Lexical Analysis
Recognizing tokens
Language to describe various token types
+; ; , ;

Regular expression pattern matching
Each token type is specified by a pattern (regular expr) in a string (or tree) matching a regular expr, or it doesn't.

Lexer analyzer reads the input until (end or) some part of the input matches one of the regular expressions.

Scanner looks for longest possible prefix of the input that matches a regex.
In C:
\[ 1 + \]

C = a+++ +bb;

Regular expr for iden
Primitives
0 matches nothing
E matches the empty string and nothing else.
A, a is an integer char (except for negation), special characters.
R is an operator

Examples:

- -> ternary, or, if, |
- is an operator, then
- ( is a regex matching anything that r matches or a child's mother

- combination (conjunction)
- matches any string with two public modifiers, and the rest of the string matching s.

- * (Kleene closure)
- matches any string, that is, the concatenation of any number (zero or more) strings matching r.

- [1|2|3|4|5|6|7|8|9|0]*
- [1|2|3|4|5|6|7|8|9|0|1|2|3|4|5|6|7|8|9|0]*

- is valid
- ( is a regex matching anything that pattern matches or a child's mother

- highest precedence (top)
- addition - lower precedence
- (top - middle)
- (top - middle - bottom)
- (top - middle - bottom)

- leftmost precedence
- (bottom - top - leftmost)
- (bottom - top - leftmost)