Day - 1

Recursion

it is a process in which a function calls itself directly or indirectly.

Syntax:-

```
def func():
        |
        | (recursive call)
        |
    fun(c) - - -
```

Example :- (factorial Example)

```
def fact(n):
    #base case
    if n==0:
        return 1
    else:
        return n * fact(n-1)
```

Note :-

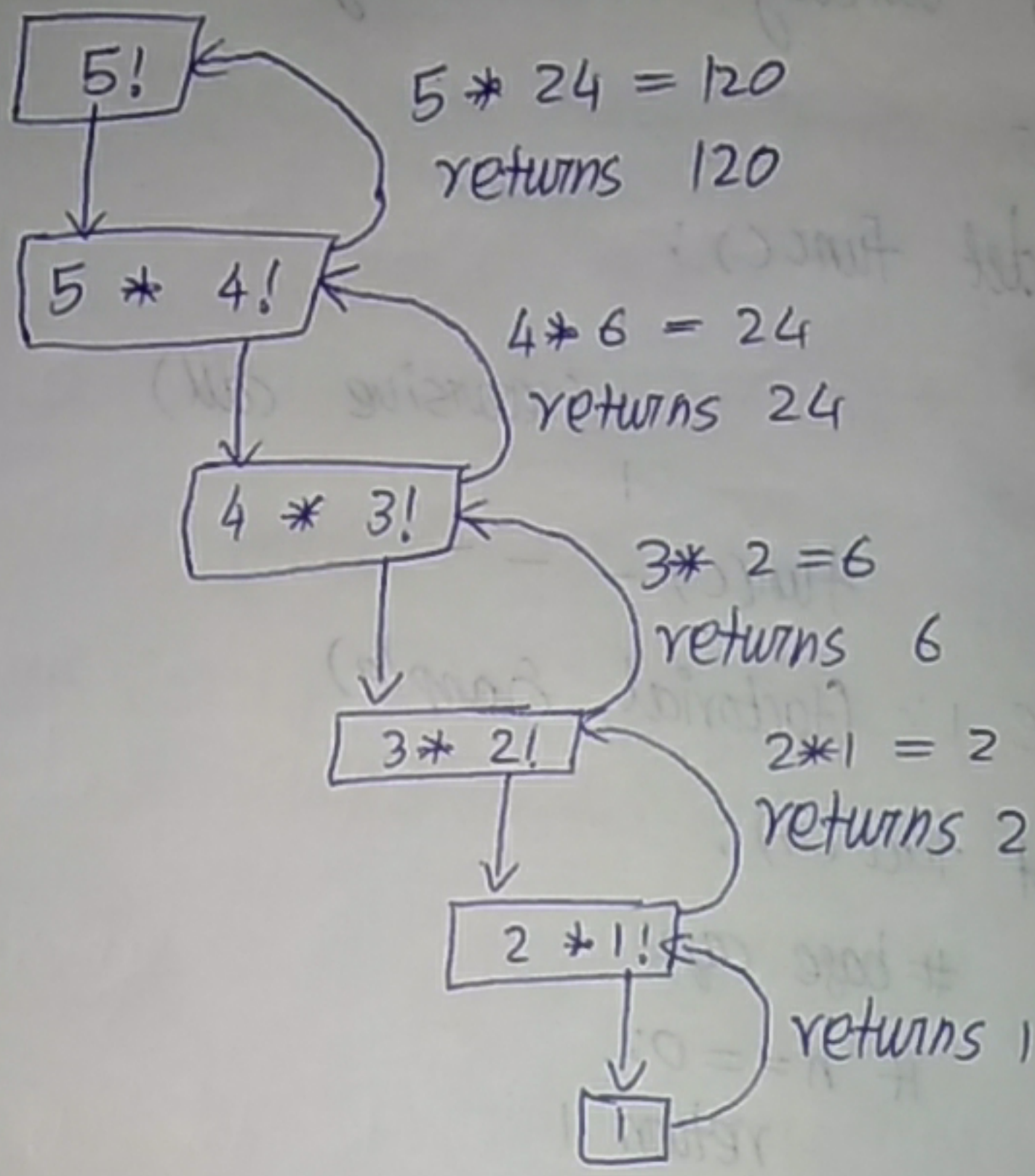
Whenever ~~you are~~ trying to develop a recursive solution it is very important to think about the base case, as your solution will need to return the base case once all the recursive cases have been worked through.

Test case 1 :-

n=5

factorial of n = 120

Explanation :- Final value =120



Types of recursion :-

* Direct recursion → it calls the same fucto again
* in-direct recursion
* Tail recursion
* Non-tail recursion.

Indirect function:-

If it calls another function (let say fun2) and then fun2 calls fun1 directly or indirectly.

```
def fun1():                    def fun2():
    // some code                   // some code
    fun2()                         fun1()
    // some code                   // some code
```

Tail recursion:-

A function is tail recursive when recursive call is last thing executed by the function.

factorial function as tail recursive

```
def fact(n, a=1):
    if (n==0):
        return a
    return fact(n-1, n*a)
```

non-tail reawsive function:-

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

Advandage of wjing recursion:-

* a complicated function can be split down into smaller sub-problems utilliziny recursion

* look simple and effective.

disadvandage of wjing recursion:-

* a lot of time and memory is taken through.

* challenging to debug