- Terminals are simply written out: `while`
- Nonterminals are enclosed in angle brackets: `<statement>`
- Productions are in the form:

  `<nonterminal> ::= <sequence of terminals or nonterminals>`

  `<sentence> ::= <noun phrase><verb phrase>`
- We can use | to represent or

Simple Example

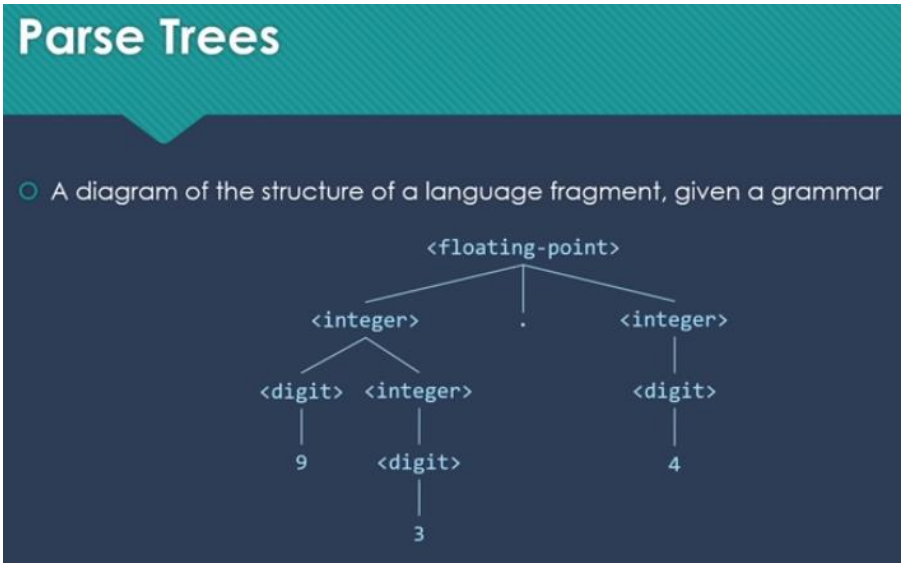
```
<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<integer> ::= <digit> | <digit><integer>
<floating point> ::= <integer>.<integer>
```

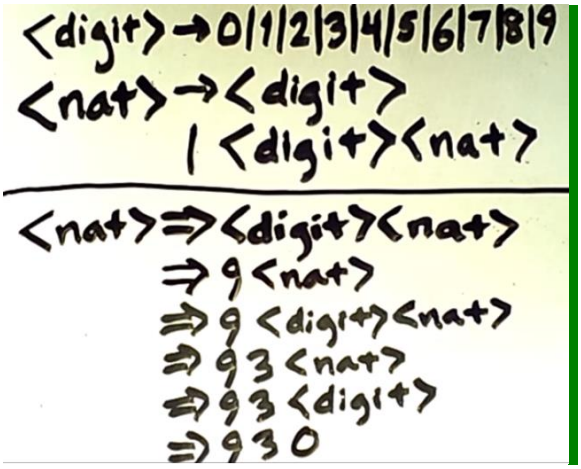
More Complex Example

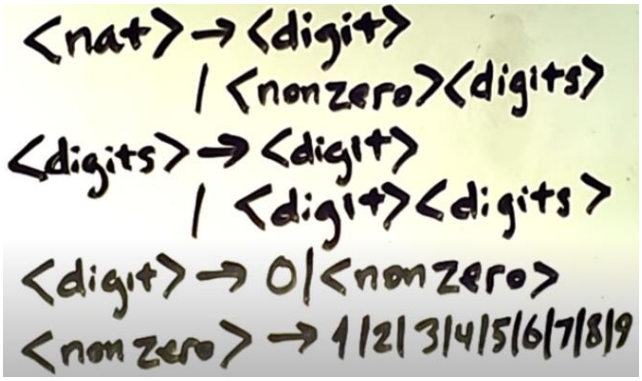
```
<statement> ::= <if-statement> | <while-statement> |
                <assignment> | <block>
<assignment> ::= <variable>=<expression>
<if-statement> ::= if (<condition>) <statement> |
                   if (<condition>) <statement> else <statement>
<while-statement> ::= while (<condition>) <statement>
<block> ::= { <statement-sequence> }
<statement-sequence> ::= <statement> | <statement><statement-sequence>
```

Lexical vs Phrase Structure

- Lexical structure is the structure of valid tokens in a language:
  - numbers
  - identifiers
- Phrase structure is the larger structure of sentences and programs.
- We usually use separate grammars to handle these because grammars for both get very complex.
- Compilers and interpreters usually also handle these separately







1st case example

<sentence>:: <subject> <predicate>

<subject>:: <article> <noun>

<article>:: the / a

<noun>:: boy / girl

<predicate>::<verb><noun>

<verb> :: kicks / sees

<noun>:: boy / girl

1. The boy sees girl = valid
2. A girl kicks boy = valid
3. Boy sees boy = invalid