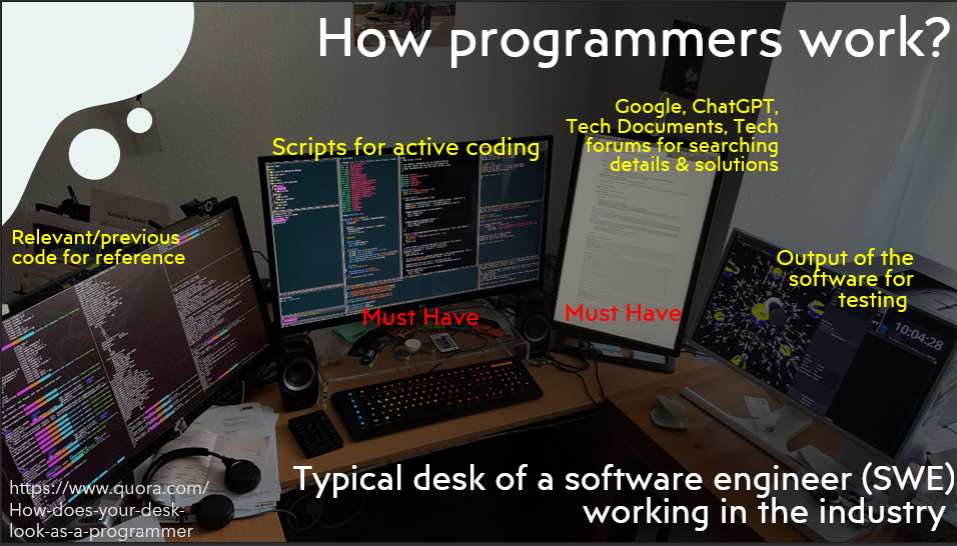
CS job market typically needs 2-3 good projects on your resume. To help everyone make a bigger project that could be one of your collections of resume projects, we will keep building the HW7 parser project to include data structure support and better output visualization.

This HW is **group work**, only 1 people in a group need to submit. When submitting, make sure you add your name under the Canvas-People-Final Project group and also include group members' name to the comment section. Only the newest submission from a group will be graded.

You will need the complete answers from the below works to form the answer for this HW.

--- HW7: python project

**--- Search engine (Google, new Bing, Github, ChatGPT etc.) to help you find libraries to use**



Programmers working in industry normally have 2+ monitors with 1 showing IDE for coding and 1 showing search engine to find libraries, forum posts, etc. No one can remember all the language and package details to code without technical searching. **Searching and directly using online resources is among the key problem-solving skills that programmers need.**

We have been trying out self-direct learning and technical searching this semester. We will keep practicing this skill in the final project. The amount of searching and integrating solution is closer to a small real work project than a HW assignments.

**You will submit the source code, video clip, and a small report.**

**Step 1: confirm your HW7 is working**

In HW7, we can read several lines of code, give out a list of tokens for each line of code, and print out verbal description of the parse tree building process. Below lines are included in the HW7 assignment. We will use the same example source code in this project.

**float mathresult1 = 5\*4.3 + 2.1;**

**float mathresult2 = 4.1 + 2\*5.5;** (test cases will be similar to these lines)

**if (mathresult1 >mathresult2):**

**print(“I just built some parse trees” );**

Again, **some tokens are separated by spaces, some are not**. In this project, you will continue adding more code into your HW7 lexer GUI. Testing cases will be limited to the ones that can be processed HW7 with small variation to confirm that you are not hard coding test cases.

**Step 2: Find and add tree data structure**

In HW7, we directly printed out the verbal description of tree building process but did not implement a tree data structure. In this project, you need to work as a team and find a tree data structure library to import or source code to include in your project.

Please put a clear citation on top of the tree script to describe the source of the code. You can also choose to implement it from scratch if you want to practice tree data structure. Here is a starting point: <https://www.tutorialspoint.com/python_data_structure/python_binary_tree.htm> .

Modify HW7, besides output tree building process to the 3rd output box, also add nodes to the tree data structure code while going through the parser steps.

You should include a print tree (to the shell/command line output) or other similar function to prove that you saved nodes into a tree data structure. You could also go over your code in debug mode in the video submission to prove you used the back-end data structure support.

**Step 3: Modify GUI**

Please add a 4th bigger output box below the 3 boxes we had from HW7. This will be the tree visualization output box. Please use the picture below as reference.

Note that there are different libraries for tkinter graphical output for tree. They have varying output format and different difficulty levels. Please read Step 4 to select a method that your team will use.

**You can keep drawing several trees for the different lines of input to form a long list of output or refresh the canvas to draw 1 tree at a time.**

**Required GUI functions for this project**:

math

=

Identifier

Keyword

Expression

ADD the 4th output box for tree visualization:

**Below is just an example tree, your output should match what the above 3rd Parse Tree output describes using the terms we used in BNF grammar from HW7**.

The actual output tree visual format will be decided by the package you found. It might look different from the example drawing. (Details see Step 4.)

Current line number: 1

Exit

Next Line

<keyword, float>

<identifier, mathresult1>

<operator, =>

<literal, 5>

and so on

float mathresult1 = 5\*4.3 + 2.1;

float mathresult2 = 4.1 + 2\*5.5;

if (mathresult1 >mathresult2):

print(“I just built some parse trees” );

Lexer and Parser for TinyPie

A good, formatted way to show the process of your parse tree building. You can use the example code's format.

**####Parse tree for line 1###**

**--- parent node…**

And so on

Parse Tree

Tokens

Source Code

**Step 4: Visualize tree output**

There are multiple ways to output graphics to tkinter. Please communicate with your team and use one of the below options to output your tree. **The better looking, more complex method will give you a higher grade than easier method that might not show tree diagram as well as other methods.**

Chart, pie chart

Description automatically generatedOption 1 (difficult, higher grade): The above picture shows drawing to the Canvas widget (<https://www.tutorialspoint.com/python/tk_canvas.htm> ). You can search on Google, Bing, etc., or use ChatGPT to help you generate graphical output, like boxes and lines, to form a tree graph. This is beginning computer graphics project. You can experience what it feels like to draw 2D CG output to computer screen.

Diagram

Description automatically generatedOption 2 (difficult, higher grade): Find any library that helps you directly call their functions to output tree diagrams. This is one possibility ( <http://books.gigatux.nl/mirror/pythonprogramming/0596000855_python2-CHP-17-SECT-10.html> ). Note that this example uses python 2 and will require significant work to merge into your code. It uses Canvas and labels. You could also read the code and learn the key pieces and write a simplified version in your code with the help of search engine or ChatGPT.

Graphical user interface, application

Description automatically generatedOption 3 (Easier, slightly lower grade): Use the Treeview (<https://tkdocs.com/tutorial/tree.html> ) widget to output your tree. It is a built-in widget library with better tutorials (<https://www.geeksforgeeks.org/hierarchical-treeview-in-python-gui-application/> ), so this option is easier to work with. But the final grade will be a bit lower than the teams who successfully implemented actual tree visualization.

A screenshot of a computer

Description automatically generated with medium confidenceOption 4 (Easier, slightly lower grade): You could print out formatted text to the Text widget and make the output look like a tree. Note that, when the input changes slightly, like 4 numbers math instead of 3 numbers, your tree format still needs to work. The part of printing out correct | - and spaces is tnhe more challenging aspect of this method.

Option 5 (Could be a difficult or easier solution. Grading based on final solution quality that looks like option 1 and 2, or option 3): Use any other method that you found but the final output needs to be a complete project software. For example, have another smaller program (could be not Python) generate tree picture and use Python Canvas to directly load the result picture. Please note that, you need to use other program’s API to do real-time query to make a software. You cannot draw pictures, manually save them locally to show. We will modify the test cases to make sure your program can process tree visualization in real time.

**Step 5: report and video recording**

Since the project is “semi-open” and you can use different ways to achieve output result, please write a short report (2-3) pages helping us understand your project. You need to have the components below in the project.

1. Team member names, individual contribution to the project, percentage contribution to the project. Your final grade will be adjusted based on your individual contribution.

|  |  |  |
| --- | --- | --- |
| Name | Contribution | Contribution percentage |
| AAA | Completed…. | 60% |
| BBB | Debugged… | 40% |

1. How did you implement the Tree data structure. Which search engine helped you? Citation and any comments.
2. How did you implement the Visualizer. Where did you find help. Citations on any library you used and any comments.
3. Screenshots of your working project.
4. What went well in your project team collaboration? Did you meet challenges? How did you solve them.
5. How to run your project. Where do you show proof that your tree data structure is working.

Please submit a small video (or several video clips) showcasing your working project. You can use Zoom screenshare + record to local to capture videos. You can submit a link for the long video (on your cloud drive) as additional resources. But you NEED to submit some smaller videos showing key components of your project on Canvas for record keeping purposes.

**Again, your parser GUI can have a different appearance from this example. We check for functionalities when grading. Submit only a few Python 3 .py file that we can run and check if your parser works + 1 pdf/word containing report + a few video clips showcasing running project & important code pieces**. This is group work.

**Grading (15pt):**

2pt – submission of working HW7 and you have the 4th output box with some information provided beyond text output showing you are trying to generate tree visualization (like drawing some squares).

4pt – you have working Tree data structure either from library or code from scratch.

6 pt—Tree visualization

3 pt—Report + video submission.