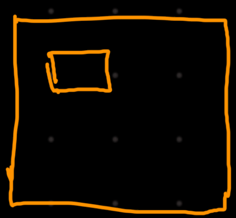
Recursion

Concepts of functions → we can ofcourse call different f^n from inside a function

Can a function call itself as well? Yes



stack

⇒ stack overflow

With this exercise 2 things are clear:

1. Functions wait in the memory till they are resolved

2. When a f^n finishes execution, then only it comes of program and gets deleted from our stack.

Recursion is a function calling itself.

printnum() → printnumfour()

printnum() → printnum()

'Recursion is when solution of a problem depends on some smaller problem'

$5 \to 4 \to 3 \to 2 \to 1$

Factorial of a number:-

$$\to fact(n) = n! = n * \underbrace{(n-1) * (n-2) \cdots 1}$$

$$\boxed{n! = n * (n-1)!}$$

$$fact(n) = n * fact(n-1)$$

$$fact(n-1) = (n-1) * (n-2) \cdots 1$$

$$= (n-1) * (n-2)!$$

recursion was giving us maximum depth exceeded.

we have to make sure that recursion stops somewhere and don't keep on going till ∞

we add a base case

Now, we will not be making thes f^n calls and everything and even think of recursion this deep.

PMI

PMI (Principle of mathematical Induction)

Initial few problem we will do can be done iteratively but we have to use recursion to get mastery. and as it is way to easy like this.

Recursion principle based on PMI, so understanding this will further help us to master recursion.

PMI

It is used to prove some fact.

eg.

$$f(n) = \text{true} \quad \forall n$$

using PMI, we can break this problem in 3 steps.

1. Prove $F(0)$ or $F(1)$ is true.
2. Assume $F(K)$ is true → Induction hypothesis
3. Using step 2, i.e. $F(K)$ is true*

   prove that $F(K+1)$ is true.

$$sum(n) = \Sigma n = \frac{n(n+1)}{2}$$

let us prove using PMI :

① Base case $F(0)$ or $F(1)$

$L.H.S. = \Sigma 0 = 0$

$R.H.S. = \frac{0(0+1)}{2} = 0$

$L.H.S = R.H.S.$

$L.H.S. = \Sigma 1 = 1$

$R.H.S = \frac{1(1+1)}{2} = 1$

$L.H.S = R.H.S.$

② Assume (don't Question) that $F(k)$ is true

$$\Rightarrow \Sigma k = sum(k) = \frac{k(k+1)}{2} \Rightarrow \text{true}$$

③ Now, we need to prove that $F(k+1)$ is true.

$$\Rightarrow \Sigma k+1 = sum(k+1) = \frac{(k+1)(k+2)}{2}$$

$L.H.S. = \Sigma k+1 = k+1 + \Sigma k$ from(2)

$$= k+1 + \frac{k(k+1)}{2}$$

$$= \frac{(k+1)*2}{2} + \frac{k(k+1)}{2}$$

$$= \frac{(k+1)(k+2)}{2} = R.H.S.$$

Hence, by using PMI, we have proven that

$$\Sigma n = \frac{(n)(n+1)}{2}$$

Now, let us go a little deep & understand how exactly it works?

So basically we know the base and how to take a step forward i.e. we can keep on taking our steps.

$f(4)$

$f(3)$

$k+1$

$f(2) = \text{true}$

$k \to 1 = \text{true} \quad f(1)$

$f(1) = \text{true}$

So in every recursion question, we just have to apply P.M.I.,
3 steps

1) Base Case

2) Assume for $(n-1)$ is true

small problem

3] using assumption is step 2 and base case, we can solve for bigger problem.

we don't need to draw these f^n calls over and over again.

Q:- Finding 2^n for a given n.
using recursion.

$$2^n = 2 * 2^{n-1}$$

$$\boxed{pow(2,n) = 2 * pow(2, n-1)}$$

recursion
fn / relation