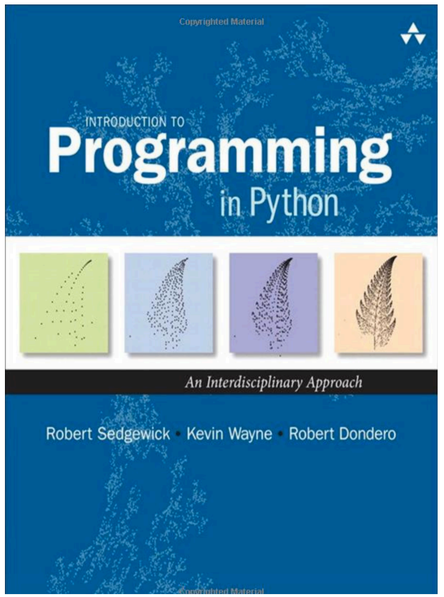
Tecnologías de la Información II

Clase 10: Funciones Recursivas

Daniela Opitz
dopitz@udd.cl



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

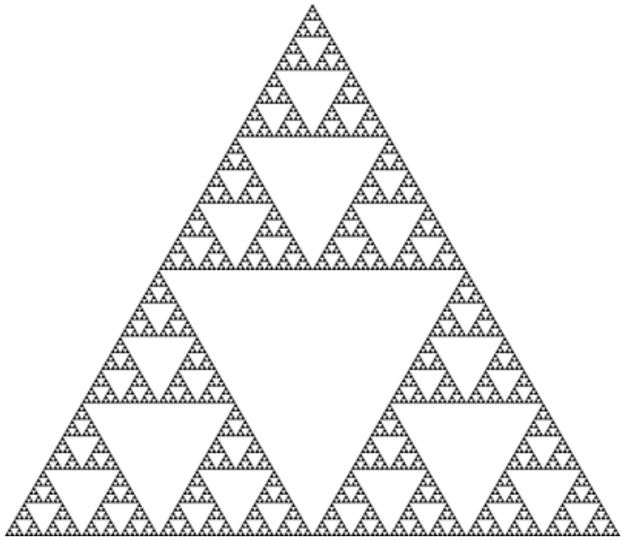
Disponible en https://introcs.cs.princeton.edu/python

Outline

- Tareas, ayudantías, certamen
- Funciones recursivas

Recursión o Recursividad

- En ciencias de la computación la recursion o recursividad es una 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.

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.

Ejemplo 2. Maximo Común Divisor

- El máximo común divisor (mcd) de dos enteros positivos es el entero más grande que se divide uniformemente en ambos. Por ejemplo, el mayor divisor común de 102 y 68 es 34, ya que tanto 102 como 68 son múltiplos de 34, pero ningún entero mayor que 34 se divide de manera uniforme en 102 y 68.

- Podemos calcular de manera eficiente el mcd usando la siguiente propiedad, que se mantiene para los enteros positivos p y q:

Si p> q, el mcd de p y q es el mismo que el mcd de q y p%q

Ejemplo 2. Máximo Común Divisor

```
def mcd(p, q):
    if q == 0:
        return p
    return mcd(q, p % q)

p = int(input('Ingrese un numero:'))
q = int(input('Ingrese un numero:'))

divisor = mcd(p, q)

print("El máximo comun divisor es:",
str(divisor))
```

```
mcd(1440, 408)
   return mcd(408, 216)
      return mcd(216, 24)
         return mcd(192, 24)
            return mcd(24, 0)
               return 24
```

Ejemplo 3. Fibonacci

La serie de **Fibonacci** es la sucesión infinita de números naturales tal que el primer y segundo termino son 0 y 1 respectivamente y a partir del tercero cada termino es la suma de los dos anteriores.

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

```
def fib(n):
    if n == 0:
        return 0
    if n == 1:
        return 1
    return fib(n-1) + fib(n-2)
```

Ejemplo 3. Fibonacci

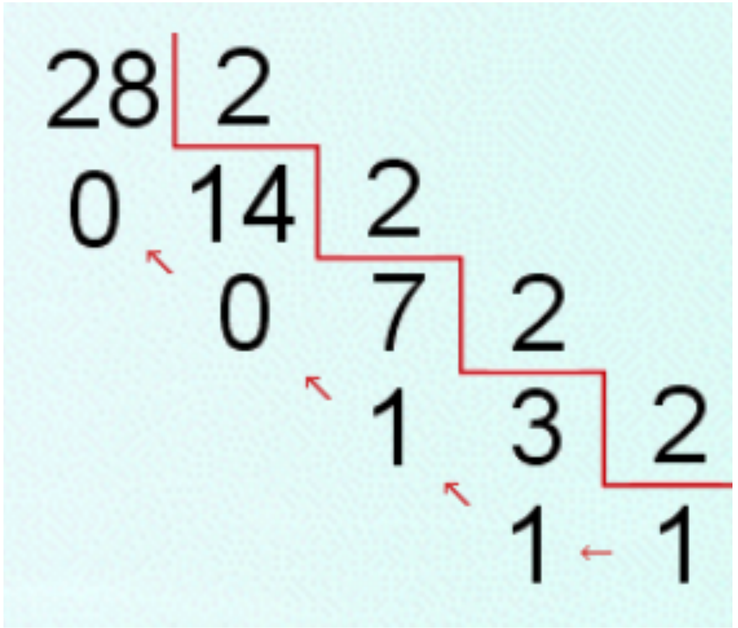
- El programa anterior es ineficiente!

- Por ejemplo, para **`fib(5) = 3`**, primero calcula **`fib(4)=2`** y **`fib(3)=1`**. Luego para calcular **`fib(4),`** debemos calcular **`fib(3)=1 y fib(2)=1`**. Ahora notamos que el programa computará **`fib(3)`**dos veces, para lo que tiene que calcular **`fib(2) y fib(1).`**

- Podemos hacer el mismo análisis anterior para **`fib(6),fib(7),…,fib(n)`** y notaremos que a medida que n crece, la cantidad de veces que el programa calcula **`fib(1)`** también crece. De hecho, cuando se calcula **`fib(n)`** la cantidad de veces que hay que calcular **`fib(1)`** es **fn**.

Ejemplo 4. 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)
```