🔷

재귀함수

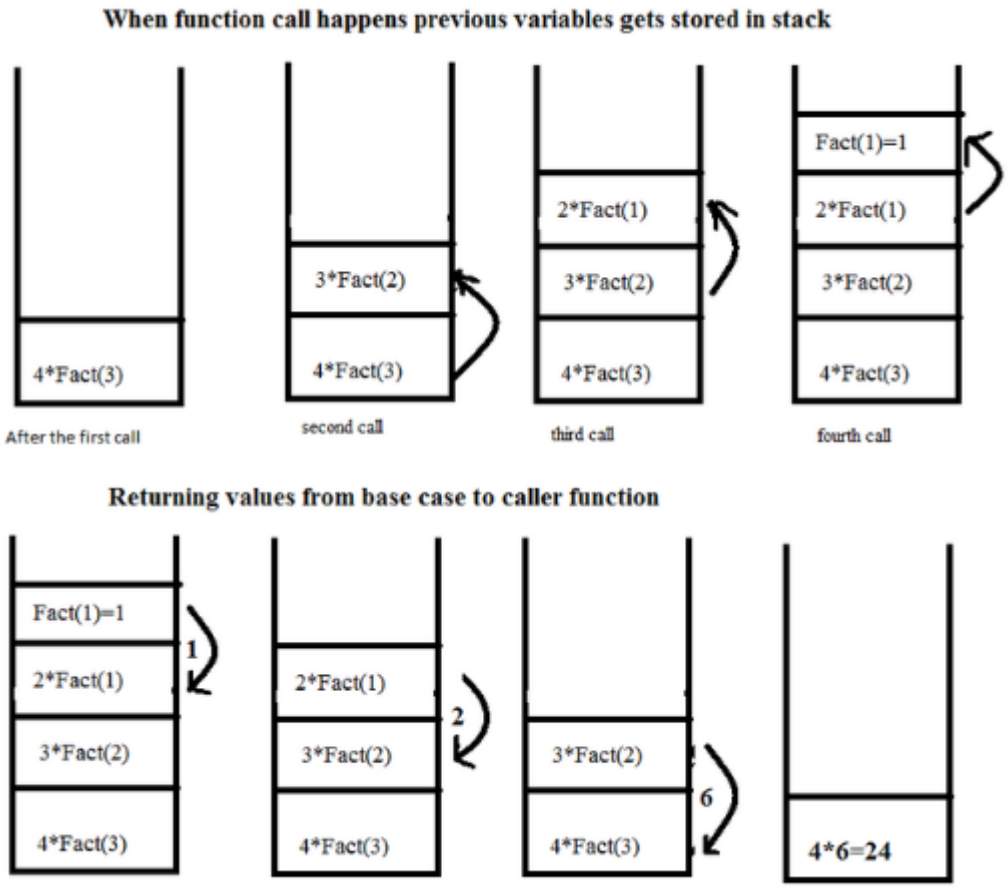
재귀함수

재귀(recursion) : 자기 자신을 호출하는 것

⚠️ 반드시 재귀 함수의 종료 조건을 명시해야함

재귀함수는 호출 될 때마다 call stack에 쌓이게 되며 한계치 이상으로 호출이 되면 StackOverflowError가 발생, 프로그램이 중단됨

Python에서는 이를 방지하기 위해 call stack을 1000개까지만 허용.



🌟 재귀 함수는 너무 많은 호출이 필요하지 않는 경우면서, 반복문으로 구현하기에 지저분하지만, 재귀 함수로 구현하면 깔끔한 경우에 사용!

<재귀 함수를 사용해 팩토리얼 구하기>

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

print("5!:factorial(5))
```

재귀함수를 사용할 때 같은 값을 구하는 연산을 반복하면 실행 시간이 오래 걸림.

⇒ 메모(memo) 사용하면 해결 됨!

메모화

: 딕셔너리를 사용해서 한 번 계산한 값을 저장함. 딕셔너리에 값이 메모되어 있으면 처리를 수행하지 않고 곧바로 메모된 값을 돌려줌.

```
# 메모 변수 생성

dictionary = {
    1 : 1,
    2 : 2,

}

def fibonacci(n):
    if n in dictionary:
        #메모가 되어 있으면 메모된 값을 리턴
        return dictionary[n]
    else:
        #메모가 되어 있지 않으면 값을 구함
        output = fibonacci(n - 1) + fibonacci(n - 2)
```

```
        dictionary[n] = output
        return output
```

조기리턴

: 흐름 중간에 return 키워드를 사용해 코드를 줄일 수 있음.

탈출 조건

함수 내에 return 을 사용해서 탈출 조건을 꼭 명시할 것.

예시.

BOJ. 11729 하노이 탑 이동 순서 (실버1)

```
def hanoi_tower(n, start, end) :
    if n == 1 :
        print(start, end)
        return

    hanoi_tower(n-1, start, 6-start-end)
    print(start, end)
    hanoi_tower(n-1, 6-start-end, end)

n = int(input())
print(2**n-1)
hanoi_tower(n, 1, 3)
```

예제문제

~~BOJ. 11729 하노이 탑 이동 순서 (실버1)~~

BOJ. 2447 별찍기 (골드5)

BOJ 17478 재귀함수가 뭔가요? (실버5)

BOJ. 2630 색종이 만들기 (실버2)

BOJ. 6603 로또 (실버2)

BOJ. 5094 Moo게임 (실버1)

BOJ. 4779 칸토어집합 (실버3)

BOJ. 23304 아카라카 (실버2)