Lexer

Kit Lao

April 24th, 2020

Goal

To make parsing easier, take the input in as a text and generate a list of tokens where each token corresponds to either a number of an operator.

Design

Tokens

TokenType is an enumeration that contains, NUMBER, PLUS, MINUS, MULTIPLY, DIVIDE, L-PAREN, AND R-PAREN.

Token class is of type TokenType. The value it holds could be the floating point representation of a number, if the type is NUMBER, or None, otherwise.

Lexer

To generate the sequence of tokens, iterate through the text, and generate a token using the token module. Ignore all white-spaces, new-lines or tabs. If there is an invalid token, raise an error.

To generate tokens for numbers, iterate through the text while it is still pointing at a number or decimal. While iterating, build up the string representation of the number, and check to see if there is a valid number of decimals. To handle the edge cases for floats without digits in front of the decimal, add a zero in front. Same for if there aren't digits behind the decimal.

Resources

dataclasses

The dataclass module from the dataclasses package will help indicating tokens with particular TokenTypes easier.

Testing

Samples

```
>>> 0.12
[NUMBER:0.12]
>>> .12
[NUMBER:0.12]
>>> 12.
[NUMBER:12.0]
>>> 1 + 4
[NUMBER:1.0, PLUS, NUMBER:4.0]
>>> 3 + 4 * 2
[NUMBER:3.0, PLUS, NUMBER:4.0, MINUS, NUMBER:2.0]
>>> 3 ^ 2
Exception: Illegal character '^'
```

Script

The output is trivial so I didn't make a testing script.