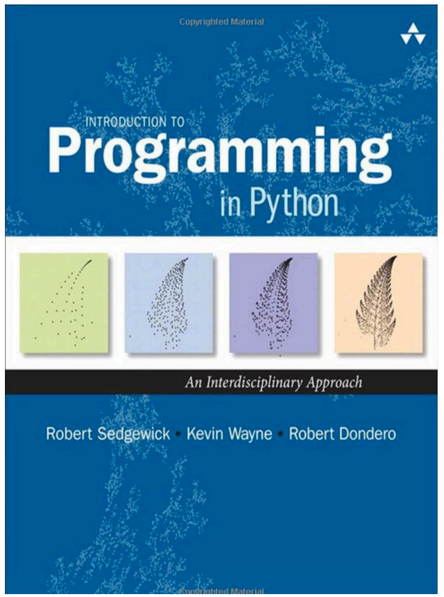
Tecnologías de la Información II

Clase 10: Funciones Recursivas

Daniela Opitz
dopitz@udd.cl

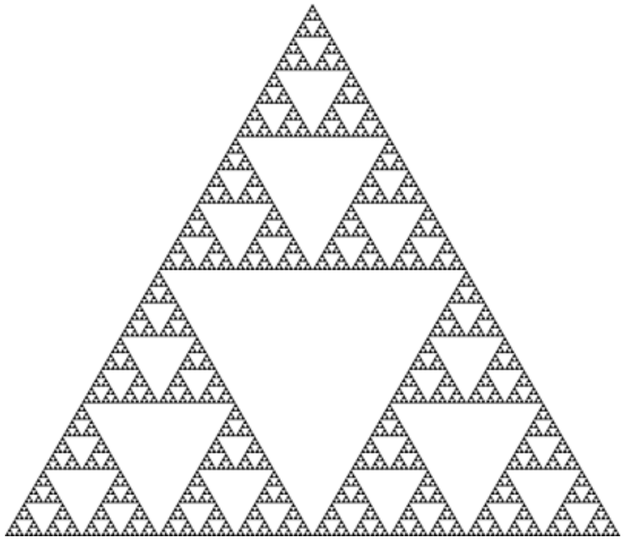


Basada en presentaciones oficiales de libro Introduction to Programming in Python (Sedgewick, Wayne, Dondero).

Disponible en https://introcs.cs.princeton.edu/python

Recursión o Recursividad

- En ciencias de la computación la recursion o recursividad es forma de resolver problemas tal qué la solución de este depende de las soluciones de pequeñas instancias del mismo problema.[2]
- Cualquier loop (ciclo) puede ser reemplazado por una función recursiva.
- La solución de algunos problemas usando recursión puede requerir una excesiva memoria.



Recursion. Algo de humor!

- Recursividad, véase *Recursividad*.
- Lo primero para entender la recursividad, es entender la recursividad».
- Buscando en Google recursión o recursividad

Ejemplo 1. Factorial

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

¿Qué hace la función?

$$(n\text{-}1)! = (n\text{-}1) \times (n\text{-}2) \times \ldots \times 2 \times 1$$

$$n! = n \times (n\text{-}1)! = n \times (n\text{-}1) \times (n\text{-}2) \times \ldots \times 2 \times 1$$

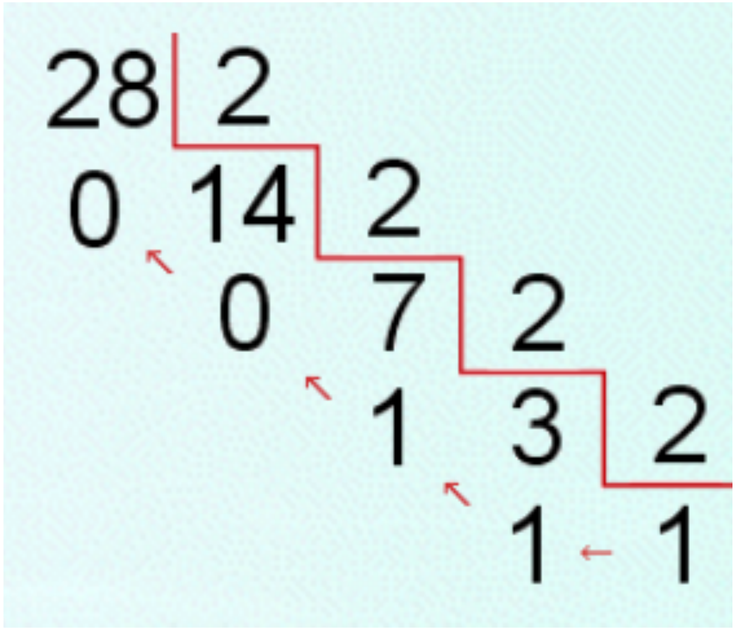
```
factorial(5)
    factorial(4)
        factorial(3)
            factorial(2)
                factorial(1)
                    return 1
                return 2*1 = 2
            return 3*2 = 6
        return 4*6 = 24
    return 5*24 = 120
```

Propiedades

1. Existe al menos un caso base que devuelve un valor sin realizar llamadas recursivas.
   - Para la función `factorial()`, el caso base es n = 1.
   - Es importante esta etapa, de lo contrario la función se ejecutará eternamente.
2. Existe un set de reglas (paso de reducción) que reduce todos los otros casos al caso base:
   - Para la función `factorial()`, el paso de reducción es `n*factorial(n-1)` donde n disminuye en uno cada vez que llamamos a la función hasta llegar al caso base n = 1.
   - Esto también se conoce como caso recursivo.

Ejemplo: Conversión a Números Binarios

Para convertir un número entero a binario, hay que dividir el número decimal por dos y guardar el resultado. Si el resto de la division por 2 es 0 , se asigna un 0 y si es distinto de cero se asigna 1. Luego se vuelve a dividir el resultado por dos y se asigna 0 o 1 dependiendo del resultado del resto. Esto se realiza hasta que el resultado de la division sea >= 1. El numero binario corresponderá a la lista de 0s y 1s invertida al finalizar las divisiones.



28=11100_2

Conversión a Números Binarios

No usando recursividad

```
def int2bin(n):
    binarynumber=""

    if (n!=0):
        while (n>=1):
            if (n %2==0):
                binarynumber=binarynumber+"0"
                n=n/2
            else:
                binarynumber=binarynumber+"1"
                n=(n-1)/2

    else:
        binarynumber="0"

    return binarynumber[::-1] #Retornar el binarynumber invertido
```

Usando recursividad

```
def recursive_bin(n):
    if n == 0:
        return ''
    else:
        return recursive_bin(n//2) + str(n%2)
```