## Implementing Tree Traversals in Python

→ **Preorder Traversal:**

- must be paramaterized by a specific position within the tree that serves as the root of a subtree to traverse
- standard solution is to define a non-public utility method with desired recursion and then have the public method invoke the nonpublic method on root of tree

```
def preorder(self):
  if not self.is_empty()
    for p in self._subtree_preorder(self.root())      // start recursion
      yield p

def _subtree-preorder(self, p):
 // generate preorder iteration of positions in subtree rooted at p
    yield p                                          // visit p before subtrees
    for c in self.children(p)                        // for each child
       for other in self._subtree_preorder(c)        // do preorder for c subtree
          yield other
```

→ **Postorder Traversal**

- similar to preorder, but we wait to yield p until after recursively yielding subtrees

```
def postorder(self):
   if not empty
     for p in self._subtree_postorder(self.root()) // start recursion
       yield p

def _subtree_postorder(self, p):
   for c in self.children(p):
      for other in self._subtree_postorder(c)    // do postorder of c's subtree
           yield other                           // yielding to caller
   yield p                                       // visit p after subtrees
```