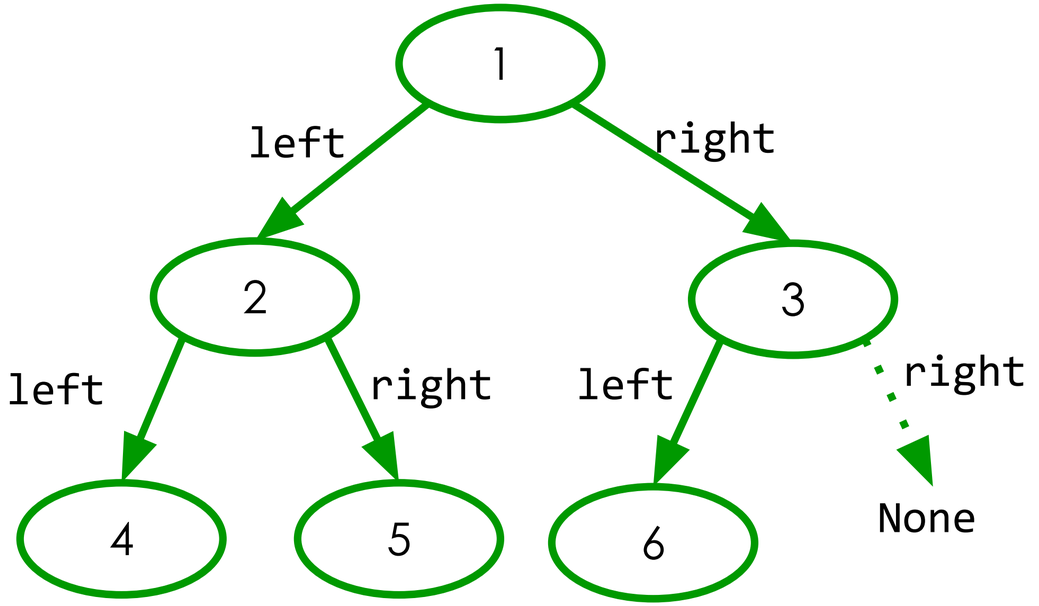
③ 二分木の実現方法

ここでは、Pythonで二分木を実装する方法を学習します。

二分木の実現方法

- 二分木を実装するには、リンクリストと同様に、各ノードから他のノードへのリンク（メモリアドレスなど）をもたせます
- 二分木とするために、以下のノードへのポイントを持たせる方法が一般的です
  - 左の子ノード
  - 右の子ノード
  - ※親ノードへのリンクを持たせることもあります
- 子ノードがない場合はNoneとします



Pythonで表現すると…

- Python で二分木を表現すると、以下のようになります
  - まずは簡単な木を作ります

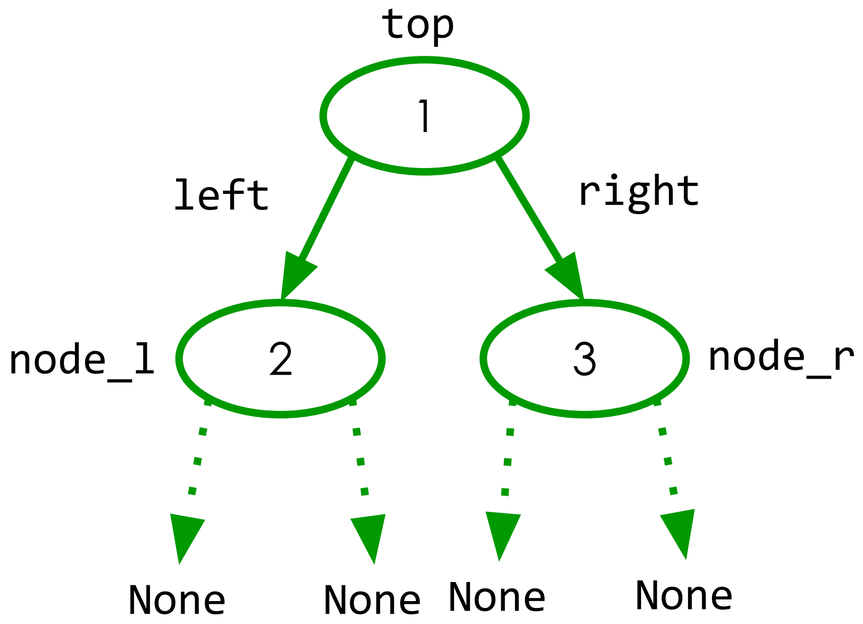
```
class Node:
    def __init__(self, value):
        self.value = value
        self.left = None
        self.right = None

    def __str__(self):
        return str(self.value)
```

value : このノードの格納する要素
left : 左の子ノード
right : 右の子ノード

str に型変換した時はノードの値とする

```
top = Node(1)
node_l = Node(2)
node_r = Node(3)
top.left = node_l
top.right = node_r
```

右のような木構造

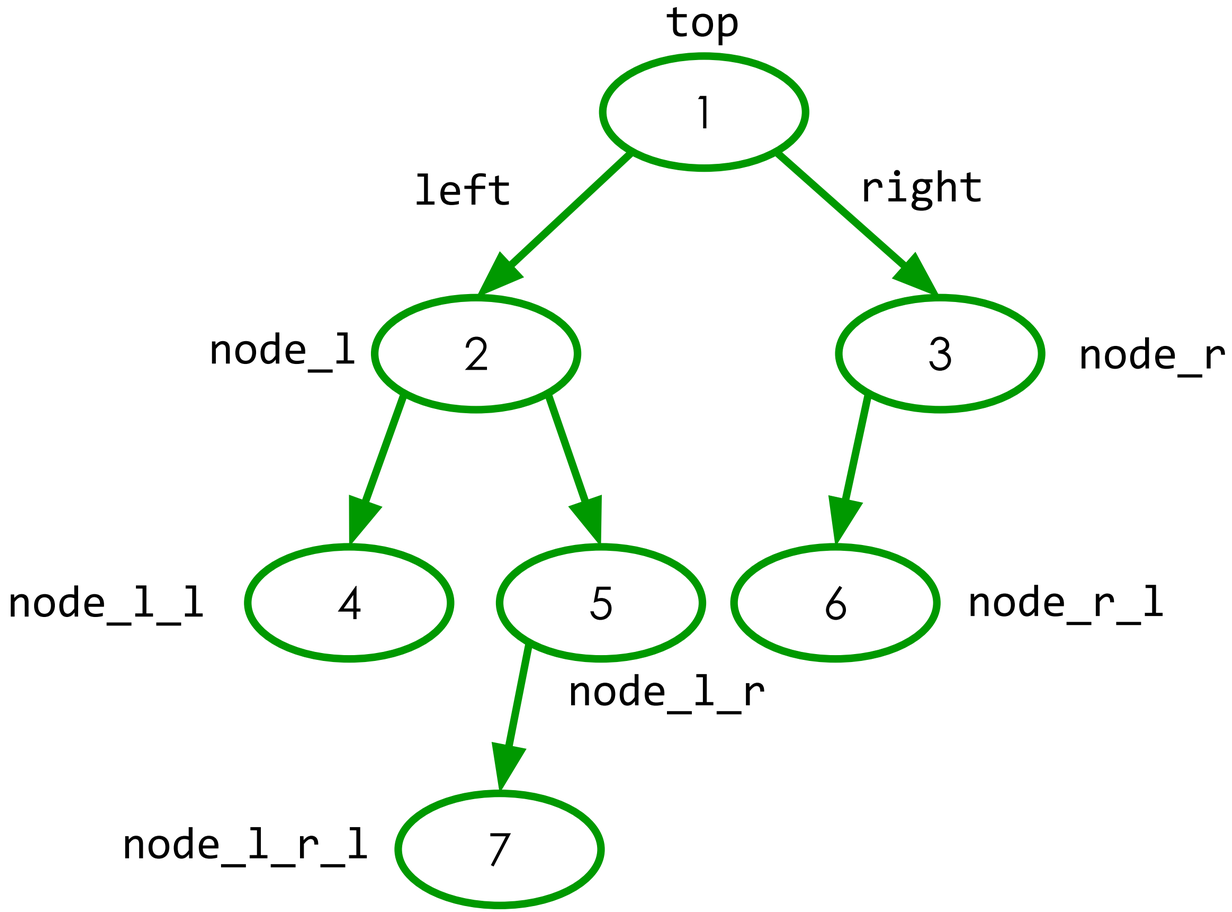


Pythonで表現すると…

- 以下のようにすると、どういう木構造になるかわかりますか？

```
top = Node(1)
node_l = Node(2)
node_r = Node(3)
top.left = node_l
top.right = node_r
```

```
node_l_l = Node(4)
node_l_r = Node(5)
node_l.left = node_l_l
node_l.right = node_l_r

node_r_l = Node(6)
node_r.left = node_r_l

node_l_r_l = Node(7)
node_l_r.left = node_l_r_l
```



Pythonで表現すると…

- このようにすることで、 left および right を辿ることで、根のノードから任意のノードにたどり着くことができます

```
node = top.left
print(node)
```

```
2
```

```
node = top.right
print(node)
```

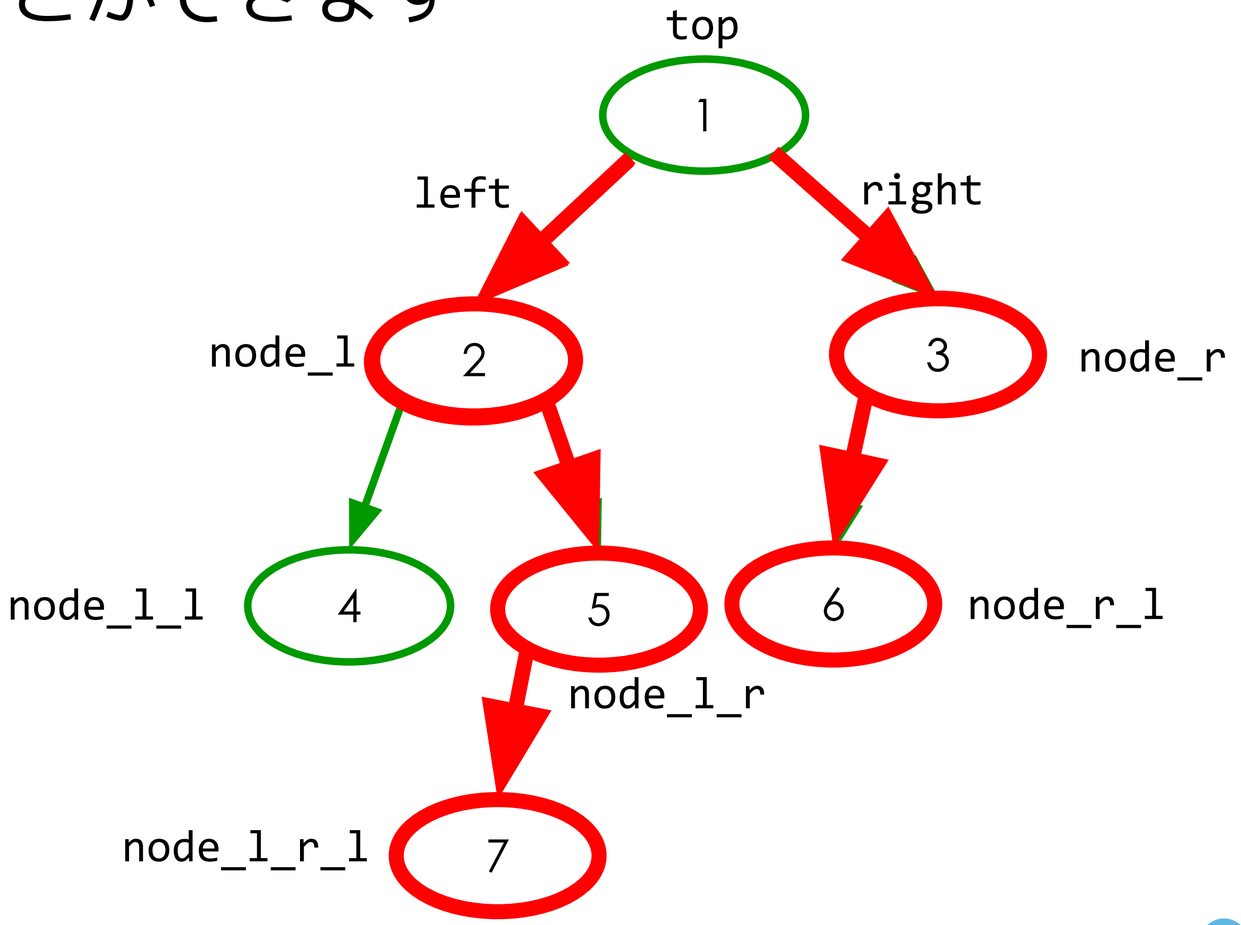
```
3
```

```
node = top.right.left
print(node)
```

```
6
```

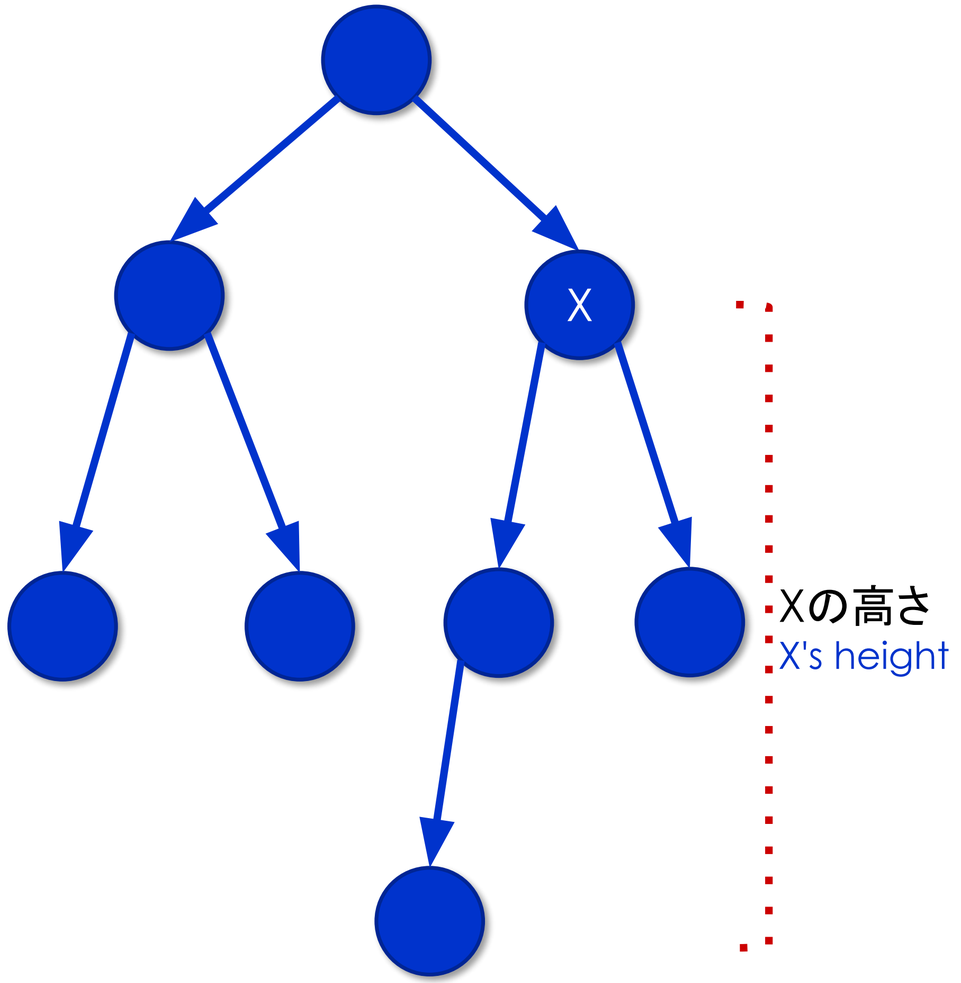
```
node = top.left.right.left
print(node)
```

```
7
```



二分木のノードの高さを求めるには？

- あるノードの高さを求めるには、どのようにすれば良いでしょうか？
  - 簡単そうで、簡単ではありません
- 次のように考え、再帰呼出しを用いて定義します
  - ノードが葉の場合、高さは０
  - 左右の子の高さの大きい方+1が、ノードの高さとなる



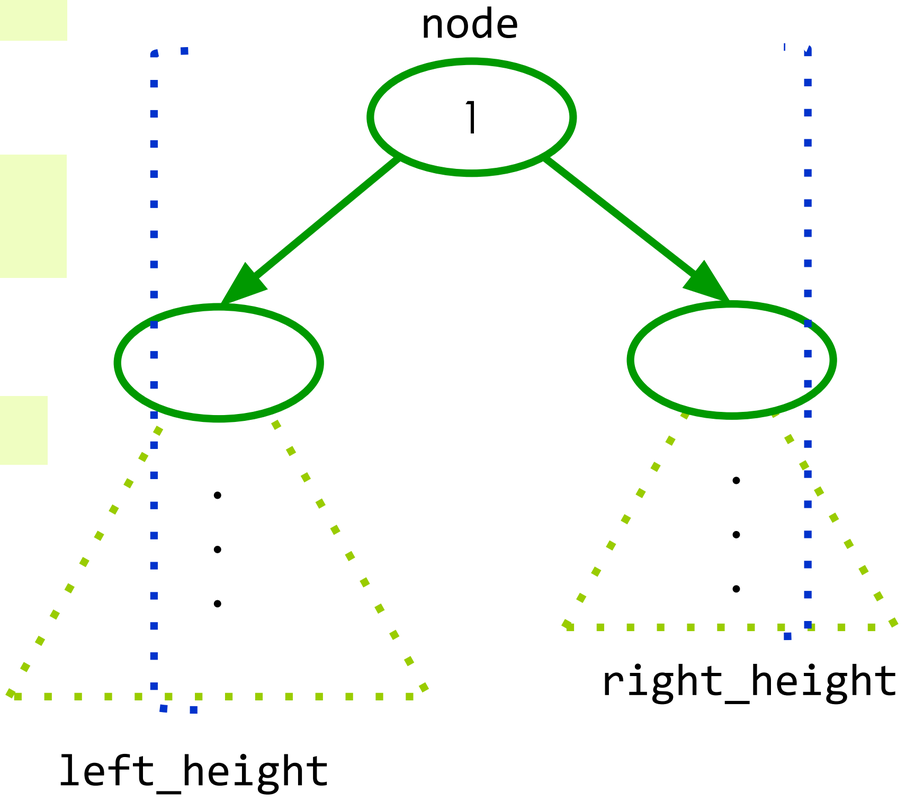
Pythonで表現すると…

```
def height(node):
    if node.left != None:
        left_height = height(node.left) + 1
    else:
        left_height = 0

    if node.right != None:
        right_height = height(node.right) + 1
    else:
        right_height = 0

    return max(left_height, right_height)
```

左側の高さを求める
ない場合は0

右側の高さを求める
ない場合は0

その大きい方




```
height(top)
```

```
3
```

```
height(node_l)
```

```
2
```