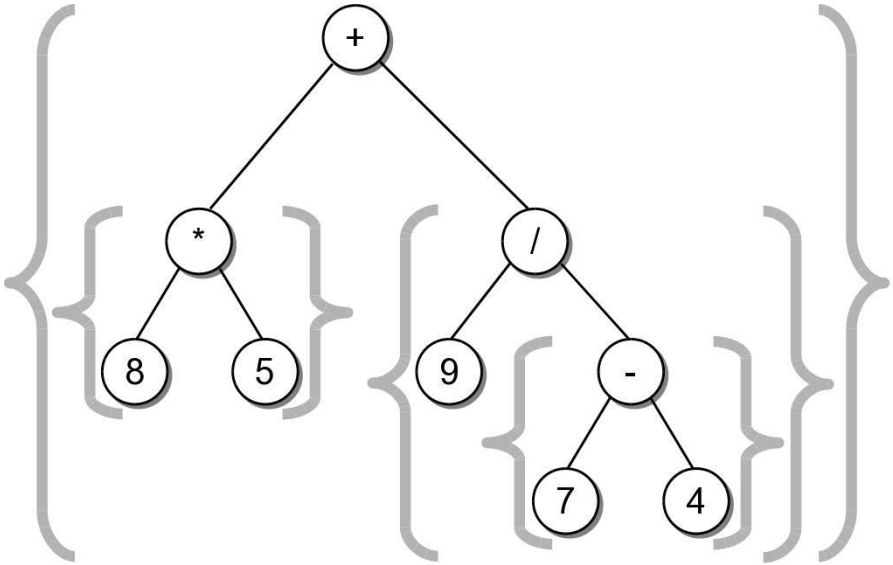
Binary Tree Application Build Expression Tree Heaps

Revised based on textbook author's notes.

String Representation

- The result was not correct because required parentheses were missing.
  - Can easily create a fully parenthesized expression.

```
((8 * 5) + (9 / (7 - 4)))
```



Class activity to implement this __str__() method.

Expression Tree Implementation

exptree.py

```
class ExpressionTree :
# ...
  def _build_string( self, tree_node ):
     # If the node is a leaf, it's an operand.
    if tree_node.left is None and tree_node.right is None :
      return str( tree_node.element )

     # Otherwise, it's an operator.
    else :
      exp_str = '('
      exp_str += self._build_string( tree_node.left )
      exp_str += str( tree_node.element )
      exp_str += self._build_string( tree_node.right )
      exp_str += ')'
      return exp_str
```

Expression Tree Construction

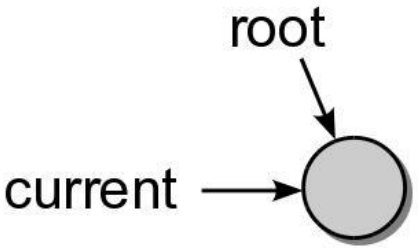
- An expression tree is constructed by parsing the expression and examining the tokens.
  - New nodes are inserted as the tokens are examined.
  - Each set of parentheses will consist of:
    - an interior node for the operator
    - two children either single valued or a subexperssion.

Expression Tree Construction

- For simplicity, we assume:
  - the expression is stored in a string with no white space.
  - the expression is valid and fully parenthesized.
  - each operand will be a single-digit or single-letter variable.
  - the operators will consist of +, -, *, /, %

Expression Tree Construction

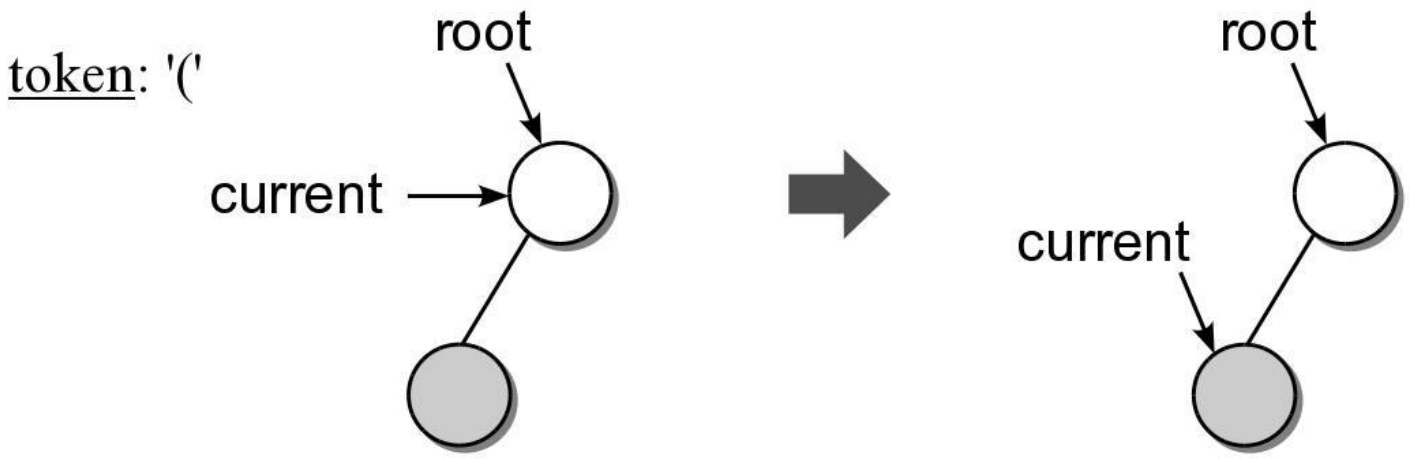
- Consider the expression `(8*5)`
- The process starts with an empty root node set as the current node:



- The action at each step depends on the current token.

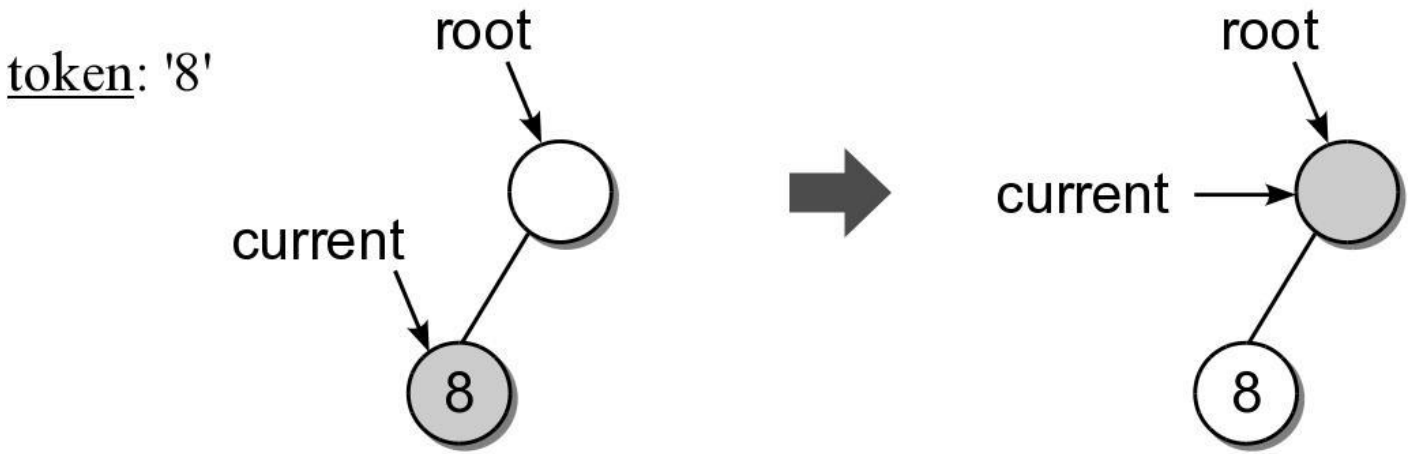
Expression Tree Construction

- When a left parenthesis is encountered:
  `(8*5)`
  - a new node is created and linked as the left child of the current node.



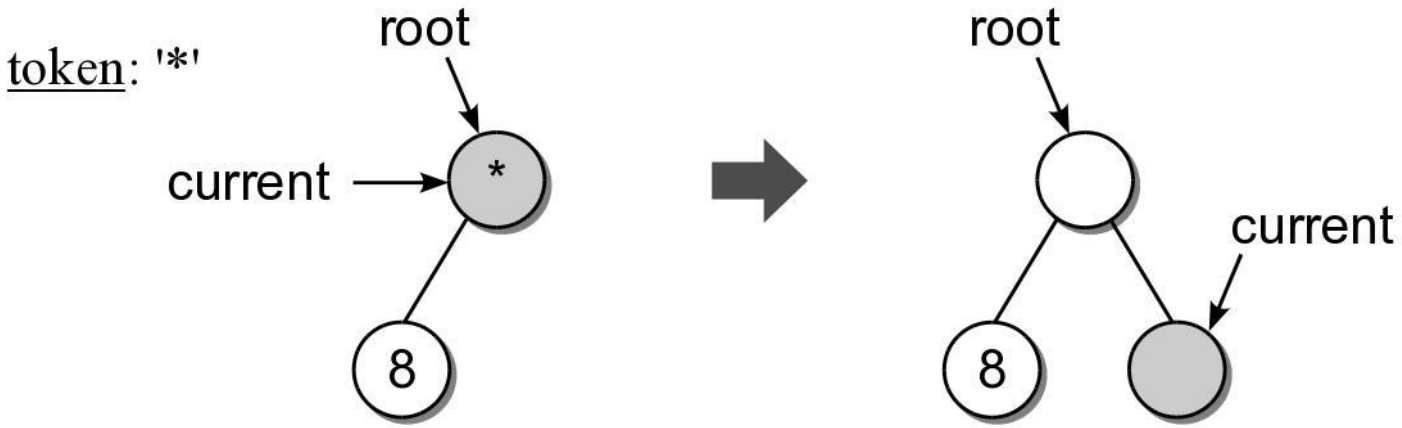
Expression Tree Construction

- When an operand is encountered:
  (**8***5)
  - the data field of the current node is set to contain the operand.



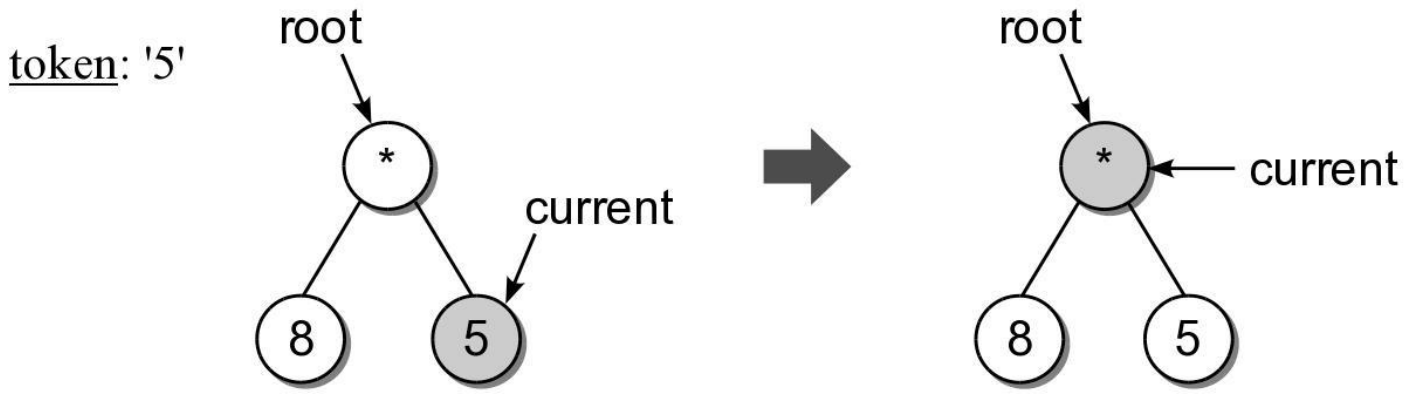
Expression Tree Construction

- When an operator is encountered:
  `(8*5)`
  - the data field of the current node is set to the operator.
  - a new node is created and linked as the right



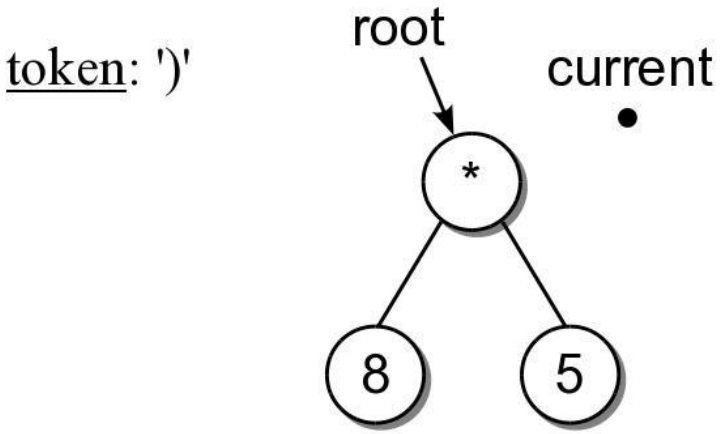
Expression Tree Construction

- Another operand is encountered: (8***5**)



Expression Tree Construction

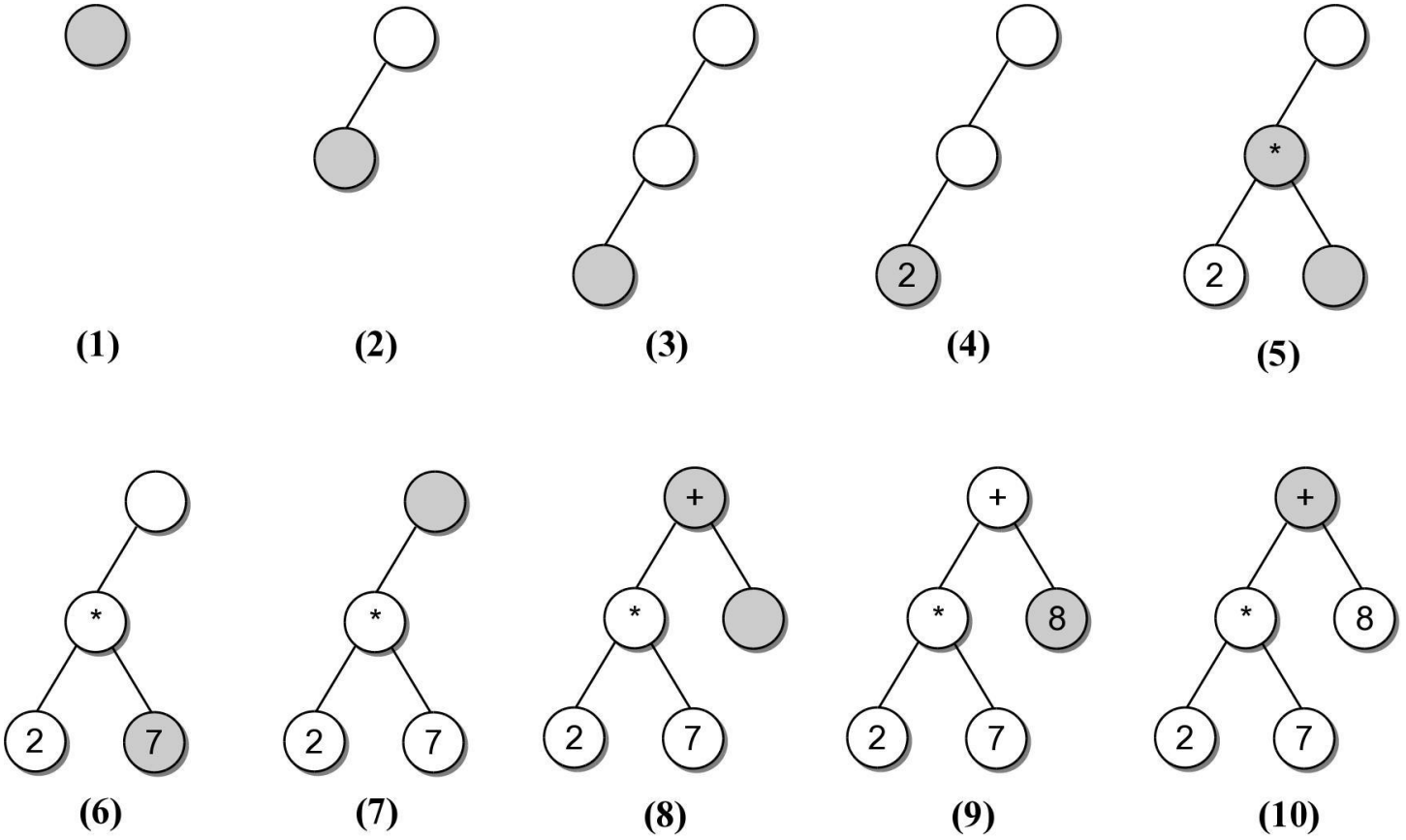
- When a right parenthesis: `(8*5)`
  - move up to the parent of the current node.



Expression Example #2

((2*7)+8)

- Consider another expression:



Expression Tree Implementation

exptree.py

```
class ExpressionTree :
# ...
  def _build_tree( self, exp_str ):
     # Build a queue containing the tokens from the expression.
    expQ = Queue()
    for token in exp_str :
      expQ.enqueue( token )

     # Create an empty root node.
    self._exp_tree = _ExpTreeNode( None )

     # Call the recursive function to build the tree.
    self._rec_build_tree( self._exp_tree, expQ )
```

Expression Tree Implementation

exptree.py

```
class ExpressionTree :
# ...
  def _rec_build_tree( self, cur_node, expQ ):
     # Extract the next token from the queue.
    token = expQ.dequeue()
     # See if the token is a left paren: '('
    if token == '(' :
      cur_node.left = _ExpTreeNode( None )
      build_treeRec( cur_node.left, expQ )
       # The next token will be an operator: + - / * %
      cur_node.data = expQ.dequeue()
      cur_node.right = _ExpTreeNode( None )
      self._build_tree_rec( cur_node.right, expQ )
       # The next token will be a ), remove it.
      expQ.dequeue()

     # Otherwise, the token is a digit.
    else :
      cur_node.element = token
```

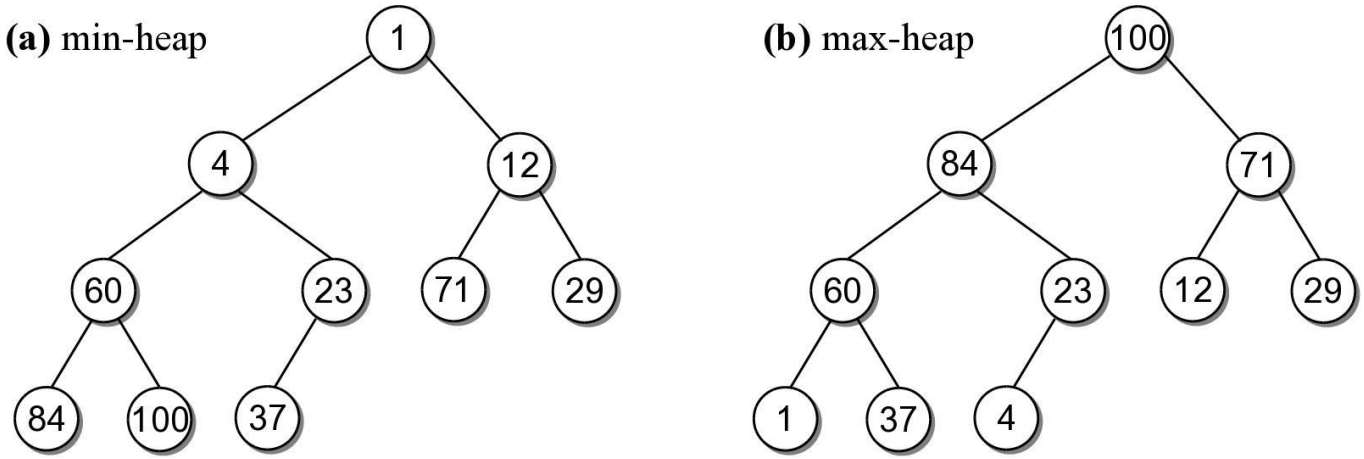
Run testexptree.py in 28_ExpressionTree/

Heaps

- A heap is a complete binary tree in which the nodes are organized based on their data values.
- **heap order property** – how the nodes in a heap or arranged.
- **heap shape property** – as a complete binary tree.

Heap property, examples

- For each non-leaf node V,
  - **max-heap**: the value in V is greater than the value of its two children.
  - **min-heap**: the value in V is smaller than the value of its two children.



Heap Operations

- The heap is a specialized structure with limited operations.
  - Insert an element into the heap.
  - Remove the element from root node.