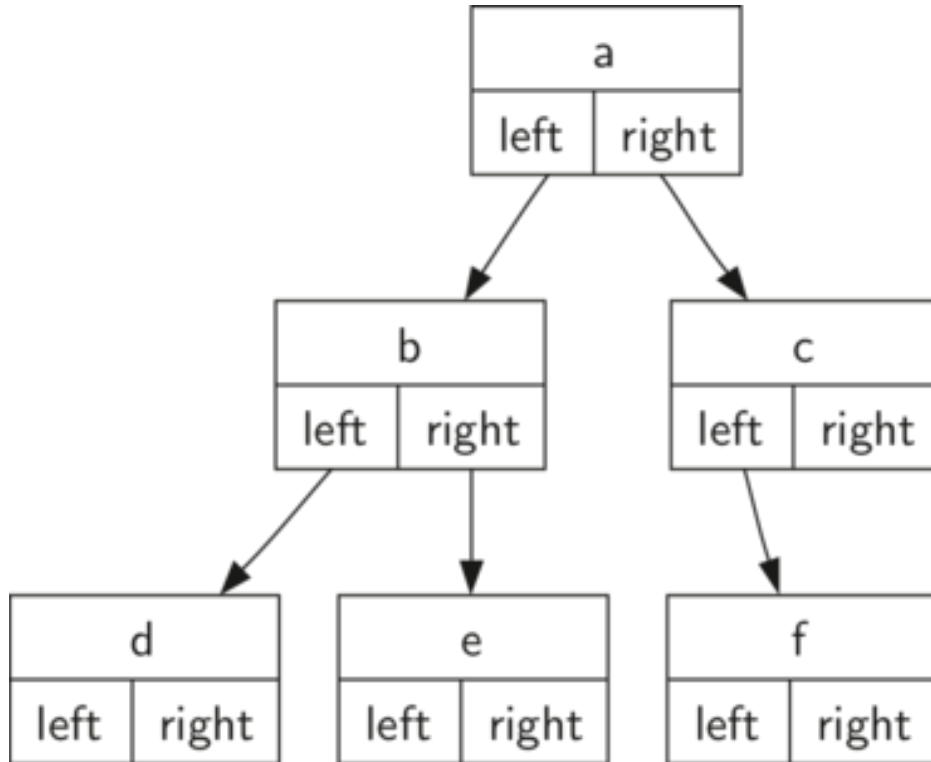


ปมและการอ้างอิง  
(nodes and references)

root และ subtree ช่าย ขวา



```
class BinaryTree:
```

```
    def __init__(self, rootObj):
```

```
        self.key = rootObj
```

```
        self.leftChild = None
```

```
        self.rightChild = None
```

# insertLeft และ insertRight

```
def insertLeft(self, newNode):  
    if self.leftChild == None:  
        self.leftChild = BinaryTree(newNode)  
    else:  
        t = BinaryTree(newNode)  
        t.leftChild = self.leftChild  
        self.leftChild = t
```

```
def insertRight(self, newNode):  
    if self.rightChild == None:  
        self.rightChild = BinaryTree(newNode)  
    else:  
        t = BinaryTree(newNode)  
        t.rightChild = self.rightChild  
        self.rightChild = t
```

อธิบายเป็นภาพการทำงานได้อย่างไร?

# method อื่นๆ

```
def getRightChild(self):  
    return self.rightChild
```

```
def getLeftChild(self):  
    return self.leftChild
```

```
def setRootVal(self,obj):  
    self.key = obj
```

```
def getRootVal(self):  
    return self.key
```

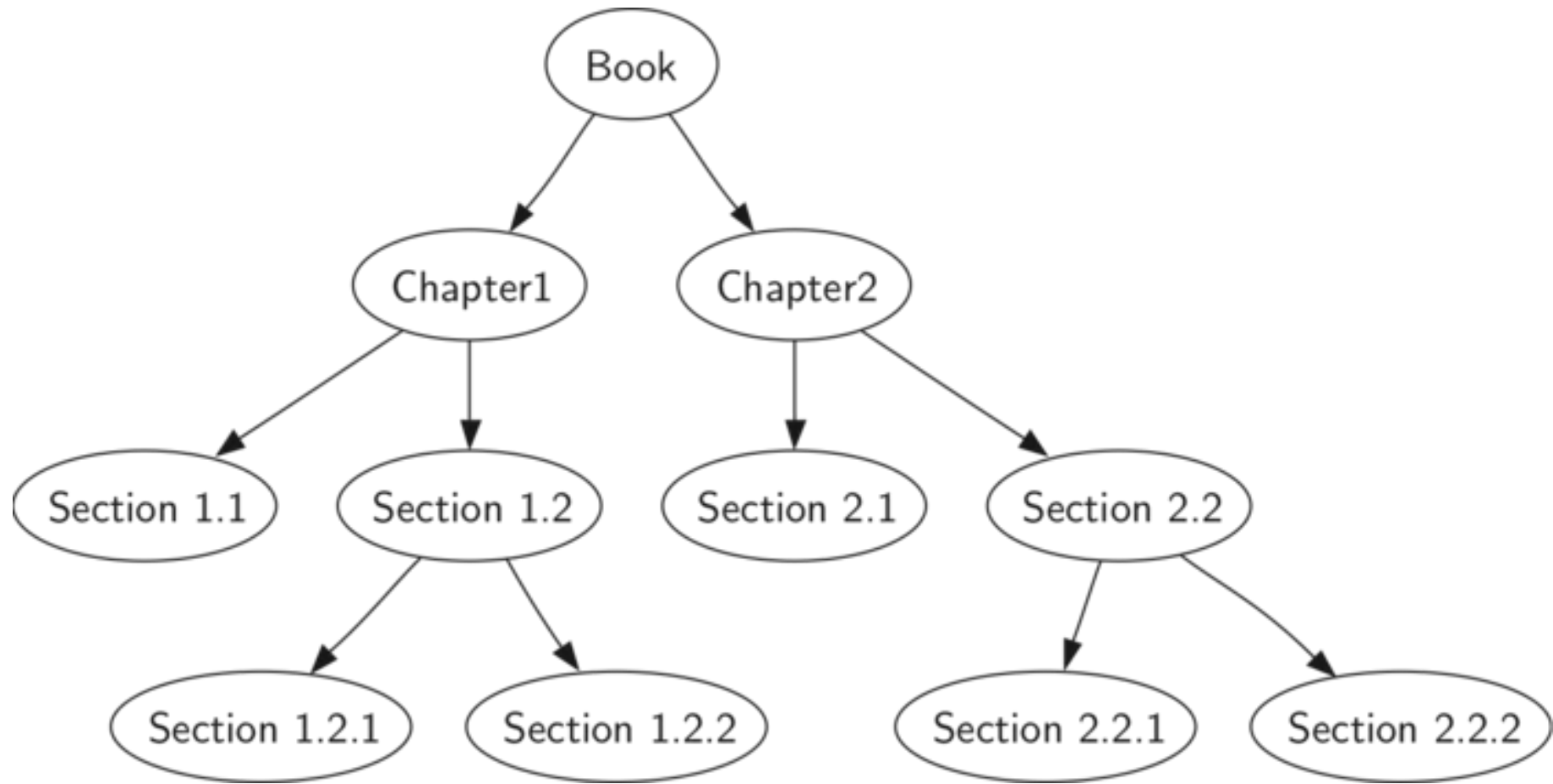
```
1 class BinaryTree:
2     def __init__(self, rootObj):
3         self.key = rootObj
4         self.leftChild = None
5         self.rightChild = None
6
7     def insertLeft(self, newNode):
8         if self.leftChild == None:
9             self.leftChild = BinaryTree(newNode)
10        else:
11            t = BinaryTree(newNode)
12            t.leftChild = self.leftChild
13            self.leftChild = t
14
15    def insertRight(self, newNode):
16        if self.rightChild == None:
17            self.rightChild = BinaryTree(newNode)
18        else:
19            t = BinaryTree(newNode)
20            t.rightChild = self.rightChild
21            self.rightChild = t
22
23
```

```
24 def getRightChild(self):
25     return self.rightChild
26
27 def getLeftChild(self):
28     return self.leftChild
29
30 def setRootVal(self, obj):
31     self.key = obj
32
33 def getRootVal(self):
34     return self.key
35
36
37 r = BinaryTree('a')
38 print(r.getRootVal())
39 print(r.getLeftChild())
40 r.insertLeft('b')
41 print(r.getLeftChild())
42 print(r.getLeftChild().getRootVal())
43 r.insertRight('c')
44 print(r.getRightChild())
45 print(r.getRightChild().getRootVal())
46 r.getRightChild().setRootVal('hello')
47 print(r.getRightChild().getRootVal())
48
```

# การเดินทางในต้นไม้ (tree traversal)

## การเดินทาง 3 แบบ

- preorder ลำดับก่อน
  - root  $\rightarrow$  left subtree  $\rightarrow$  right subtree
- postorder ลำดับหลัง
  - left subtree  $\rightarrow$  right subtree  $\rightarrow$  root
- inorder ในลำดับ
  - left subtree  $\rightarrow$  root  $\rightarrow$  right subtree



ควรจะเดินทางแบบไหน?



# preorder

```
def preorder(tree):  
    if tree:  
        print(tree.getRootVal())  
        preorder(tree.getLeftChild())  
        preorder(tree.getRightChild())
```

แบบ function แยก

```
def preorder(self):  
    print(self.key)  
    if self.leftChild:  
        self.leftChild.preorder()  
    if self.rightChild:  
        self.rightChild.preorder()
```

แบบ method

```
def postorder(tree):
```

```
    if tree != None:
```

```
        postorder(tree.getLeftChild())
```

```
        postorder(tree.getRightChild())
```

```
    print(tree.getRootVal())
```

# postorder

```
def postordereval(tree):
```

```
   opers = {'+':operator.add, '-':operator.sub, '*':operator.mul, '/':operator.truediv}
```

```
    res1 = None
```

```
    res2 = None
```

```
    if tree:
```

```
        res1 = postordereval(tree.getLeftChild())
```

```
        res2 = postordereval(tree.getRightChild())
```

```
        if res1 and res2:
```

```
            return opers[tree.getRootVal()](res1,res2)
```

```
        else:
```

```
            return tree.getRootVal()
```

# inorder

```
def inorder(tree):  
    if tree != None:  
        inorder(tree.getLeftChild())  
        print(tree.getRootVal())  
        inorder(tree.getRightChild())
```

```
def printexp(tree):  
    sVal = ""  
    if tree:  
        sVal = '(' + printexp(tree.getLeftChild())  
        sVal = sVal + str(tree.getRootVal())  
        sVal = sVal + printexp(tree.getRightChild())+')'  
    return sVal
```