Zackery Devers

12/6/2021

Final project

Trivia pl language

Cmpsci 201

For my project I will be taking the project 1 track of creating our own programming language with the use of python. I felt I would do best in this project track because we did the creation of our own project file during our time in class. In my design I want to make a programming language that deals with something that will interest me. Most likely I wanted to make a program that could spit out facts to the user when asking questions pertaining to it. In my language you could input a number and with that number the random information fact could spit out. The purpose of the language is to just get users more informed on random facts that I enjoy stating. It won't be the most complicated language because if I can understand it I can believe my peers will as well. I will in all create a programming language that will act as a fact reader for whatever number a user inputs. The user will use a command line prompt in order to get these facts displayed.

To get my project started I needed to create the file that will incase the contents of my code. I named this main file trivial.py so it would be usable with the python language. To start off the file I chose to import the lexer and parser from the sly packaging. This will allow us to have an interactive programming language. Next we initialized a class tittle MPLLexer that will act as our lexer for the programming language. The tokens were then declared with the two inputs of fact and option. Stating fact in the command line text

will allow you to then sort through all of our options. The ignore statement allows the lexer to ignore the tabs in the text. The literals are the inputted command in the command line text that allows the parser to not bug out from typing the full input to get the PL to work. The fact token is also passed through with the allowed text of “fact”. Next up in the code is the option that allows the user to type in any combination of letters. The final chunk of code in our lexer section is a newline occurrence and a error occurrence that allows the program to create new lines when needed and also display errors if an option was inputted and found to be incorrect.

The next part of our code was the parser class that we initialized as MPLParser. The tokens for these was set to the lexer tokens in order to keep the program running correctly and that the lexer could be parsed with the same tokens. Next we designed the cases for the tokens. Our first if statement was the “one” that prints a quoted fact that we wrote in. You are able to select one through twenty for the options available for our tokens. To correctly use this program you must type in console “ fact : one, two, three… all the way to twenty”. The rest of the options we have from the 2-20 are all elifs that allow the parser to select from one to twenty with ease.

Our final chunk of code in the project is the main. The main has the two lexer and parser variables defined to be able to call back from the previous chunks of code. This allows the main to be able to use all the declared rules and options for our user to get facts. There is a use of a while statement that keeps the program running. The try statement is then ran through only while “true”. The input instruction is printed out with the statement “Type -fact : number 1-20” in letters. If it is unable to try then the except eoferror is ran and makes the code break. Under these an if statement is created that runs if the text is passable text. The lex variable is made and it is a new lexer.tokenize the text. This takes the created text and makes it work with the derived tokens which than can be parsed. After these tokens are created the parse then pareses the lex. That is the end of our code.