More on Binary Trees

Arnamoy Bhattacharyya

A1

Marks has been released

A week to submit the remark request

If you have any one line problems that made your code fail, write a FULL description of the problem and send a remark request

5% will be deducted, but worth a try if you are confident on your code

A2:

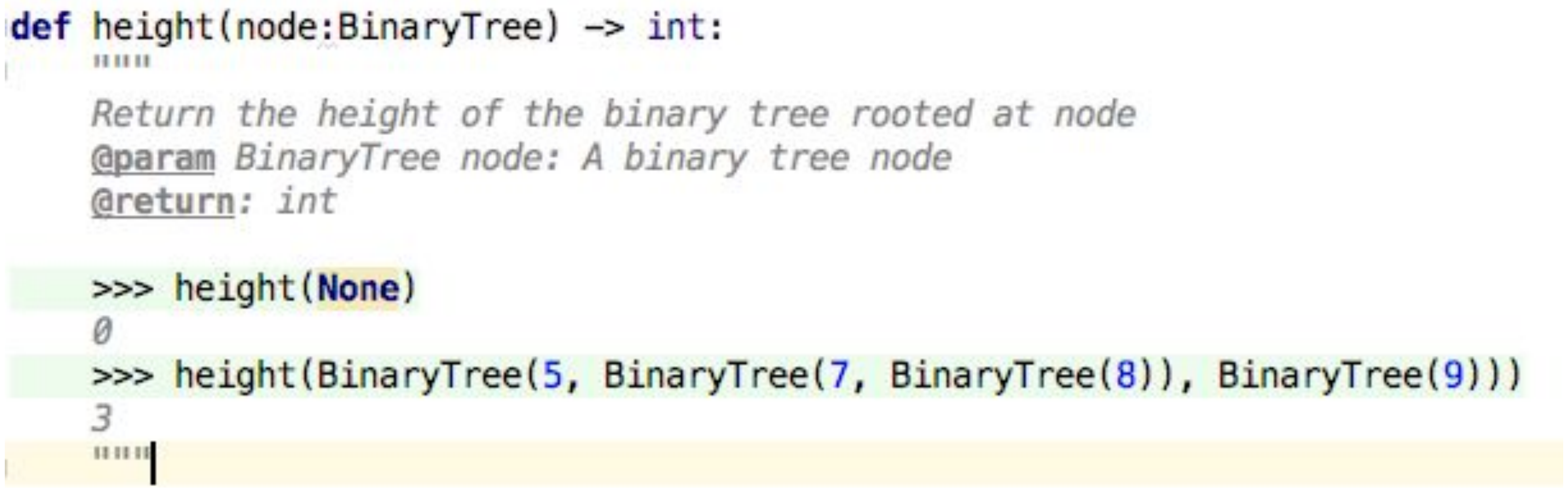
Extra hours happening, 4-8 pm today

Agenda

A few more binary tree functions (height, find)

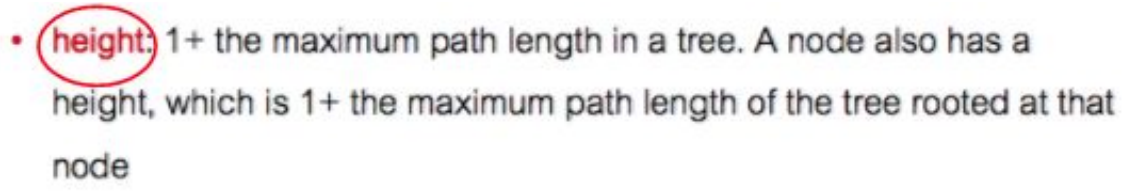
Solving arithmetic expressions as binary trees

Height of a binary tree



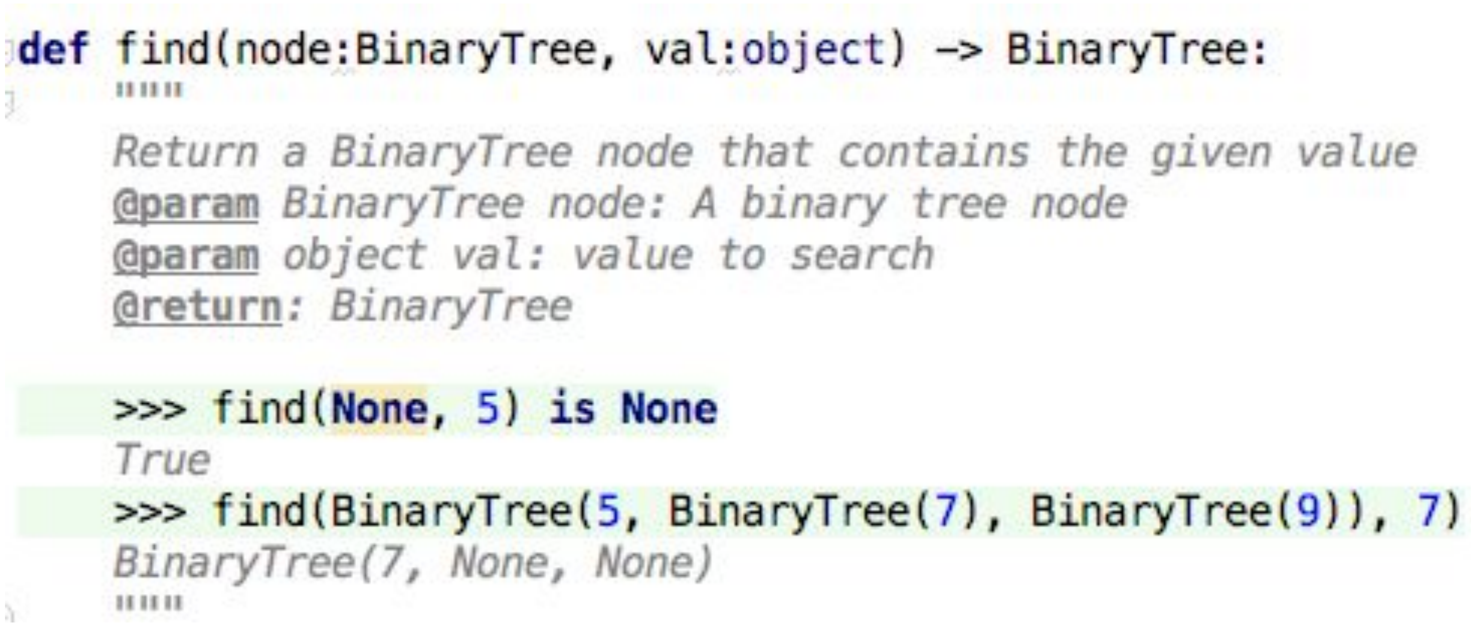
```
def height(node:BinaryTree) -> int:
    """
    Return the height of the binary tree rooted at node
    @param BinaryTree node: A binary tree node
    @return: int

    >>> height(None)
    0
    >>> height(BinaryTree(5, BinaryTree(7, BinaryTree(8)), BinaryTree(9)))
    3
    """
```



- height: 1+ the maximum path length in a tree. A node also has a height, which is 1+ the maximum path length of the tree rooted at that node

Find a value and return the node with the value

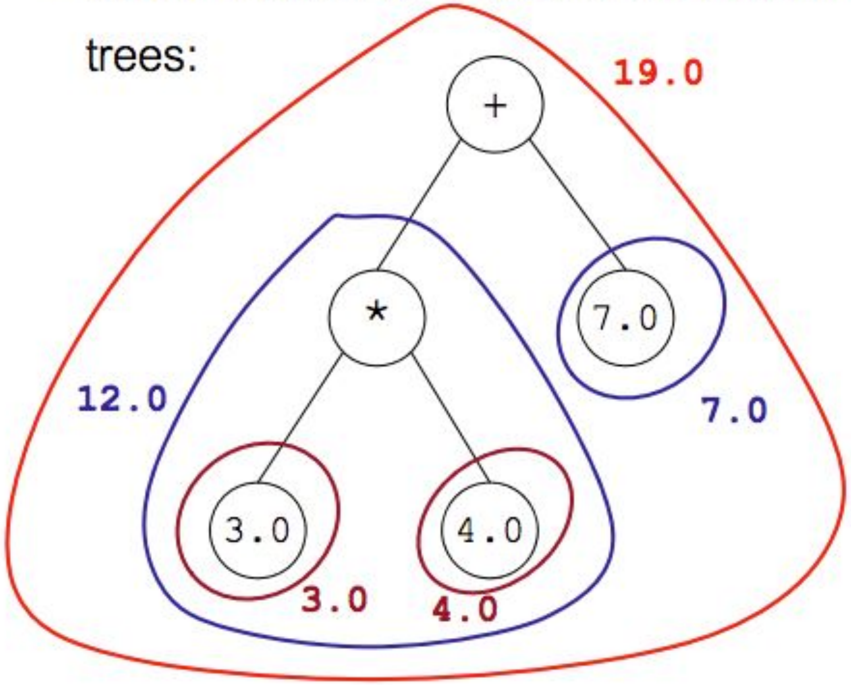


```
def find(node:BinaryTree, val:object) -> BinaryTree:
    """
    Return a BinaryTree node that contains the given value
    @param BinaryTree node: A binary tree node
    @param object val: value to search
    @return: BinaryTree

    >>> find(None, 5) is None
    True
    >>> find(BinaryTree(5, BinaryTree(7), BinaryTree(9)), 7)
    BinaryTree(7, None, None)
    """
```

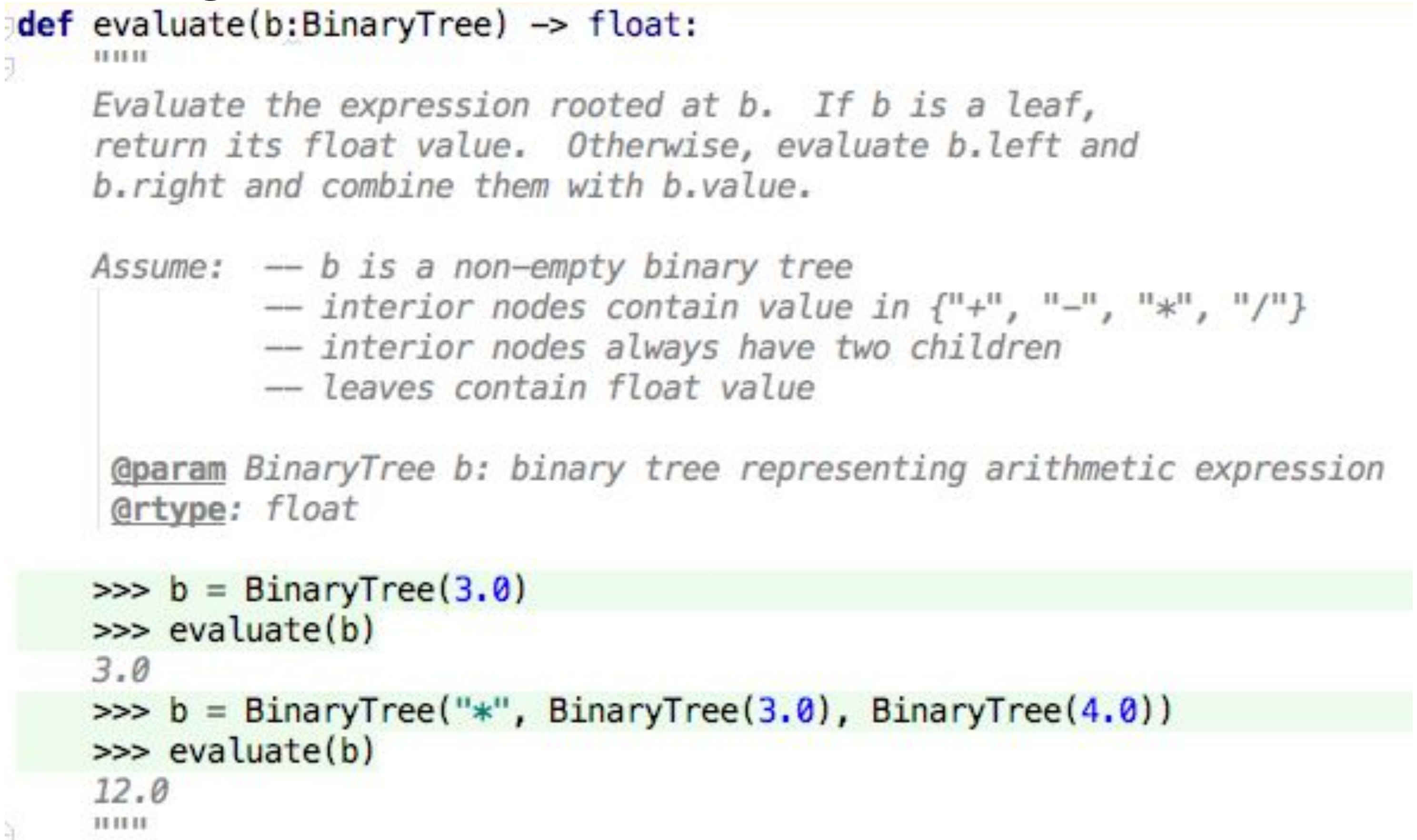
Arithmetic Expression Trees

- Binary arithmetic expressions can be represented as binary trees:



What's the strategy to evaluate an expression from a tree?

Evaluating arithmetic expressions



```
def evaluate(b:BinaryTree) -> float:
    """
    Evaluate the expression rooted at b.  If b is a leaf,
    return its float value.  Otherwise, evaluate b.left and
    b.right and combine them with b.value.

    Assume:  -- b is a non-empty binary tree
             -- interior nodes contain value in {"+", "-", "*", "/"}
             -- interior nodes always have two children
             -- leaves contain float value

     @param BinaryTree b: binary tree representing arithmetic expression
     @rtype: float

    >>> b = BinaryTree(3.0)
    >>> evaluate(b)
    3.0
    >>> b = BinaryTree("*", BinaryTree(3.0), BinaryTree(4.0))
    >>> evaluate(b)
    12.0
    """
```