

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

- $FORMAS(N) = FORMAS(N-1) + FORMAS(N-2)$
- $FORMAS(1) = 1, FORMAS(2) = 2$
- $FORMAS(7) = FORMAS(6) + FORMAS(5) = FORMAS(5) + FORMAS(4) + FORMAS(4) +$
 $FORMAS(3) = \dots\dots (\dots) \dots (\dots) \dots\dots$

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))

```

    $_='eval
    al("seek\040D
    0;");foreach(1..3)
    {<DATA>;my
    my$Camel;while(
    9s",$_);my@dromedary
    =<DATA>){@camel_lhump
    ryl){my$camel_lhump=0
    t(@dromedary1
    $CAMEL--;if(d
    $camel_lhump+=1
    @camel_lhump){&&/S/){$camel_lhump+=1<<$CAMEL;
    defined($_)=shift(@dromedary1)&&/S/){
    $camel_lhump+=1
    defined($_)=shift(@camel_lhump)&&/S/){$camel_lhump+=1<<$CAME
    L;};$camel=(split//,"040..n"/J\047\134)L^7FX")){$camel_lh
    ump];}$camel.="n";}@camel_lhump=split(/n/,$camel);foreach(@
    camel_lhump){chomp;$Camel=$_/LJF7\173\175\047\061\062\063\
    064\065\066\067\070//y/12345678/JL7F\175\173\047//;$_reverse;
    print"$\040$Camel\n";}foreach(@camel_lhump){chomp;$Camel=$_/y
    /LJF7\173\175\047\12345678//y/12345678/JL7F\175\173\047//;
    $_reverse;print"\040$_$Camel\n";};s/\s*/g;eval;eval
    ("seek\040DATA,0,0;");undef$;$_=<DATA>;s/\s*/g;(
    );s
    ;^.*;;map(eval"print\"$_\"";)/.(4)/g;DATA
    \1
    50\145\040\165\163\145\040\157\146\040\141\0
    40\143\141
    \155\145\154\040\151\155\141
    \147\145\040\151\156\040\141
    \163\163\
    157\143\151\141\164\151\157\156
    \040\167\151\164\150\040\120\1
    45\162\154\040\151\163\040\14
    1\040\164\162\141\144\145\
    155\141\162\153\040\157
    \146\040\117\047\122\1
    45\151\154\154\171\040
    \046\012\101\163\16
    3\157\143\15
    1\164\145\163\054
    \040\111\156\143\056
    \040\125\163\145\144\040\
    167\151\154\150\040\160\
    145\162\155\151
    \163\163\151\1
    57\156\056
```

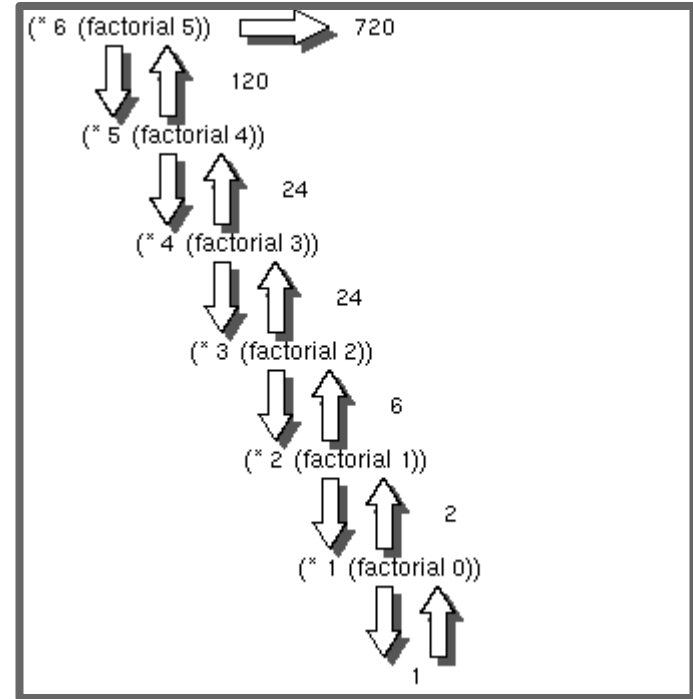
FACTORIAL DE UN NÚMERO

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

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

$$1! = 1$$

¡Prográmenla!

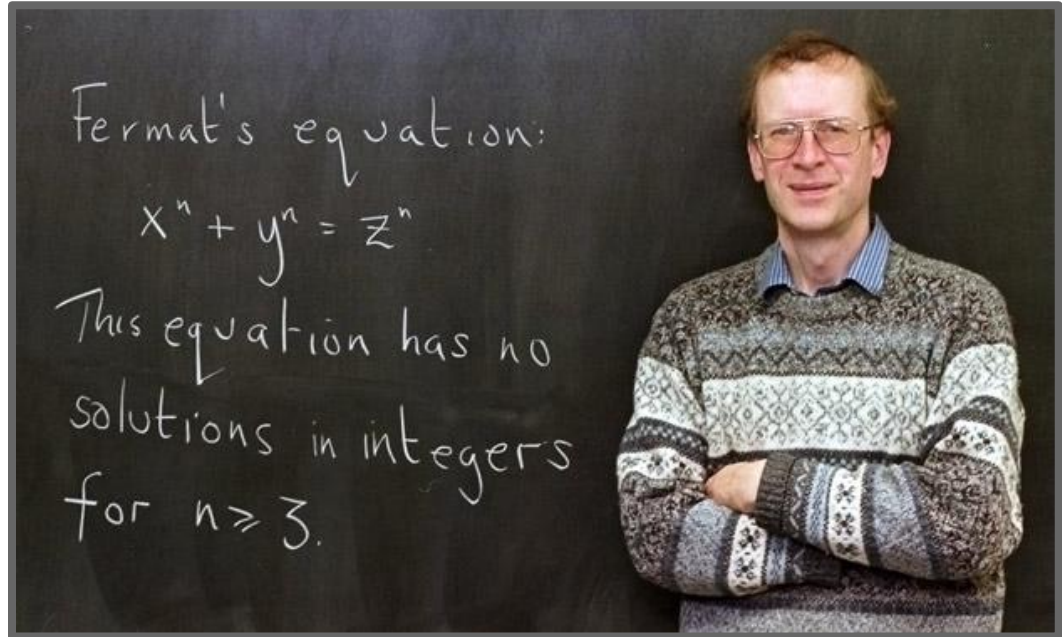


EXPONENCIACIÓN

$$x^n = x * x^{(n-1)}$$

$$x^0 = 1$$

¡Prográmenla!



¿Se puede hacer más rápido?

EXPONENCIACIÓN RÁPIDA

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

$$x^0 = 1$$

$$O(\log(n)) \ll O(n)$$

```
## 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)
```

¿Aplicaciones?

¿Que le falta a ese código?



NÚMEROS DE FIBONACCI

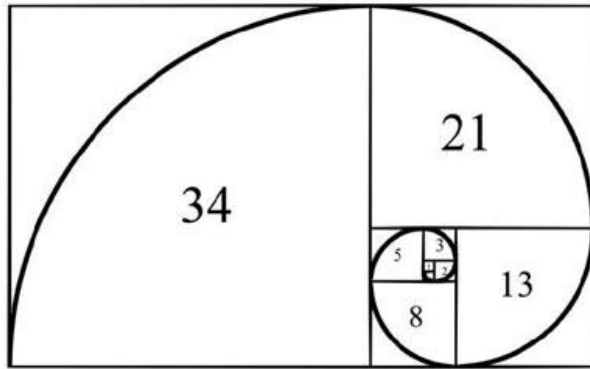
$$F(0) = 1$$

$$F(1) = 1$$

$$F(n) = F(n-1) + F(n-2) \quad \text{¿Les suena de alguna parte?}$$

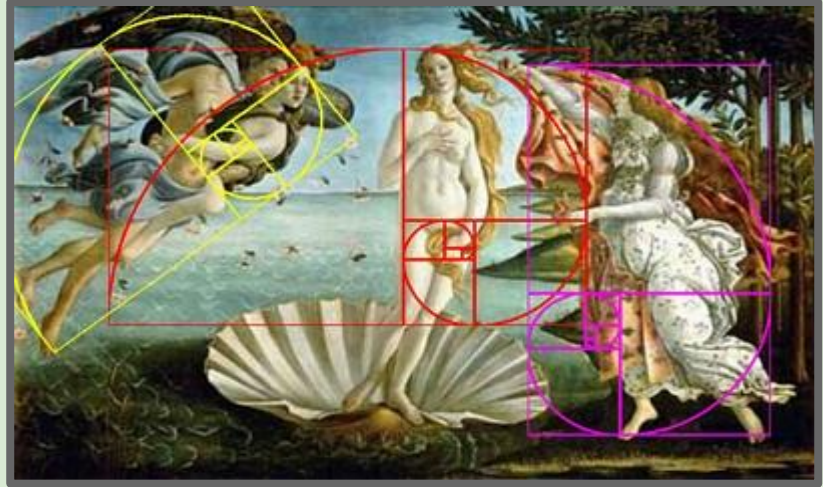
```
## Función que calcula números de Fibonacci
## int -> int
## Ejemplo: fib(8) = 21
def fib(n):
    if n <= 1:
        return n
    return fib(n-1) + fib(n-2)
assert fib(8) == 21
```

¿Y PARA QUÉ?



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

$$\begin{aligned}0 + 1 &= 1 \\1 + 1 &= 2 \\2 + 1 &= 3 \\3 + 2 &= 5 \\5 + 3 &= 8 \\8 + 5 &= 13 \\13 + 8 &= 21 \\21 + 13 &= 34 \\34 + 21 &= 55 \\55 + 34 &= 89 \\89 + 55 &= 144\end{aligned}$$



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?

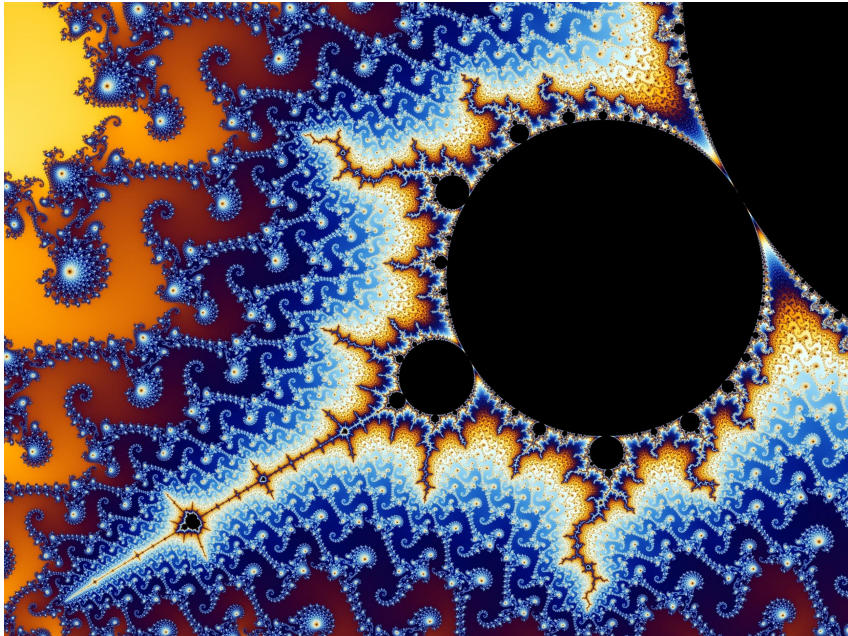
- SI TODO LO QUE TIENES ES UN MARTILLO, TODOS LOS PROBLEMAS TE PARECERÁN UN CLAVO. -MASLOW
- DIVIDE ET IMPERA (DIVIDE Y VENCERÁS) -JULIO CÉSAR, NAPOLEÓN

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

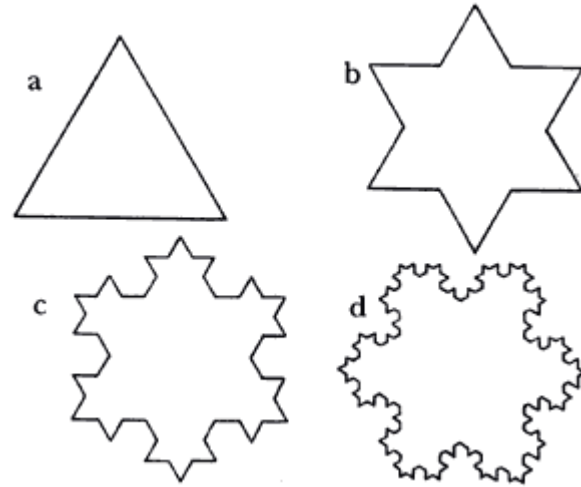
¿ FRACTALES ?

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