Introducción a la programación

Clase 5 - La recursión hace la magia

Bernardo Subercaseaux , CC1002 - 6 , 15 de septiembre 2015

Desafío de Motivación

Si una persona es capaz de subir 1 o 2 peldaños en cada paso, ¿De cuántas formas puede subir una escalera de 7 peldaños ?

Solución

- Formas(n) = Formas(n-1) + Formas(n-2)
- Formas(1) = 1, Formas(2) = 2
- Formas(7) = Formas(6) + Formas(5) = Formas(5) + Formas(4) + Formas(4) + Formas(3) = ... (....)....(....).....

EN PYTHON...

```
## Función que retorna las formas de subir una escalera
## int --> int
## Ejemplo: Formas(2) = 2
def formas(n):
    if n == 1:
        return 1
    if n == 2:
        return 2
    return formas(n-1) + formas(n-2)
```

```
>>> formas(7)
21
```

Fundamentos de La recursión

- Diremos que una función es recursiva si se llama a sí misma.

- Toda función recursiva debe tener un caso base, esto es, un caso en que responde directamente, sin llamarse a sí misma.

- Cuando una función recursiva se llama a sí misma, debe hacerlo con parámetros que representan un problema menos complejo, y que finalmente alcanzan el caso base.

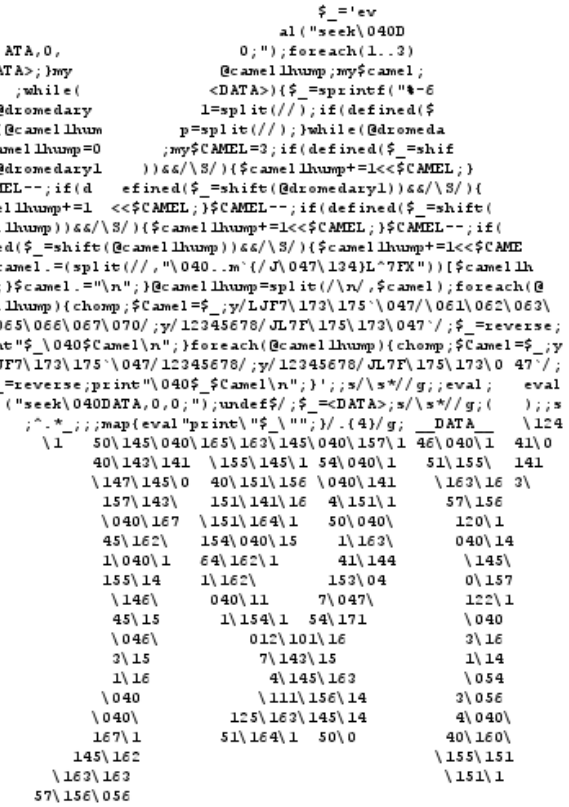
¿Cómo se ve en código?

##Descripción

##Dominio -> Recorrido

##Ejemplo

```
def funcion(n):
    if f(n):
        return blah
    return funcion(g(n))
```



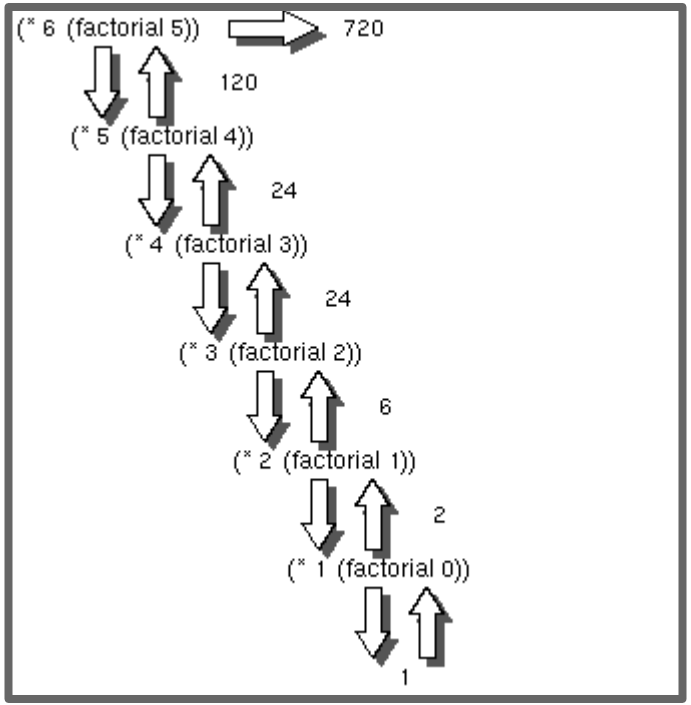
Factorial de Un Número

7! = 7*6*5*4*3*2*1 = 7 * 6!

n! = n * (n-1)!

1! = 1

¡Prográmenla!

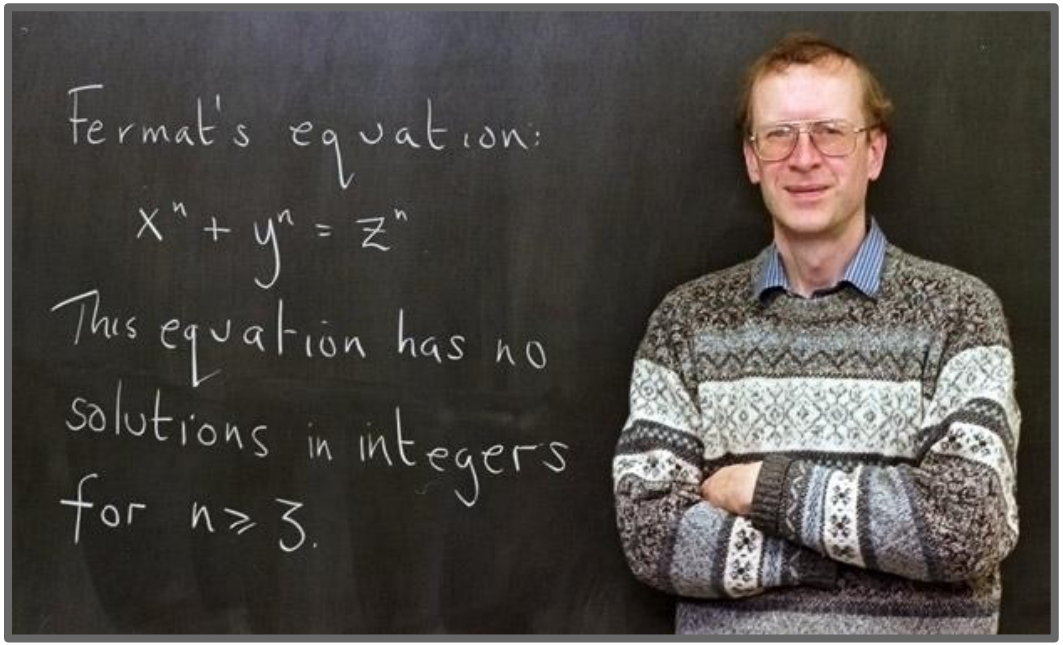


Exponenciación

x^n = x * x^(n-1)

x^0 = 1

¡Prográmenla!



¿Se puede hacer más rápido?

Exponenciación Rápida

$x^n = x^{(n/2)} * x^{(n/2)}$

$x^0 = 1$

$O(\log(n)) << O(n)$

¿Aplicaciones?

¿Que le falta a ese código?

```
## Función que realiza exponenciación rápida
## num, num --> num
## Ejemplo: fastExp(2,3) --> 8
##          fastExp(3.14,2) --> 9.8596
def fastExp(base,exp):
    if exp == 0:
        return 1
    if exp == 1:
        return base
    var = fastExp(base,exp/2)
    return var * var * fastExp(base,exp%2)
```

Números de Fibonacci

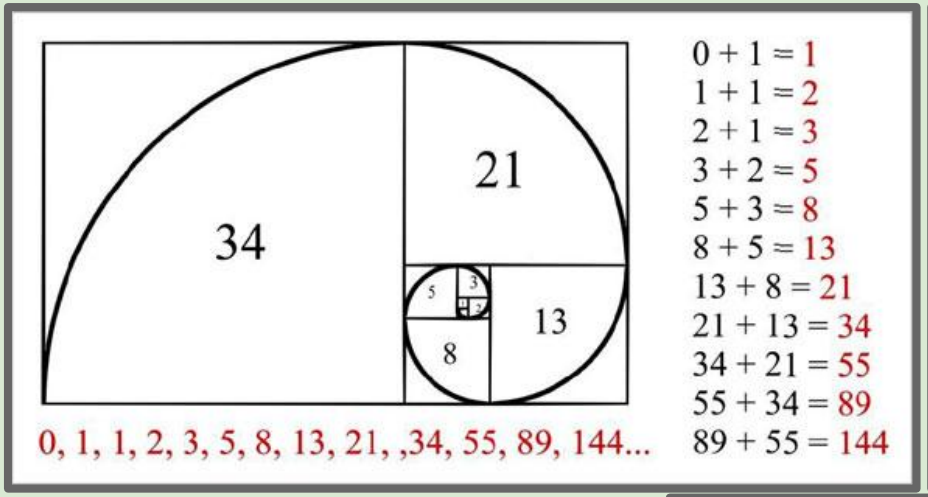
F(0) = 1

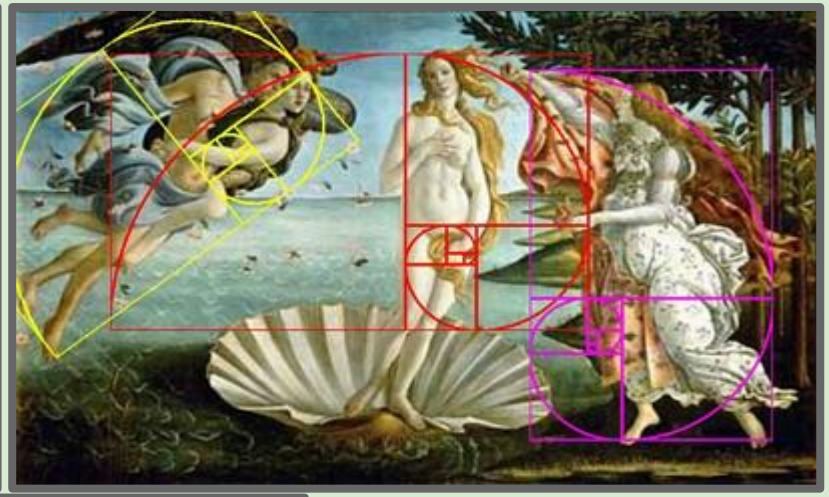
F(1) = 1

F(n) = F(n-1) + F(n-2) ¿Les suena de alguna parte?

```
## Función que calcula números de Fibonacci
## int -> int
## Ejemplo: fib(8) = 21
def fib(n):
    if n <= 1:
        return n
    return fib(n-1) + fib(n-2)
assert fib(8) == 21
```

¿ Y PARA QUÉ?


34
21
13
8
5
3
2
1
0, 1, 1, 2, 3, 5, 8, 13, 21, ,34, 55, 89, 144...
0 + 1 = 1
1 + 1 = 2
2 + 1 = 3
3 + 2 = 5
5 + 3 = 8
8 + 5 = 13
13 + 8 = 21
21 + 13 = 34
34 + 21 = 55
55 + 34 = 89
89 + 55 = 144



Those who cannot remember the past are condemned to repeat it.

-Dynamic Programming

```
## Función que calcula números de Fibonacci
## int -> int
## Ejemplo: fib(8) = 21
global fibo
fibo = [0 for i in range(10000)]
def fib(n):
    if n <= 1:
        return n
    if fibo[n] != 0:
        return fibo[n]
    fibo[n] = fib(n-1) + fib(n-2)
    return fibo[n]
assert fib(8) == 21
```

¿Qué problemas se Resuelven con Recursión?

- Si todo lo que tienes es un martillo, todos los problemas te parecerán un clavo. -Maslow

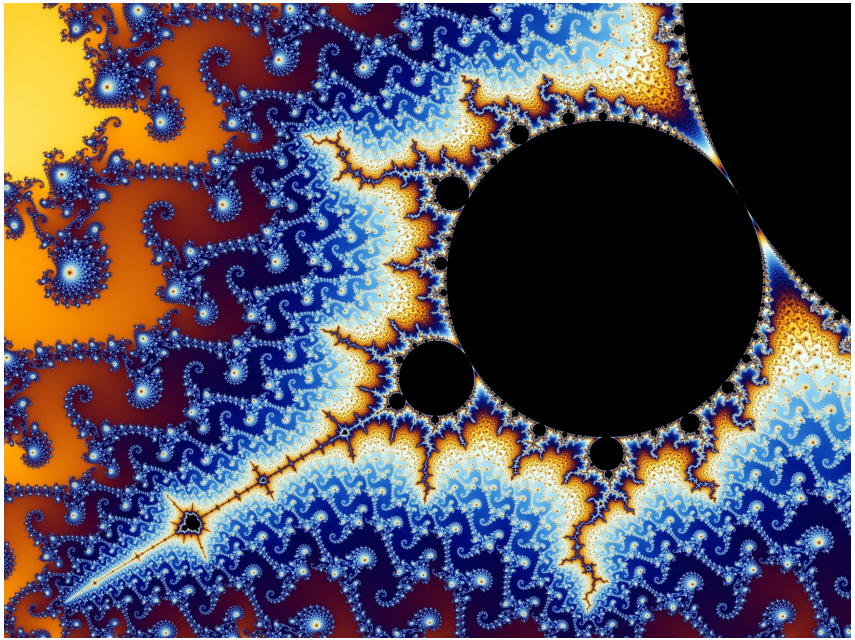
- Divide et Impera (Divide y vencerás) -Julio César,Napoleón

Ejemplos de Recursión

- Determinar si una palabra es palíndrome.
- Invertir un número
- Convertir de una base a otra
- Salir de un laberinto
- Buscar cosas
- Resolver problemas de conteo
- Calcular sumatorias
- Resolver problemas de maximización/minimización
- Teoría de Números (Máximo Común Divisor, Función Phi, etc)
- Torre de Hanoi
- Fractales

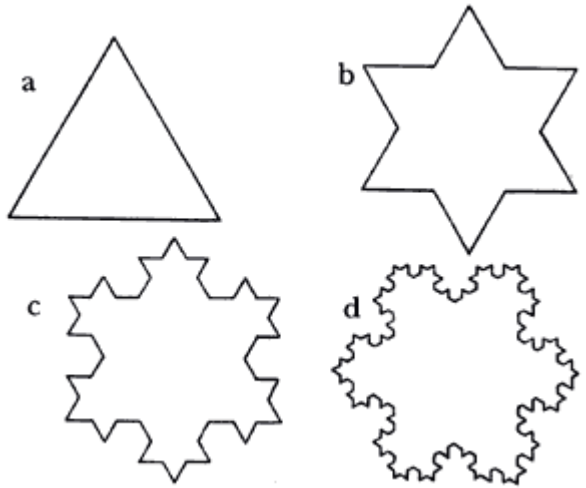
¿ FRACTALES ?

Estructura geométrica que se repite a diversas escalas.



Copo de Nieve de Koch

- Módulo Turtle
- turtle.forward()
- turtle.right()
- turtle.left()
- turtle.speed()
- turtle.done()
- Divide And Conquer x 2



Final: Dudas, Resolución de Problemas, Comentarios

correo: bernardosubercaseaux@gmail.com