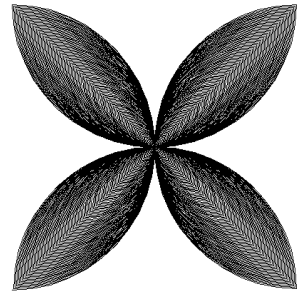
Recursão



Recursão

- ✓ Um objeto é dito recursivo se pode ser definido em termos de si próprio
  - *o que está sendo definido aparece como parte da definição*

Definição recursiva de x^y, para $y \in N$:

$$x^y = \begin{cases} 1, y = 0 \\ x * x^{y-1}, y > 0 \end{cases}$$

- ✓ Em toda definição recursiva, existe um caso base (ou caso trivial), cuja solução é conhecida e interrompe a recorrência

Definição recursiva

- ✓ Composta de duas partes:
  - a) No mínimo, um caso base
    - solução conhecida por definição, sem recursão
  - b) Um passo recursivo (ou indutivo) onde se resolve uma parte do problema (sub-problema)
    - de mesma natureza do problema original, porém menor

3

Exemplo

- ✓ Definição recursiva de x^y, com $y \in N$

$3^4 = 3 * \overbrace{3 * 3 * 3}^{3^3}$

$3^3 = 3 * \overbrace{3 * 3}^{3^2}$

$3^2 = 3 * \overbrace{3}^{3^1}$

$3^1 = 3 * \overbrace{1}^{3^0}$

$3^0 = 1$

$$x^y = \begin{cases} 1, y = 0 \\ x * x^{y-1}, y > 0 \end{cases}$$

Caso base? $x^0 = 1 \rightarrow y = 0$

Passo recursivo? $x^y = x * x^{(y-1)}$

4

Exemplo

- ✓ Definição recursiva de x^y, com $y \in N$

$3^4 = 3 * \overbrace{3 * 3 * 3}^{3^3}$

$3^3 = 3 * \overbrace{3 * 3}^{3^2}$

$3^2 = 3 * \overbrace{3}^{3^1}$

$3^1 = 3 * \overbrace{1}^{3^0}$

$3^0 = 1$

$$x^y = \begin{cases} 1, y = 0 \\ x * x^{y-1}, y > 0 \end{cases}$$

Relação de recorrência finita
- caso base tem solução conhecida
- não envolve recursão

Funções recursivas

- ✓ Resolvem o problema usando como método de solução a **recursão**
- ✓ No corpo de toda função recursiva há:
  - no mínimo, uma condição de parada (caso base)
  - chamada de uma cópia de si mesma entregando como argumento um problema mais simples do que o recebido
    - passo recursivo, convergindo para o caso base
    - a chamada recursiva é uma nova instância da função
- ✓ Instância da função
  - cópia com parâmetros e variáveis locais distintos
  - só os parâmetros/variáveis criados pela instância são visíveis

Funções recursivas

- ✓ Esquematicamente, as funções recursivas possuem o seguinte formato

```
se <condição para o caso base1>
    solução para o caso base1
senão se <condição para o caso de baseN>
    solução para o caso baseN
senão
    dividir instância em pelo menos uma chamada recursiva mais simples, convergindo para o caso base
    retornar combinação de resultado das chamadas recursivas
```

7

Exemplo

- ✓ Função que calcula x^y, com $y \in N$

$$x^y = \begin{cases} 1, y = 0 \\ x * x^{y-1}, y > 0 \end{cases}$$

```
def potencia (x, y):
    if y == 0:
        return 1
    return x * potencia (x, y - 1)
```

8

Exemplo - fatorial

$$n! = \begin{cases} 1, \text{ se } n = 0 \text{ ou } n = 1 \\ n \times (n-1)!, \text{ se } n > 1 \end{cases}$$

caso base

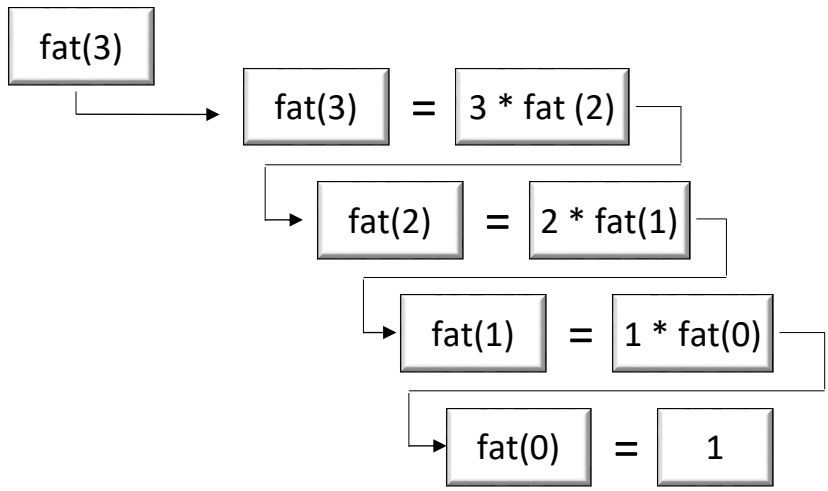
passo recursivo

```
def fat (n):
    if n == 0 or n == 1:
        return 1
    fatAnt = fat (n - 1)
    return n * fatAnt
```

```
def fat (n):
    if n == 0 or n == 1:
        return 1
    return n * fat (n - 1)
```

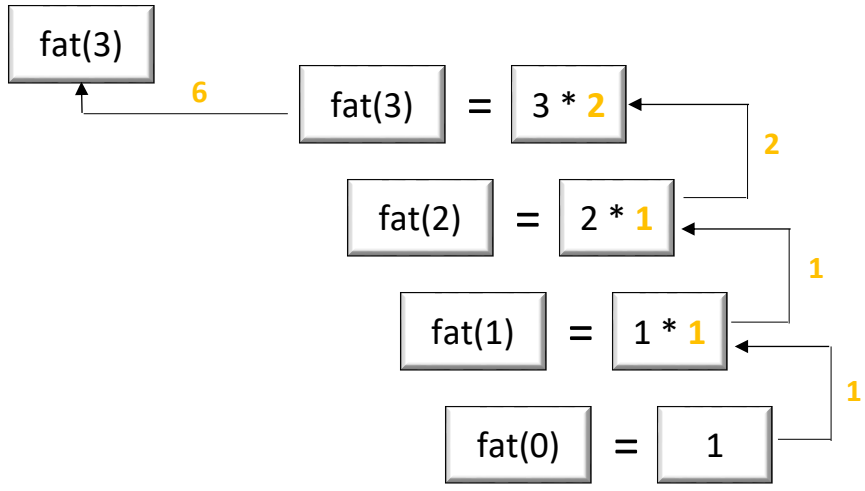
9

Executando o fatorial de 3



10

Executando o fatorial de 3



11

Roteiro recursivo

- ✓ Para se aplicar a **recursão**, deve-se pensar na definição recursiva do problema

1. Caso base (regra de parada)
   - Qual é o caso base? Há mais de um?
   - Como reconhecer o caso base?
   - O que deve ser feito no caso base?
2. Passo Recursivo
   - Como dividir o problema em problemas menores de mesma natureza?
   - O que deve mudar na chamada recursiva para se aproximar do caso base?
3. O que deve ser feito após a chamada recursiva da função, usando o valor de retorno da função (se houver)?

12

DEPARTAMENTO DE INFORMÁTICA PUC-RIO

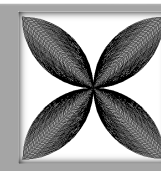
Curiosidade

- ✓ Programa que gerou a imagem do primeiro slide

```
import turtle

def desenha_folhas(tart, raio):
    if raio <= 0:
        return
    tart.circle(raio, 180)
    tart.right(270)
    desenha_folhas(tart, raio - 0.5)

flash = turtle.Turtle()
flash.speed(0)
desenha_folhas(flash, 170)
```



13