Description

I have created a JSON parser using the Ply module. It works by first constructing a lexer with the appropriate tokens, and then constructing the parser with appropriate parsing rules. I also applied a series of unit tests to determine whether my parser is working correctly or not, and they have passed all 35 tests. Instructions to use the program is in the README file.

The tokens are as follows: LBRACE, RBRACE, LSQBRACKET, RSQBRACKET, COLON, COMMA, QUOTATION, BACKSLASH, FLOAT, INT, TRUE, FALSE, NULL, UNICODE_CHAR; The parsing rules are as follows: OBJECT, MEMBERS, MEMBER, STRING, CHARACTERS, CHARACTER, NUMBER, ARRAY, VALUELIST, VALUE.

An interesting note about parsing strings: In my program, it finds the appropriate character tokens and using a suitable parsing rule, parses them up to be a string. This could have been changed to a string token with the correct regex expression, however I chose not to do so because of the requirements that strings have Unicode characters. I preferred to explicitly declare Unicode character tokens rather than use the ^ operator on a group of undesired symbols to form a string, so I know what characters I am using. However, I understand that doing so might lose some efficiency and speed due to additional parsing rules. Besides that, to avoid single backslashes and quotations in my string, I chose to tokenise them and explicitly avoided using them in my string parsing rule.

Another interesting design choice was to use INT and FLOAT tokens instead of a NUMBER token, I felt that this was easier and more understandable to use two tokens instead of one long regex expression.

There were some warnings about shift/reduce conflicts but because Ply automatically solves these conflicts using shift, nothing needed to be changed as this is what was intended.

References

1. David M. Beazley**.** *PLY (Python Lex-Yacc) Documentation.* Available from: https://www.dabeaz.com/ply/
2. Andrew Dalke. *Parsing with PLY.* Available from: http://www.dalkescientific.com/writings/NBN/parsing_with_ply.html
3. JSON Specification. Available from: https://www.json.org/