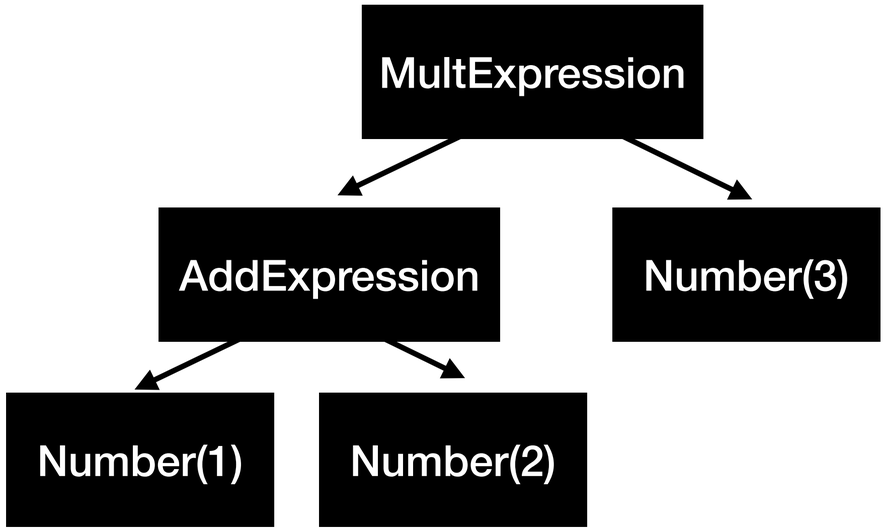
Overview: Parsing

- Starts with lexemes produced by lexer
- Groups lexemes into expressions, turning them into an AST (abstract syntax tree)
- Makes sure that program is semantically valid
- Example: "(1 + 2) * 3" could turn into this AST:
- But "(1 + 2" would be thrown out as an invalid program, because the parentheses are mismatched



Overview: AST evaluation

- Once the AST is generated, the interpreter walks along the AST (in our case, we will make a "tree-walk" interpreter that really does this) and evaluates each group of nodes accordingly
- Typically will have a function for each type of node, that knows how to deal with processing a node and its children
- Classic example of recursion, as a given node type may have a node of the same type in one of its subtrees - for example, a list may contain another list, and each may be processed by the same "process_list(tree)" function