

# [Python] \_\_init\_\_(??? ???) & ????

17. 클래스를 만들 때, 클래스의 인스턴스를 생성할 때 자동으로 호출되는 메서드인 `__init__` 메서드를 정의할 수 있다.

```
class Node:
    def __init__(self, value):
        self.value = value
        self.children = []

def tree(li):
    nodes = [Node(i) for i in li]
    for i in range(1, len(li)):
        nodes[(i - 1) // 2].children.append(nodes[i])
    return nodes[0]

def calc(node, level=0):
    if node is None:
        return 0
    return (node.value if level % 2 == 1 else 0) + sum(calc(n, level + 1) for n in node.children)

li = [3, 5, 8, 12, 15, 18, 21]

root = tree(li)

print(calc(root))
```

- `__init__`: 클래스의 인스턴스를 생성할 때 자동으로 호출되는 메서드인 `__init__` 메서드를 정의할 수 있다.
- `children`: 클래스의 인스턴스의 자식 노드를 저장하는 리스트
- `tree()`: 리스트를 받아서 트리 구조로 변환하는 함수
- `calc()`: 트리를 순회하여 각 노드의 값을 계산하는 함수

이 i 1인 경우 ?

li = [3, 5, 8, 12, 15, 18, 21] 이고 i = 0일 때 nodes[(0 - 1) // 2] = nodes[-1]은 배열의 맨 끝을 가리키는 인덱스이다.

nodes[0]은 루트 (root) 노드, nodes[i]는 i번째 노드이다.

18, 21은 2번째 레벨, 이들을 처리할 때 2번째 레벨의 노드들을 어떻게 처리할지 ?

“이진 탐색을 위한 노드 인덱스 계산”

- 노드 인덱스 i를 받아서 노드 값을 반환하는 함수를 작성한다. "이진 탐색을 위한 노드 인덱스 계산"
- 이진 탐색을 위한 노드 인덱스 계산 함수를 작성한다. "이진 탐색을 위한 노드 인덱스 계산"

노드 인덱스	노드 레벨
nodes(index)	노드 인덱스 i를 받아서 노드 값을 반환하는 함수 (0부터 시작)
nodes(level)	노드 레벨을 받아서 노드 값을 반환하는 함수 (0부터 시작)

- index 3 (값 12)의 노드 인덱스는 1 → 6
- index 4 (값 15)의 노드 인덱스는 1 → 6
- index 5 (값 18)의 노드 인덱스는 2 → 8
- index 6 (값 21)의 노드 인덱스는 2 → 8

if node is None: 이 조건을 어떻게 처리할지 ?

- node가 None이면 "이진 탐색을 위한 노드 인덱스 계산" 함수를 호출하여 노드 값을 반환한다.
- if node is None: → "이진 탐색을 위한 노드 인덱스 계산" 함수를 호출하여 노드 값을 반환한다.
- 이진 탐색을 위한 노드 인덱스 계산 함수를 호출하여 노드 값을 반환한다. (=이진 탐색을 위한 노드 인덱스 계산)

이진 탐색을 위한 노드 인덱스 계산 함수를 호출하여 노드 값을 반환한다. 이진 탐색을 위한 노드 인덱스 계산 함수를 호출하여 노드 값을 반환한다.

```
3
 / \
5   8
```

```
/ \
12 15
```

如果 node 是 None，那么 node.children 就是 None，那么 node.value 就是 0，那么 node.value 就是 0。

```
if node is None:
    return 0 # 返回 0 表示没有子节点
```

## 1. ??? Node

```
class Node:
    def __init__(self, value):
        self.value = value # 存储节点的值
        self.children = [] # 存储节点的子节点 (列表)
```

- Node 是一个类，用于创建节点。
- value: 存储节点的值。
- children: 存储节点的子节点 (列表，最多 2 个)。

[ ]

```
n = Node(3)
print(n.value) # 3
print(n.children) # []
```

## 2. ?? ?? ?? tree()

```
def tree(li):
    nodes = [Node(i) for i in li]
    for i in range(1, len(li)):
        nodes[(i - 1) // 2].children.append(nodes[i])
    return nodes[0]
```

- li 是一个列表，包含要创建节点的数字。
- Node 是一个类，用于创建节点。
- [(i - 1) // 2] 是父节点的索引。

?: li = [3, 5, 8, 12, 15, 18, 21]

```

index : value
0      : 3    → root
1      : 5    → 3의 왼쪽 자식
2      : 8    → 3의 오른쪽 자식
3      : 12   → 5의 왼쪽 자식
4      : 15   → 5의 오른쪽 자식
5      : 18   → 8의 왼쪽 자식
6      : 21   → 8의 오른쪽 자식

```

### 3. ?? ?? calc()

```

def calc(node, level=0):
    if node is None:
        return 0
    return (node.value if level % 2 == 1 else 0) + sum(calc(n, level + 1) for n in
node.children)

```

- 이진 트리의 모든 노드를 방문하는 방법 .
- **level** (1, 3, ...)은 노드의 value를 더함 .
- 이진 트리의 모든 노드를 방문하는 방법 . level 1의 value를 더함 .  $5 + 8 = 13$

