

COMP 410: Abstract Syntax Trees and Evaluation

Kyle Dewey

Abstract Syntax Tree

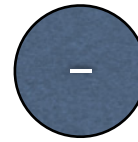
- Abbreviation: AST
- Unambiguous tree-based representation of a sentence in a language
- Very commonly used in compilers, interpreters, and related software

–Generally we work with ASTs instead of Strings or any other code representation

$$(1 + 2) - 3 * 4$$

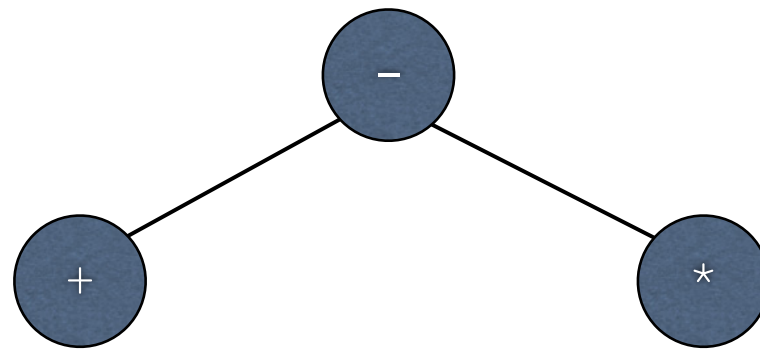
–Key parts: we need parentheses to direct that $1 + 2$ happens first. We know that the $3 * 4$ should happen after the part in parentheses from PEMDAS rules

(1 + 2) - 3 * 4



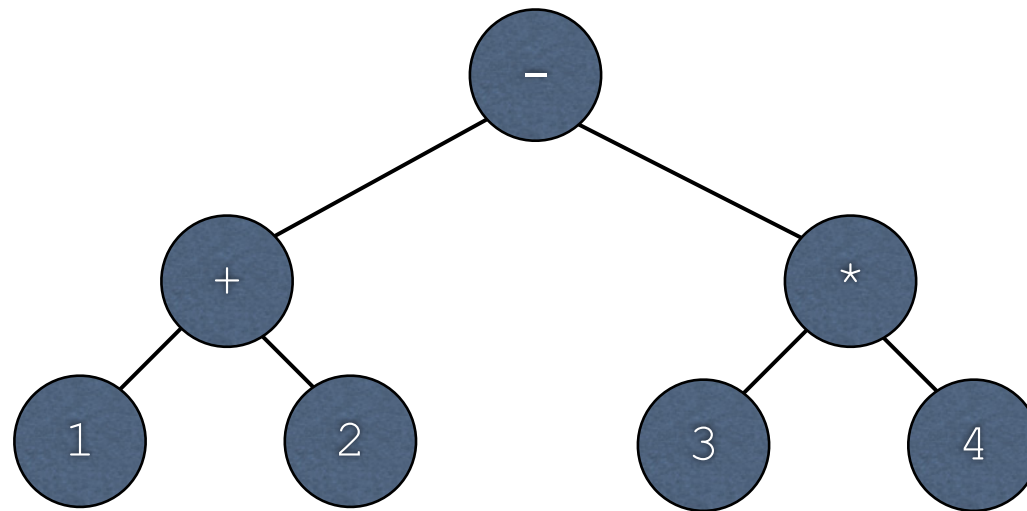
-Lowest priority thing ends up in the top of the tree

(1 + 2) - 3 * 4



-Next level of priority

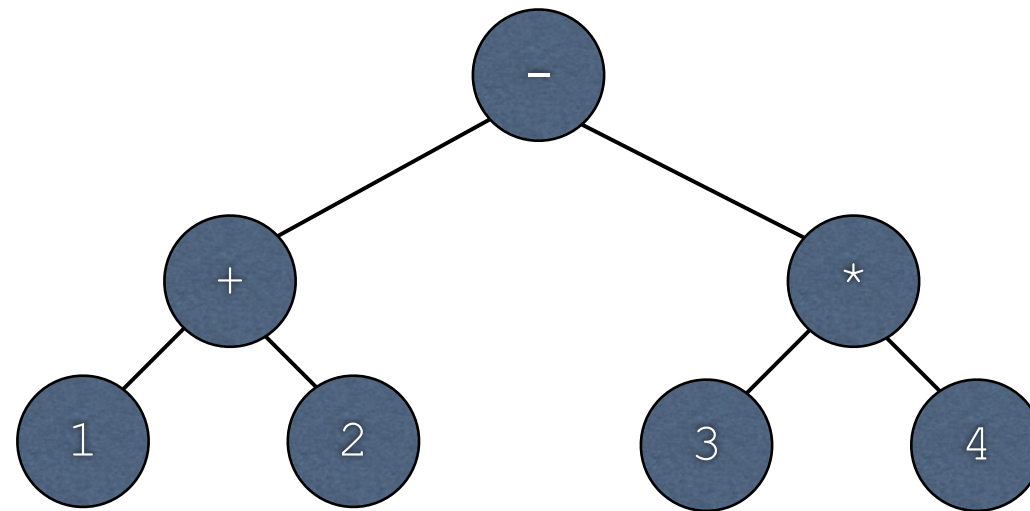
$$(1 + 2) - 3 * 4$$



–Next level of priority

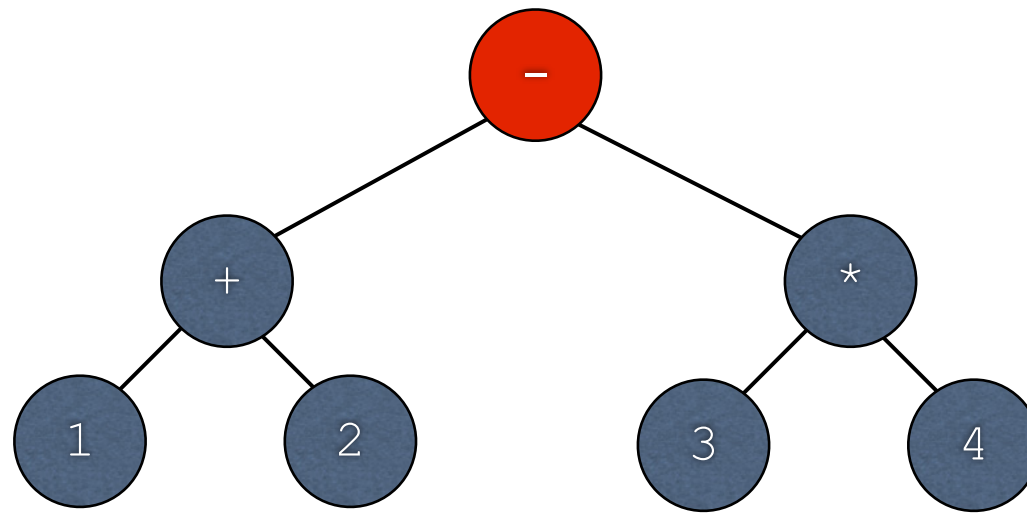
Exercise: First Side of AST/Evaluation Sheet

Evaluation



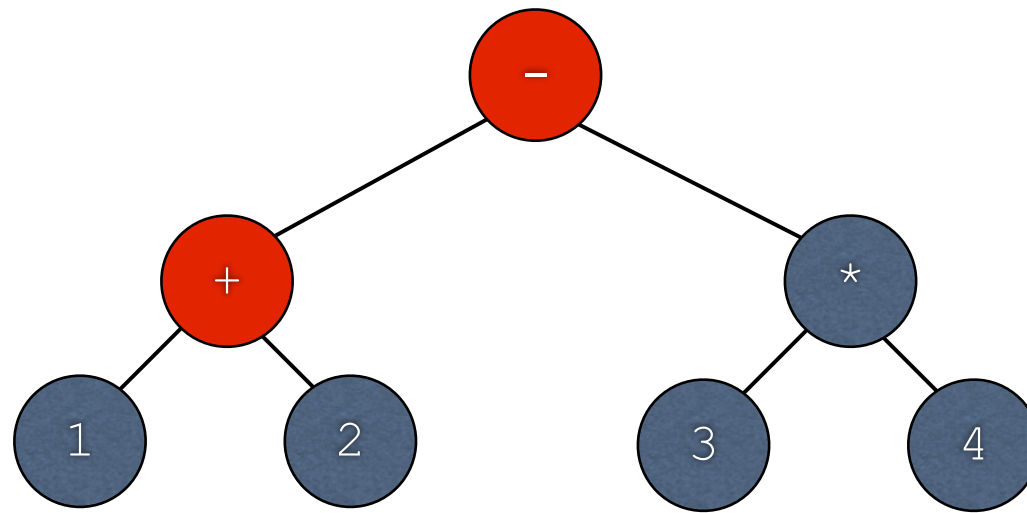
- Key point: bubble-up values from the leaves
- This can be implemented in code via a recursive function starting from the root (code in a bit later)

Evaluation



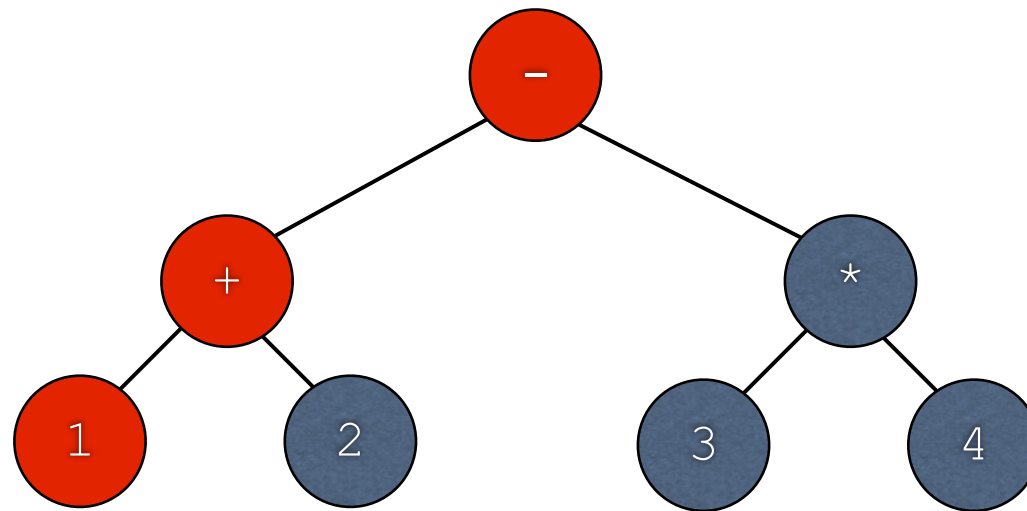
–We start evaluation from the root...

Evaluation



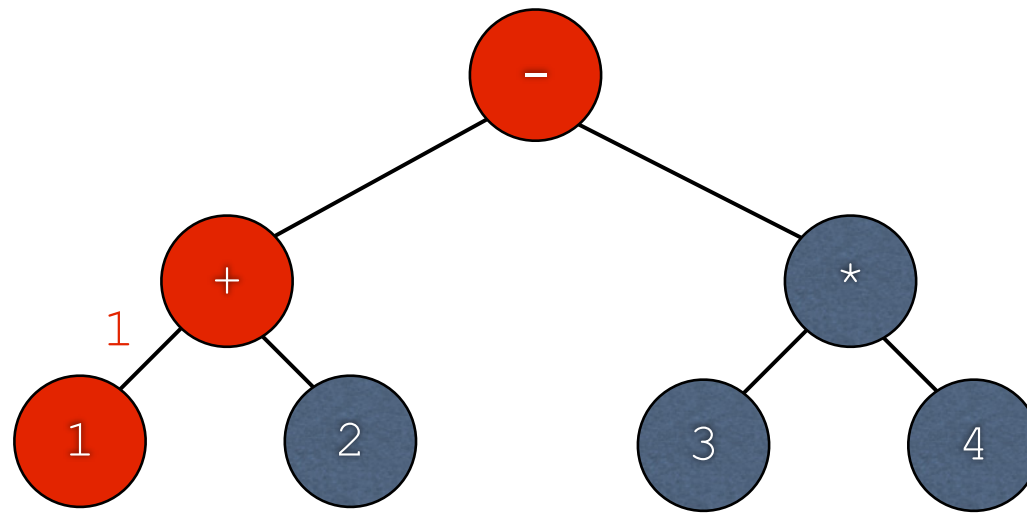
–In order to evaluate the root, we need to evaluate the left subtree of the root (+)

Evaluation



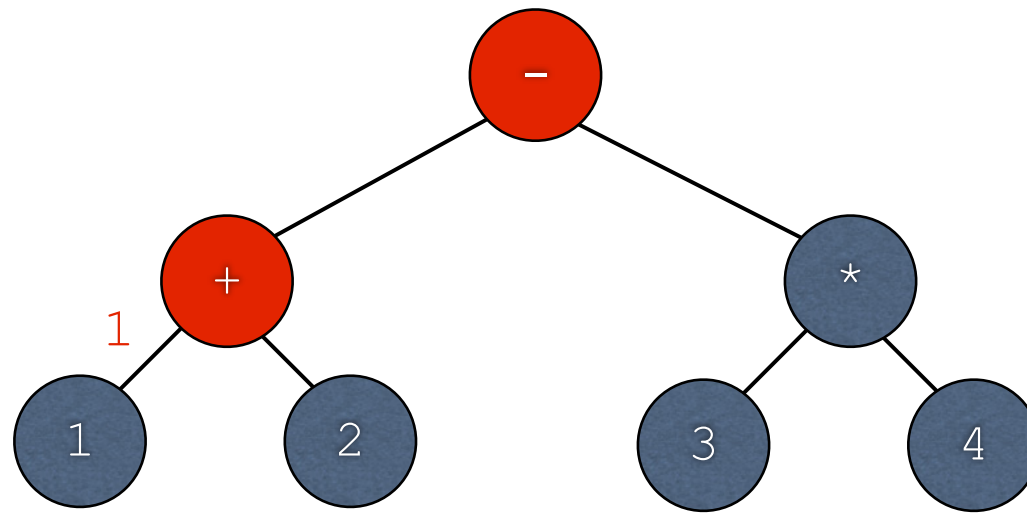
-In order to evaluate +, we need to evaluate the left subtree (as with the root)

Evaluation



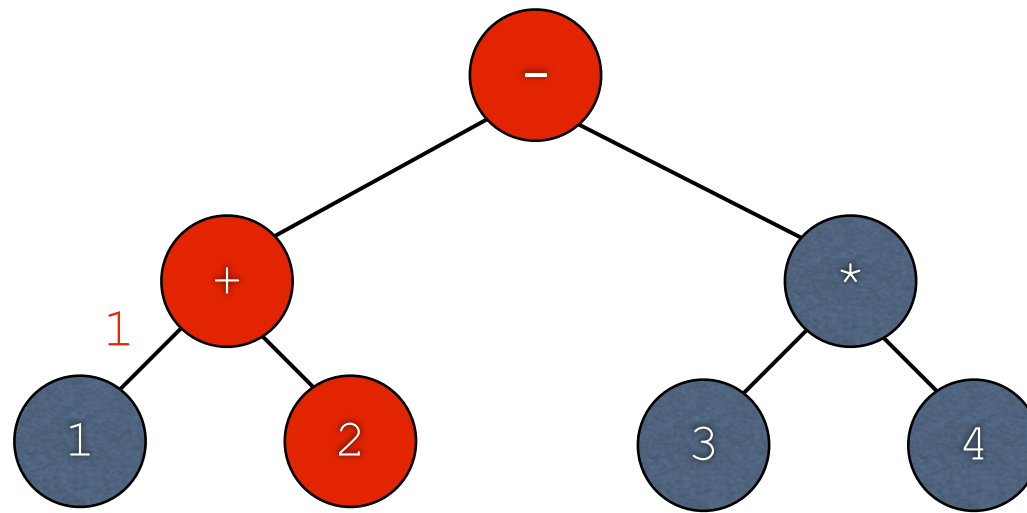
- For arithmetic, leaves are simply numbers
- Evaluating a leaf returns the number held within

Evaluation



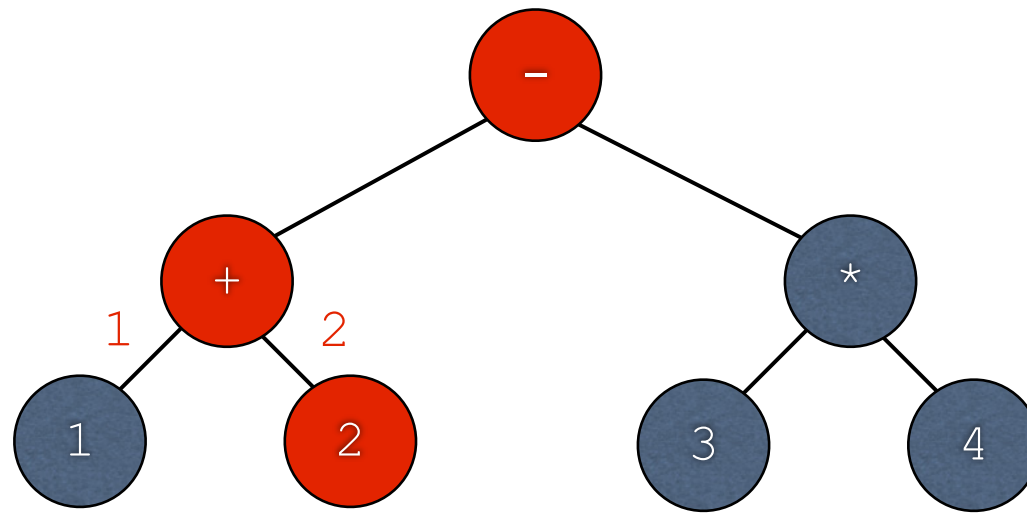
- The left subtree of + has now been evaluated
- Now + needs the value of the right subtree

Evaluation



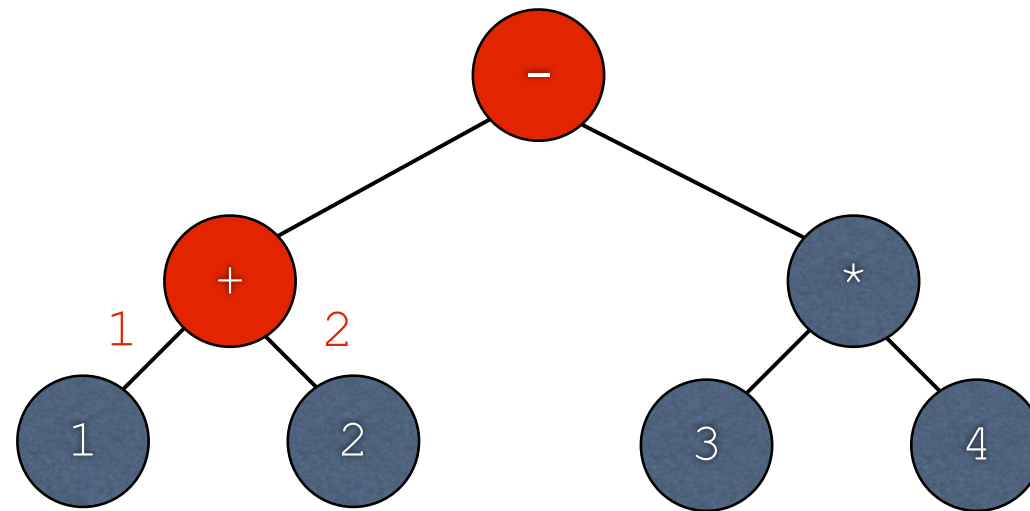
- The left subtree of + has now been evaluated
- Now + needs the value of the right subtree

Evaluation



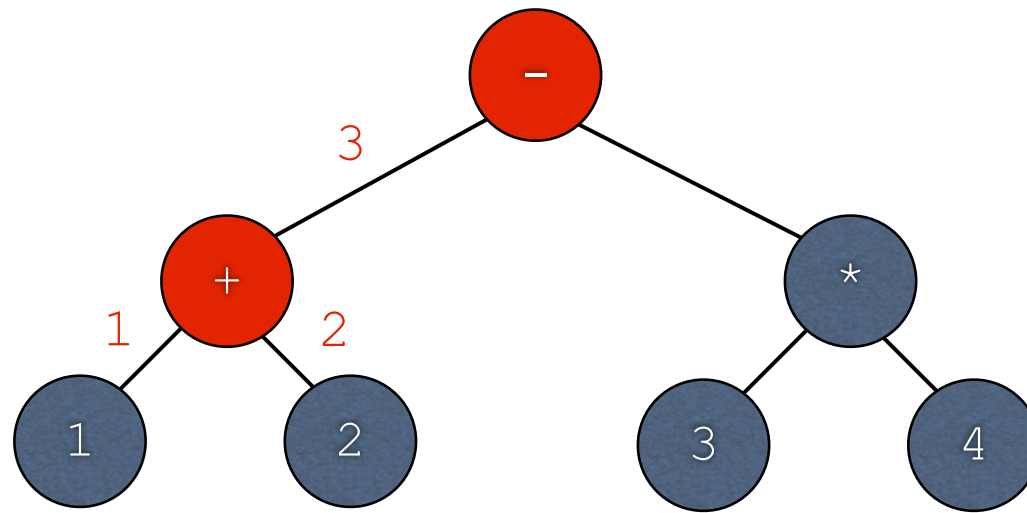
–As before, leaves just return the value held within

Evaluation



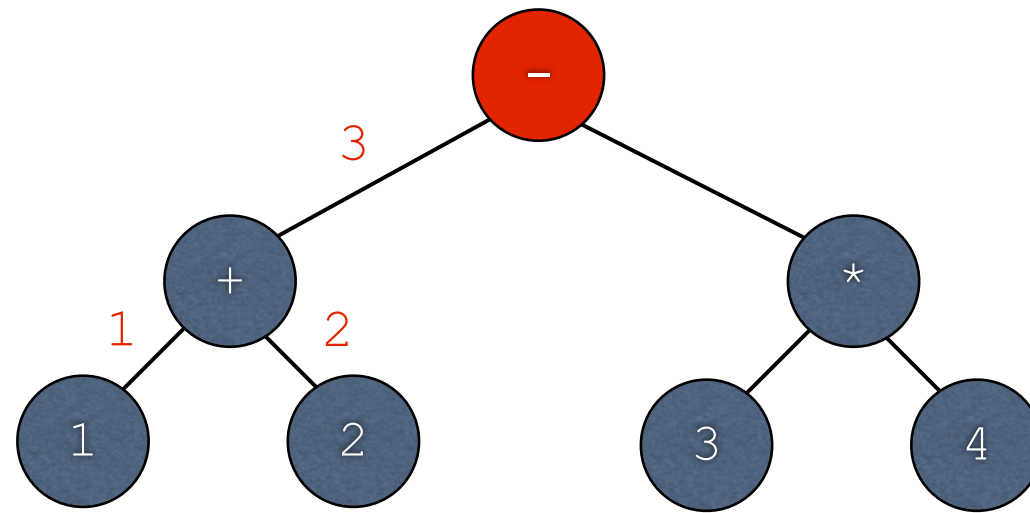
- Subtrees of + are now taken care of
- Now + has two values that it needs to work with...

Evaluation



−+ performs the actual addition

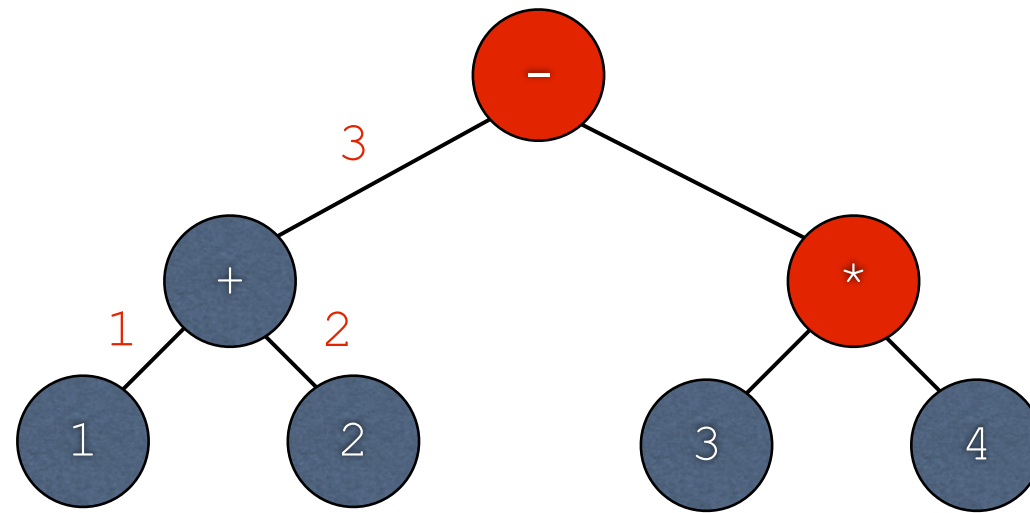
Evaluation



-Now + is taken care of

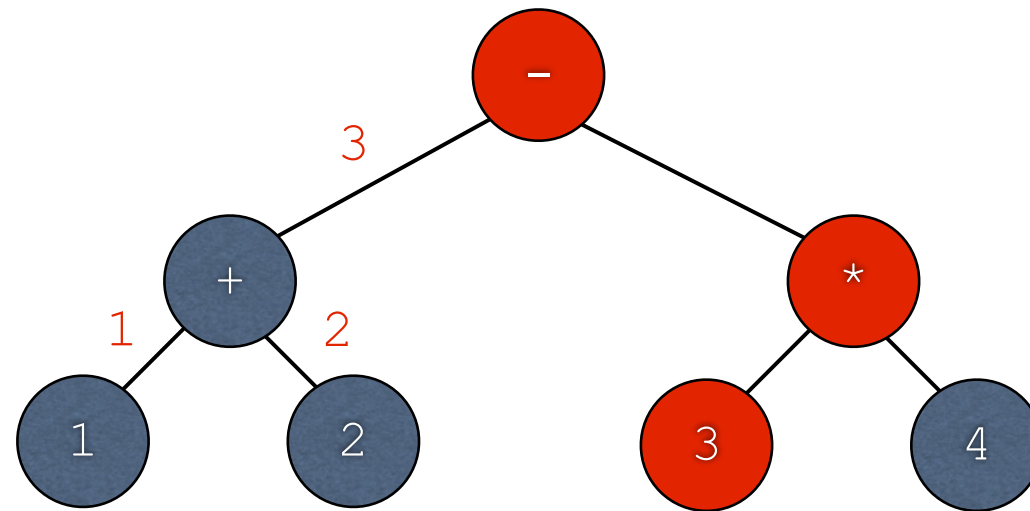
-Going back to −, − now has the value of the left subtree, and it needs the value of the right subtree

Evaluation



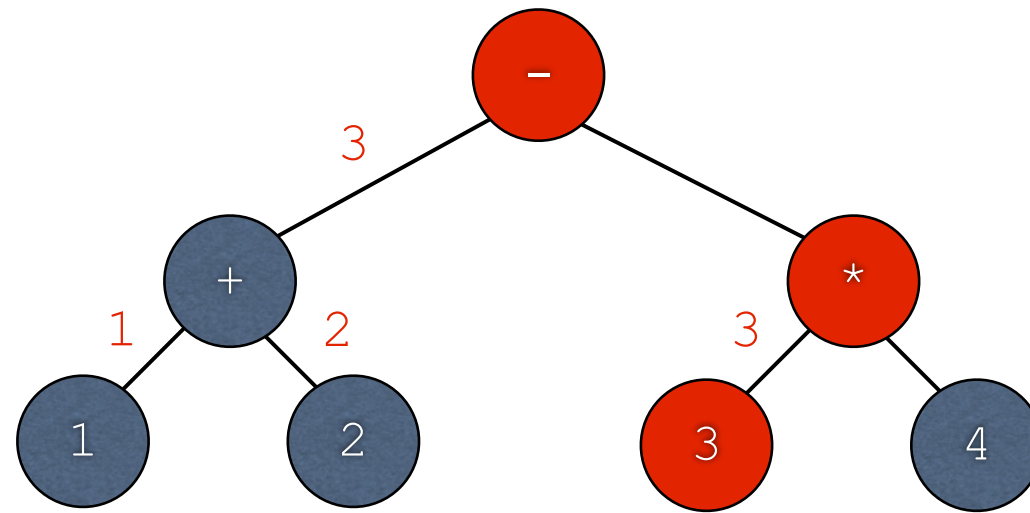
–Now we're on *, which needs the value of the left subtree...

Evaluation



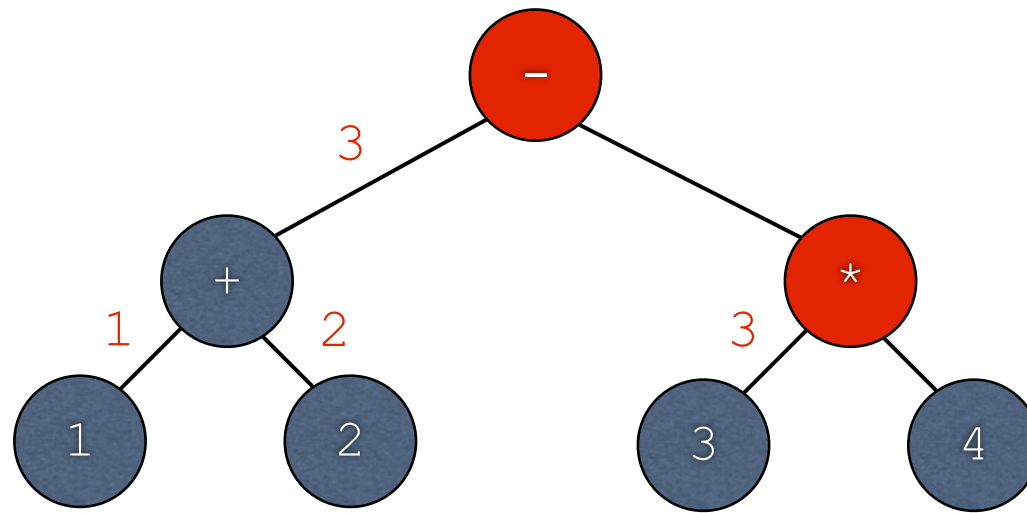
–Now we're on *, which needs the value of the left subtree...

Evaluation



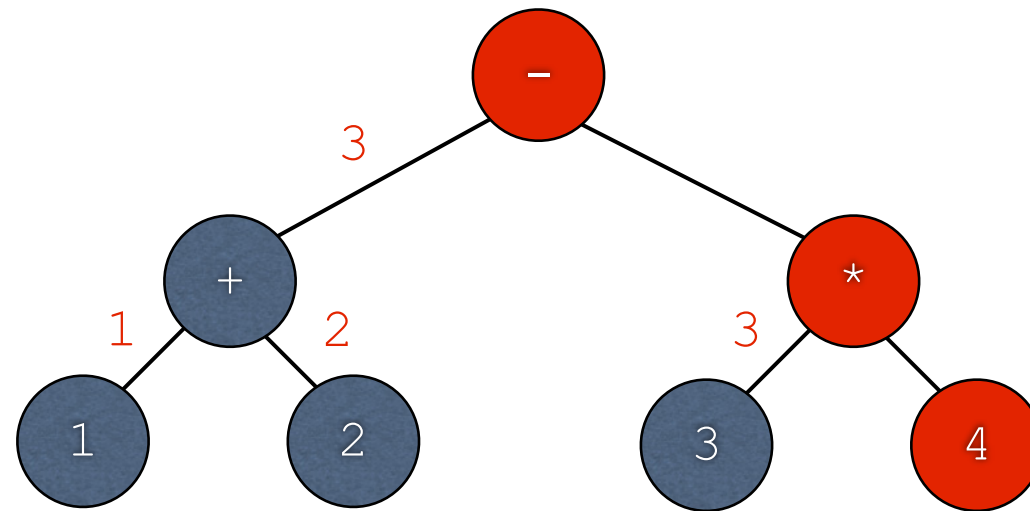
–Leaves again return the values held within...

Evaluation



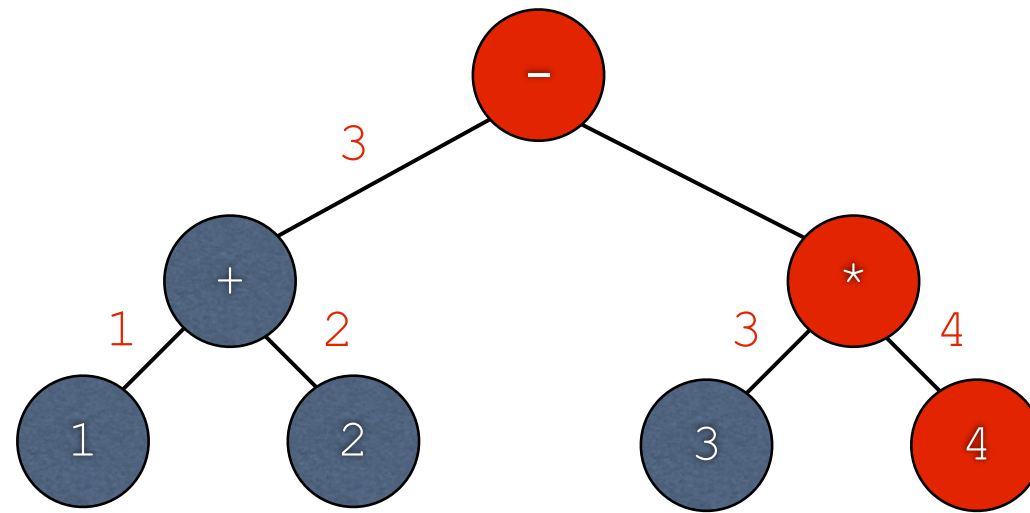
–Left subtree done; * now needs the value of the right subtree...

Evaluation



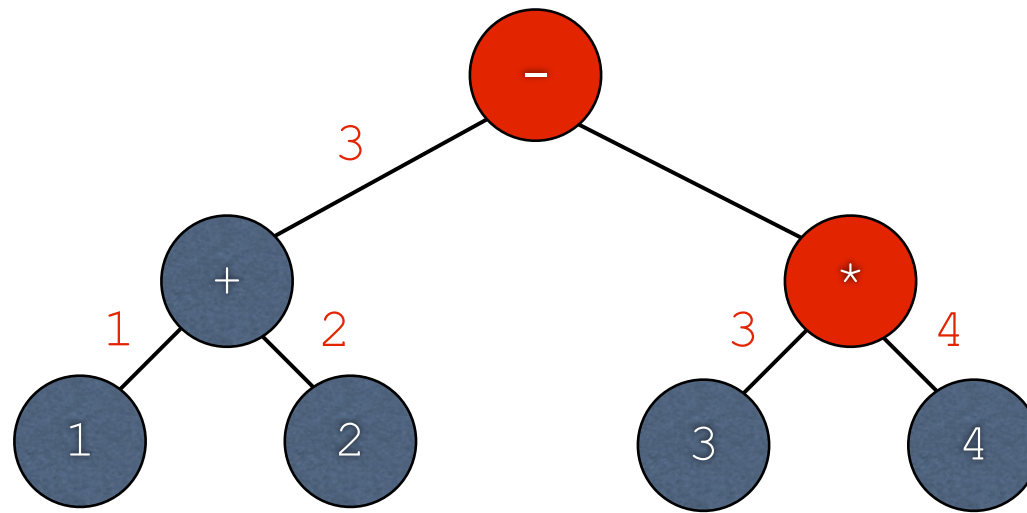
–Left subtree done; * now needs the value of the right subtree...

Evaluation



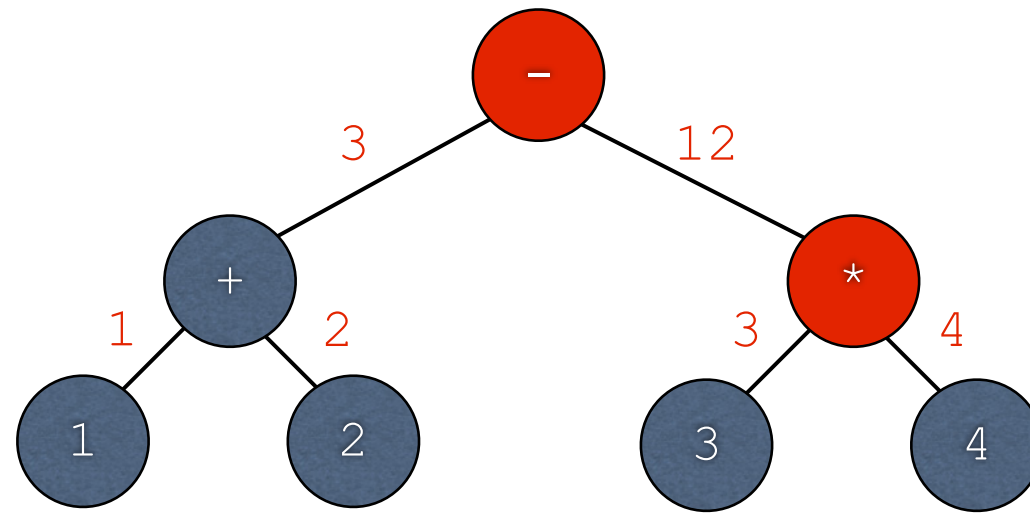
–Leaf returns value held within

Evaluation



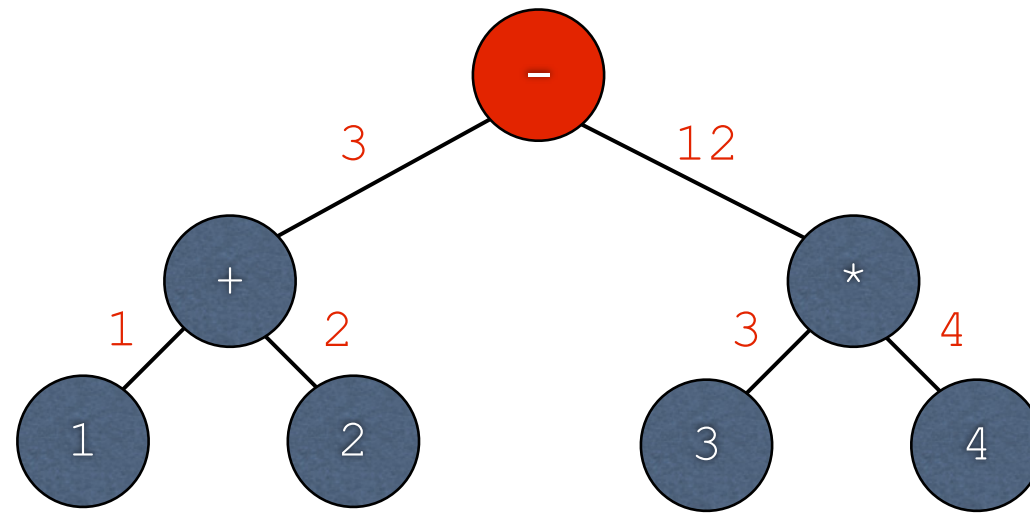
–Leaf is done. * now has both operands it needs...

Evaluation



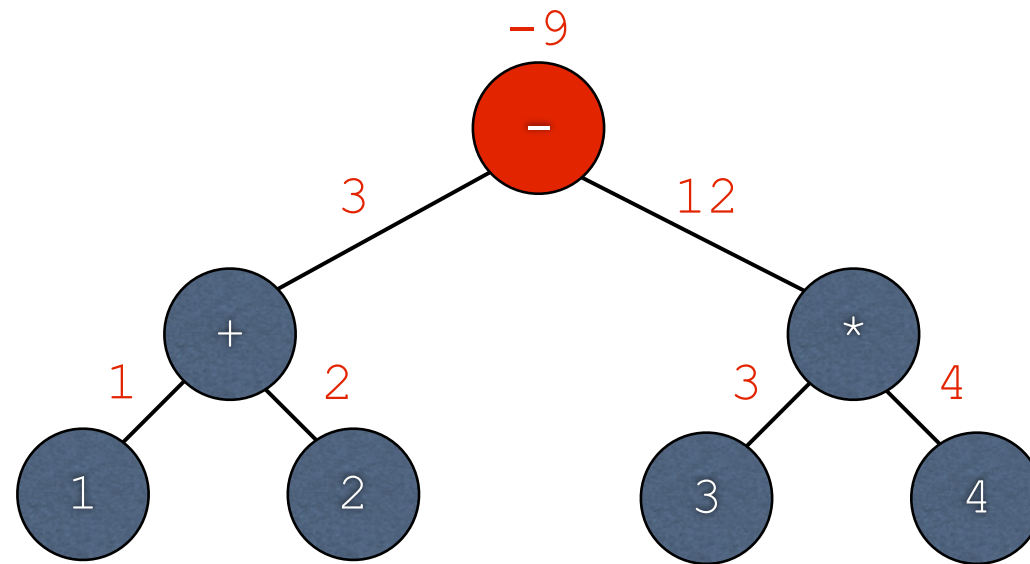
−* performs the multiplication and returns the value

Evaluation



–The root – node now has both operands...

Evaluation



–...and it returns the result of the subtraction

Exercise: Second Side of AST/Evaluation Sheet

Evaluator Example:

`arithmetic_evaluator.py`

–Complete example online; we'll live-code this in class