Parser

https://github.com/GabiMarginean/FLCD

An LL(1) parser implemented in python.

- `Grammar` class consists of lists for non terminal symbols, terminal symbols, start symbol and a map for productions and the indexes assigned to each production.

  > The constructor takes as argument a filename containing the grammar represented in the following form:

  ```
  line1 = list                /* nonterminals     */
  line2 = list                /* terminals        */
  line3 = symbol              /* start symbol     */
  remaining_lines = rules     /* production rules */

  list = terminals_lst = symbol | symbol "," terminals_lst
  rules = rule | rule "newline" rules
  rule = symbol "->" symbol_lst | symbol_lst
  symbol_lst = symbol | symbol "space" symbol_lst
  symbol = "a-z" | "A-z" | "1-9" | "ε" | "(" | ")" etc
  ```

```
class Grammar():

  def __init__(self, fileName):

    self.non_terminals  = []
    self.terminals    = []
    self.start      = None
    self.productions  = {}
    self.prod_idx       = {}

    self._readFile(fileName)

    for (idx, prod) in enumerate(self.productions.keys(), 1):
      self.prod_idx[prod] = idx
```

- The `Parser` class is used to parse a sequence of characters and verify whether or not it is accepted and output the errors if any.

  The `Parser` is using a `Grammar` object and has internally the following methods:

- `first`
- `generate_follow_dict`
- `gen_parsing_table`

The method `parse(sequence)` is using the aforementioned methods in order to parse a given sequence.

- If the **sequence is accepted** the message `"Sequence accepted!"` will be printed on screen and the `productions string` will be returned as a string of numbers consisting of the indexes of the productions from the grammar file, numbered in the order they appear.
- If the **sequence is rejected** the message `"Sequence rejected!"` will be printed on screen as well as a message in the form `"Unexpected symbol '0' at index 1: -01"` and `None` will be returned.

- The class `ParserOutput` is a helper class that consists of a `Grammar` and a `Parser` and checks wether a sequence from a file is accepted or not and writes the resulting production string in a file . The input grammar and output file name are passed as parameters to when constructing an 'ParserOutput' object. If no output file name is passed then the result will be written to screen only.

  The method `check_sequence(sequence)` is used to verify a sequence with the specified grammar.