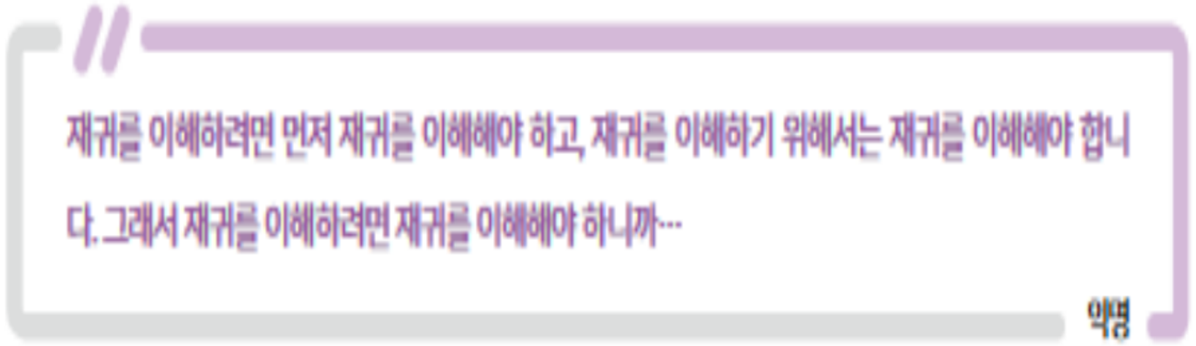
재귀



반복 알고리즘

- 동일한 루프로 단계를 반복해 문제를 해결하는 알고리즘
- 재귀(Recursion)는 문제를 더 작은 부분으로 나누고, 각 부분의 문제를 해결한 후 결과를 조합해 전체 문제의 답을 찾는 해결 방법!

재귀 알고리즘

- 재귀 알고리즘은 자기 자신을 호출하는 함수나 메서드 사용
- 재귀 알고리즘 함수는 입력을 변경하고 자신을 호출하며 결과를 전달하는 방식으로 작동 -> 무한 호출 되지 않도록 재귀 알고리즘을 빠져나가는 종료 조건(Base Case) 반드시 필요
- 자기 자신을 호출할 때 마다 종료 조건에 가까워 져야 한다.

Python 복사 캡션 ···

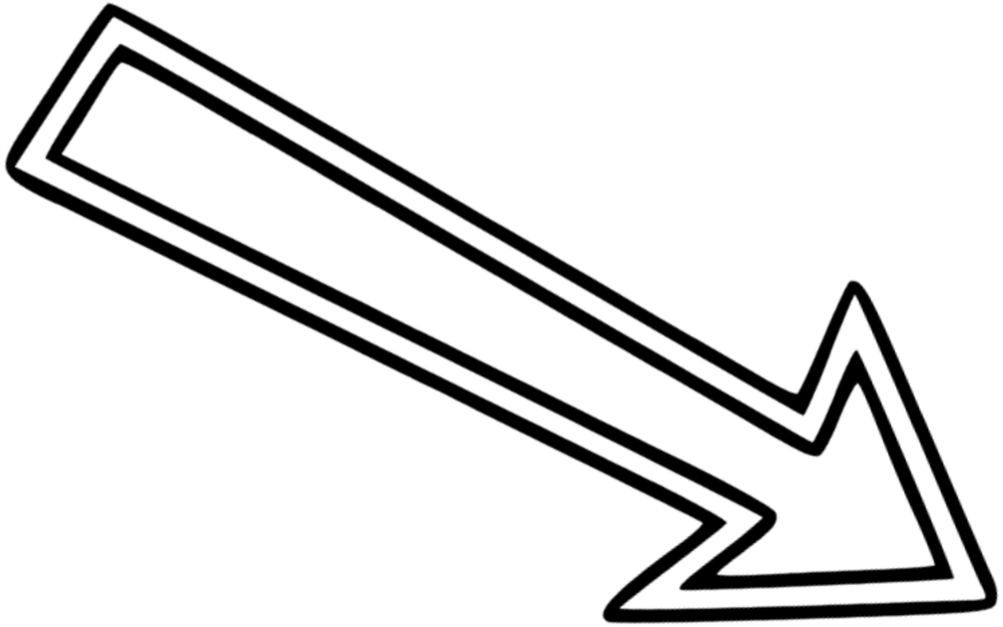
```python
def factiorial(n):
  the_product = 1
  while n > 0:
    the_product *= n
    n = n -1
  return the_product
```

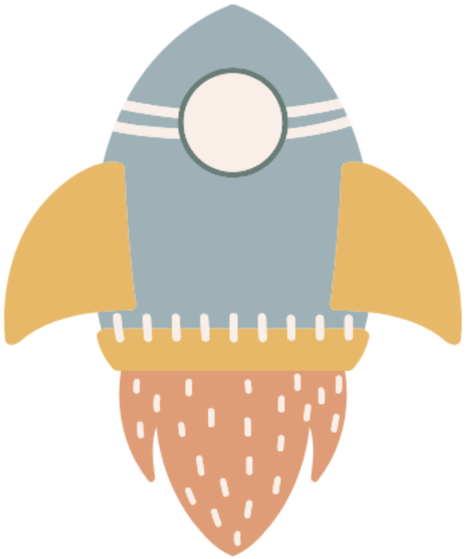
⇒ 이것을 재귀 함수로 바꿔 준다면

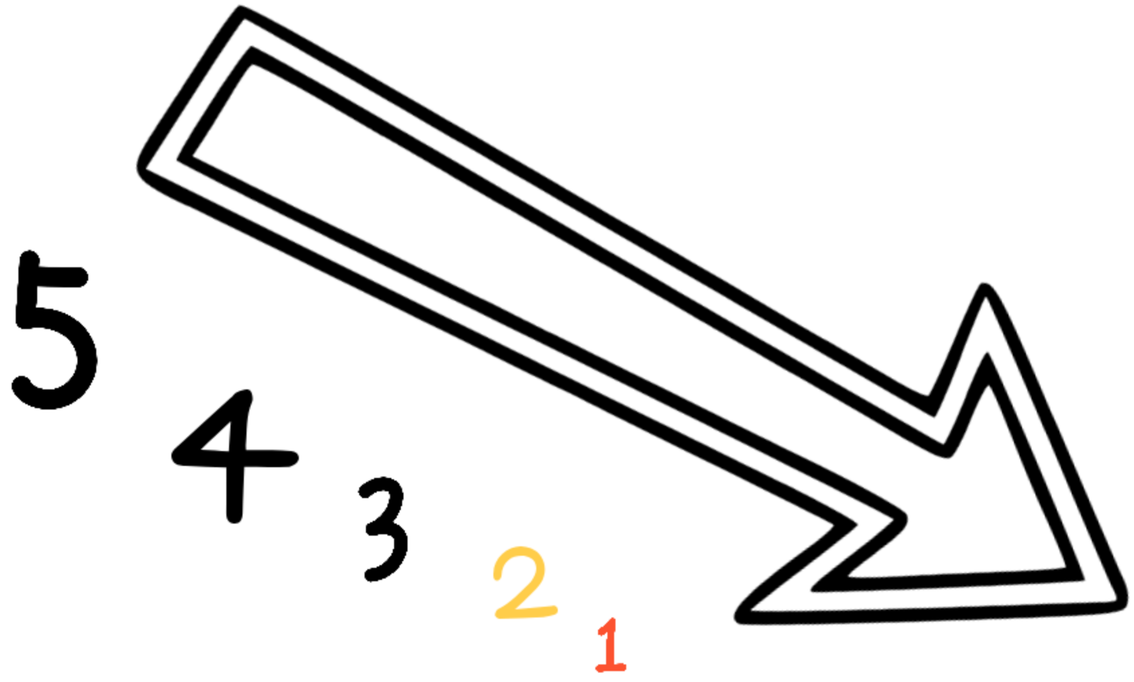
```python
def factioial(n):
  if n == 0:
    return 1
  return n * factorial(n - 1)
```

재귀란 무엇인가?

재귀란 무엇인가?






5
4
3
2
1

재귀의 세 가지 법칙

- 반드시 종료 조건이 있어야 한다.
- 반드시 자기 자신의 상태를 변경하면서 종료 조건에 가까워져야 한다.
- 반드시 자기 자신을 재귀적으로 (다시 돌아와) 호출해야 한다.

재귀를 사용할 때?

사실 재귀로 해결할 수 있는 문제는 반복문(for...)으로 해결 가능

- but, 재귀의 장점은 간결함
- 하지만, 파이썬 내부 스택에 데이터를 저장하므로 메모리를 더 소비하고,

반복 알고리즘에 비해 디버깅(오류 수정)이 어려움

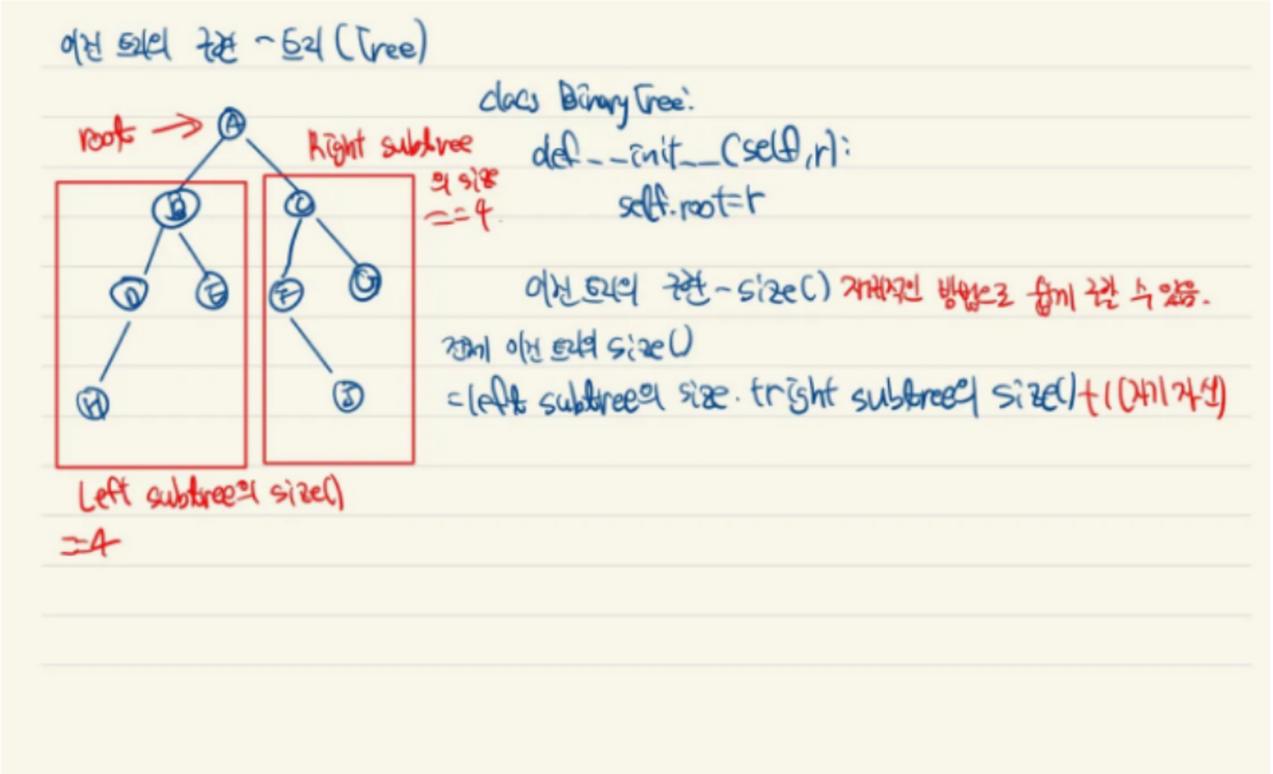
Day5 > 08-data-structure > test.py > ...

```python
def recursion(n):
    if n ==1:
        print(1)
        return  # 재귀 종료
    recursion(n-1)
    print(n)
def iteration(n):
    for i in range(1, n+1):
        print(i)

recursion(10)
iteration(10)
```

재귀를 사용할 때?

이진 트리에서 활용 가능! (알아만 두자!)



- 이진 트리의 구현 - size() 재귀적인 방법으로 쉽게 구할 수 있음
- 전체 이진 트리의 size() = left subtree의 size + right subtree 의 size +1 (자기 자신)

이진 트리의 구현 - size()

```
class Node:
  def size(self):
    l = self.left.size() if self.left else 0
    r = self.right.size() if self.right else 0
    return l + r + 1

class BinaryTree:
  def size(self):
    if self.root:
      return self.root.size()
    else:
      return 0
```

고생하셨습니다!