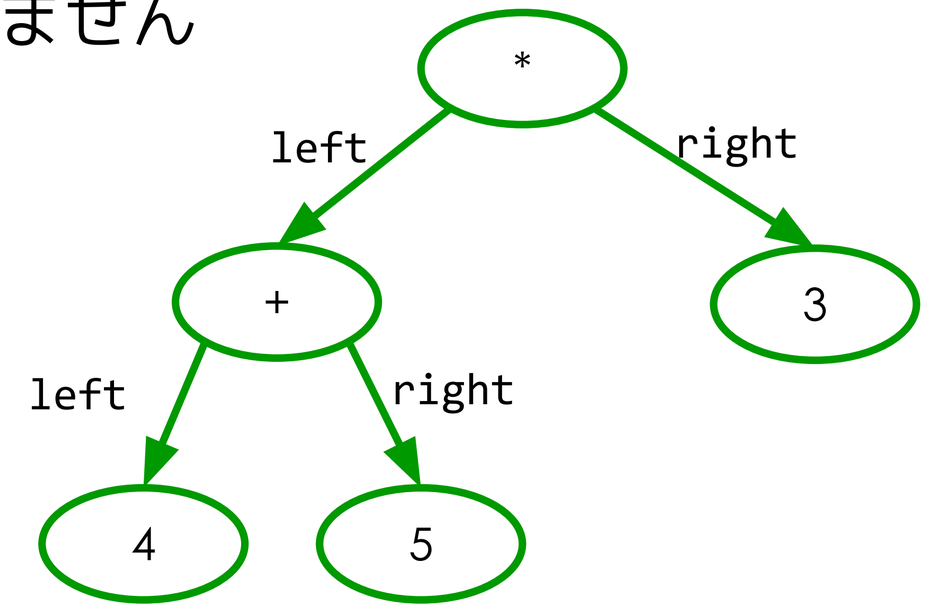
③ 数式の木の実装

最後に、数式の木をPythonで実装してみましょう。

数式の木の実現方法

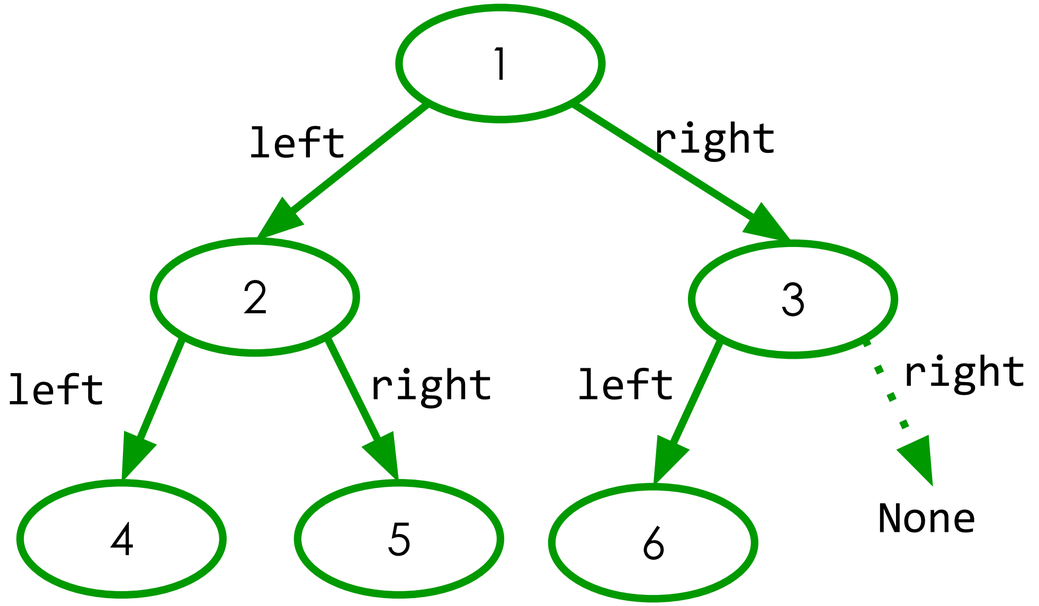
- 前回学習した二分木と同様の方法で、数式の木を実装することができます
- 各ノードについて、以下のようにします
  - 演算子の場合は、必ず2つの子ノードへのリンクを持たせます
  - 数値ノードの場合は、子ノードを持ちません

```
(4 + 5) * 3
```



（復習）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

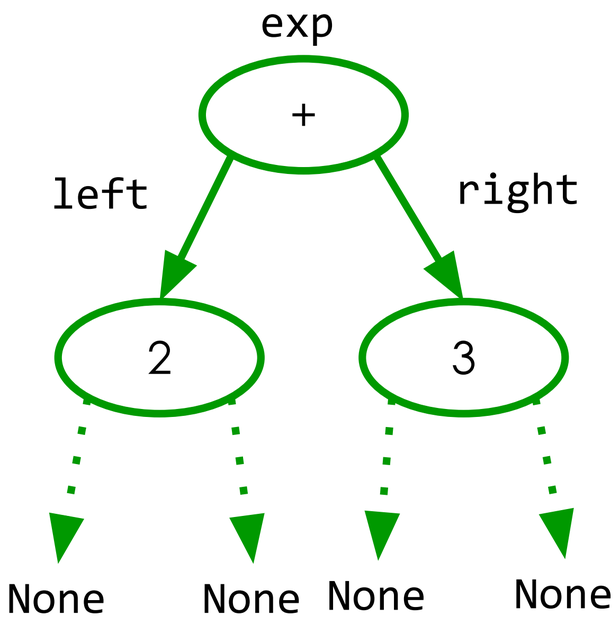
```
class Node:
    def __init__(self, value):
        self.value = value
        self.left = None
        self.right = None

    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

```
exp = Node('+')
exp.left = Node(2)
exp.right = Node(3)
```

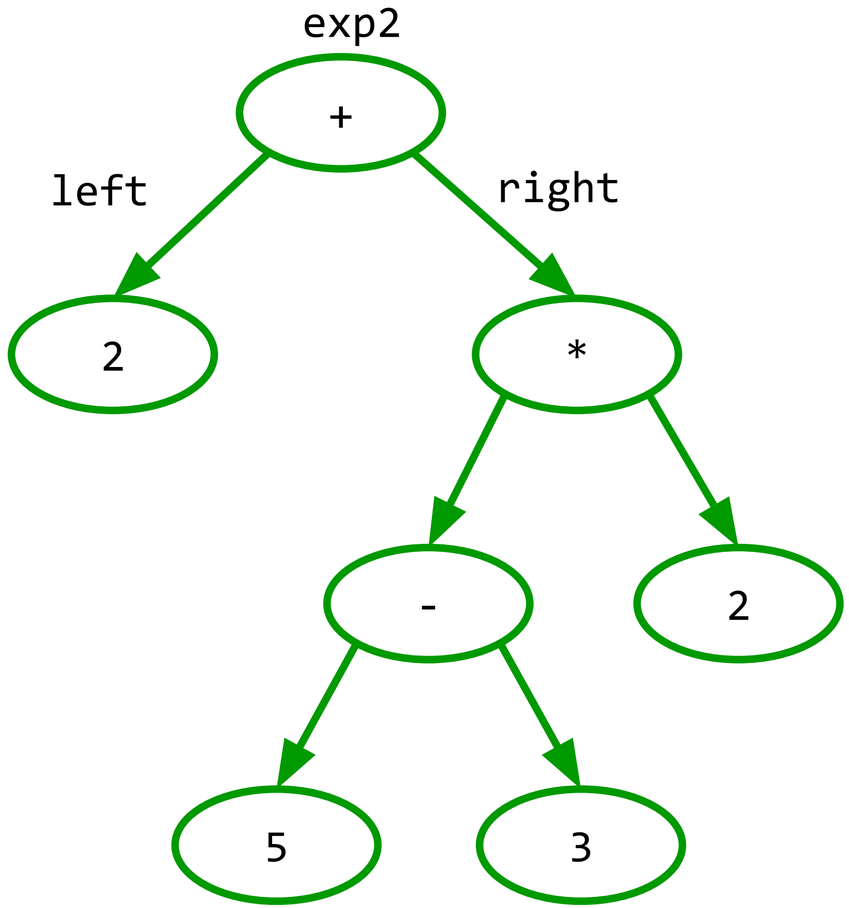
右のような木構造



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？

```
exp2 = Node('+')
exp2.left = Node(2)
exp2.right = Node('*')

exp2.right.left = Node('-')
exp2.right.right = Node(2)

exp2.right.left.left = Node(5)
exp2.right.left.right = Node(3)
```



数式を表示するには…

- 先ほど学習したように、数式には3種類の記法がありました

1. 中置記法
   - 演算子を真ん中に書く、一般的な数式の記法
   - 括弧が必要

➡ 通りがけ順

2. 前置記法
   - 演算子を最初に書く記法
   - 括弧は不要

➡ 行きがけ順

3. 後置記法
   - 演算子を後に書く記法
   - 括弧は不要

➡ 帰りがけ順

前置記法：Pythonで表現すると…

```
def print_prefix(node):
    print(node, end=' ')

    if node.left != None:
        print_prefix(node.left)

    if node.right != None:
        print_prefix(node.right)
```

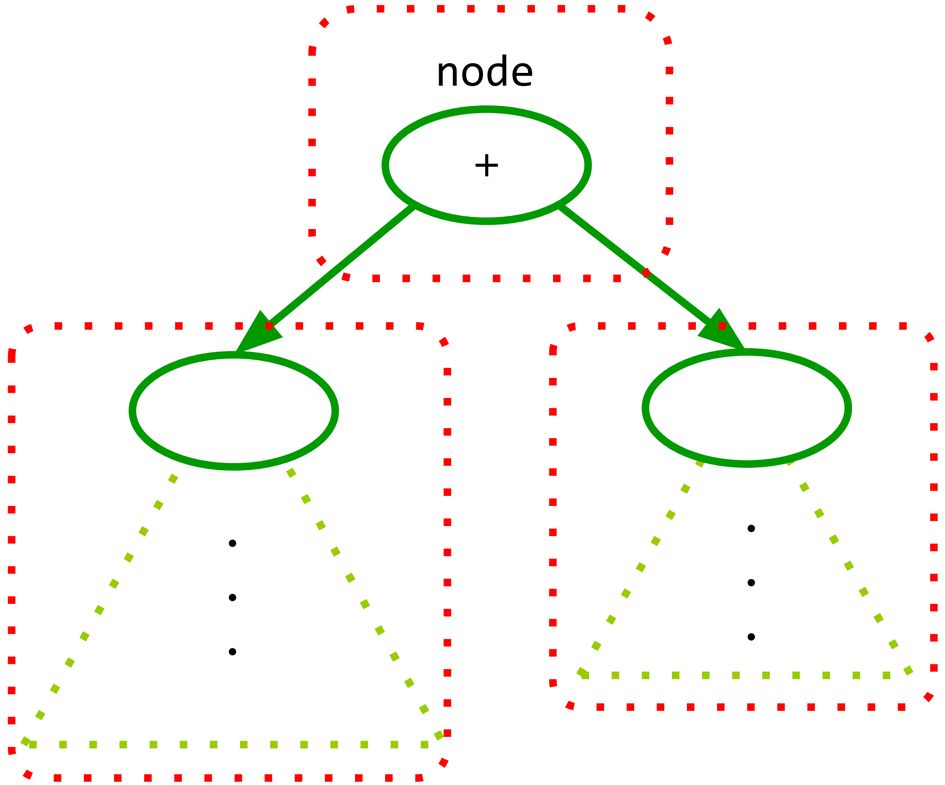
演算子を表示する

左の部分木を表示する

右の部分木を表示する

```
print_prefix(exp2)
```

```
+ 2 * - 5 3 2
```



空白を追加していることに注意！

後置記法：Pythonで表現すると…

```
def print_postfix(node):
    if node.left != None:
        print_postfix(node.left)

    if node.right != None:
        print_postfix(node.right)

    print(node, end=' ')
```

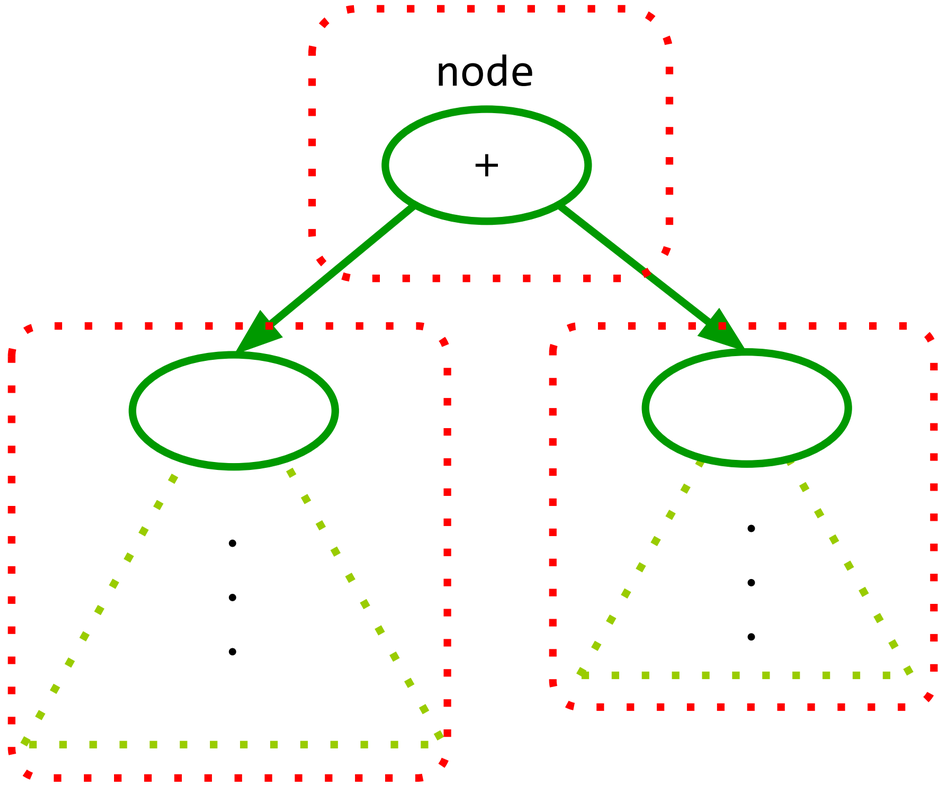
左の部分木を表示する

右の部分木を表示する

演算子を表示する

```
print_postfix(exp2)
```

```
2 5 3 - 2 * +
```



空白を追加していることに注意！

中置記法：Pythonで表現すると…

```
def print_infix(node):
    if node.left != None:
        print('(', end='')
        print_infix(node.left)

    print(node, end='')

    if node.right != None:
        print_infix(node.right)
        print(')', end='')
```

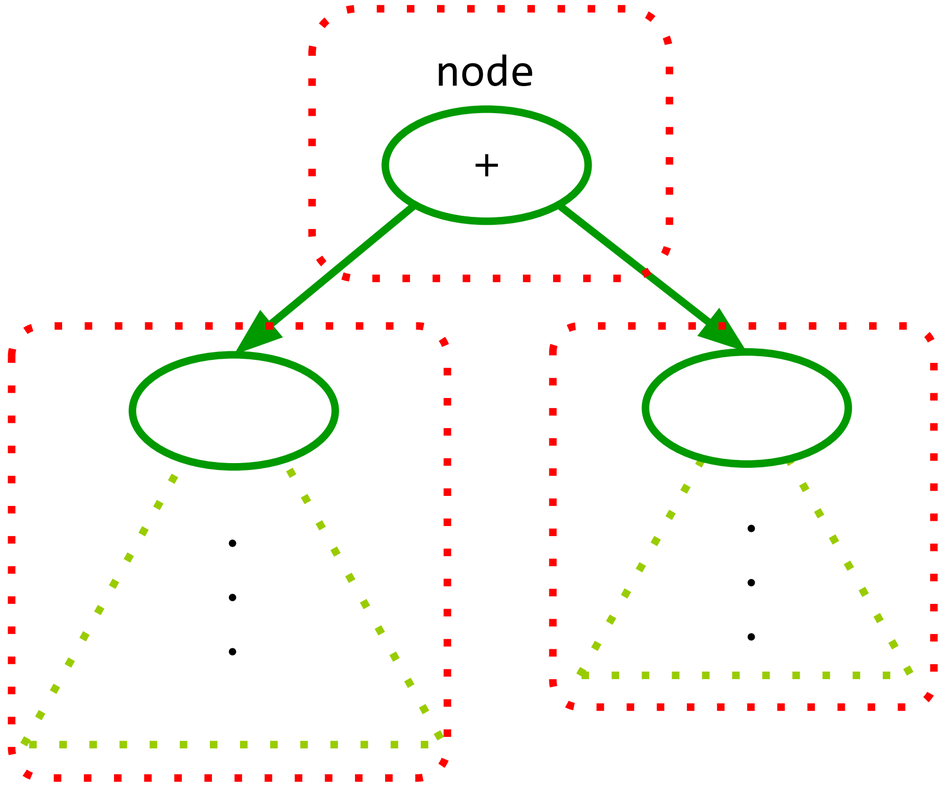
左の部分木を表示する

演算子を表示する

右の部分木を表示する

```
print_infix(exp2)
```

```
(2+((5-3)*2))
```



括弧を追加していることに注意！

数式を計算するには…

- 数式の木を作ってしまえば、再帰を使うことで簡単に計算を行うことができます
- 具体的には、以下のように考えます
  - このノードが演算子の場合は、左の部分木と右の部分木を計算し、その結果をもとに演算を行った結果を返します
  - このノードが数値の場合は、そのノードの値を返します

Pythonで表現すると…

演算子の場合は、再帰的に左の部分木と右の部分木を計算し、その結果をもとに演算をう

```
def eval_node(node):
    if node.value == '+':
        return eval_node(node.left) + eval_node(node.right)

    if node.value == '-':
        return eval_node(node.left) - eval_node(node.right)

    if node.value == '*':
        return eval_node(node.left) * eval_node(node.right)

    if node.value == '/':
        return eval_node(node.left) / eval_node(node.right)

    return node.value
```

数値の場合は、そのまま返す

```
eval_node(exp)
```

```
5
```