Introdução ao Processamento de Dados
Turma 3 (2020.1)

Recursividade e Tratamento de Erros

Gilson. A. O. P. Costa (IME/UERJ)

gilson.costa@ime.uerj.br

Recursividade

- Na ciência da computação, **recursão** é um método de resolver um problema onde a solução depende de soluções para **instâncias menores do mesmo problema**.
- Estes **problemas recursivos** geralmente podem ser resolvidos por iteração, o que pode ser mais custoso computacionalmente.
- A recursão resolve esses problemas usando **funções que chamam a si mesmas** de dentro de seu próprio código.
- A abordagem pode ser aplicada a muitos tipos de problemas, e a recursão é uma das **ideias centrais da ciência da computação**.

Recursividade

- Uma função é recursiva quando **chama a si própria**.
- Muito utilizado na técnica de programação de **divisão e conquista**.
- Possui dois passos básicos:
  - **Reduzir o problema** a outro de menor instância.
  - Até chegar à **solução trivial**.

Recursividade

Exemplo: calcular x^y

- Solução trivial: $x^0 = 1$

```
def potencia(x, y):
    if y == 0:
        return 1; # solução trivial
    else:
        return x * potencia(x,y-1); # recursão

x = int(input("Entre com a base: "))
y = int(input("Entre com a potência: "))
print("Potência: ", potencia(x,y))
```

Recursividade

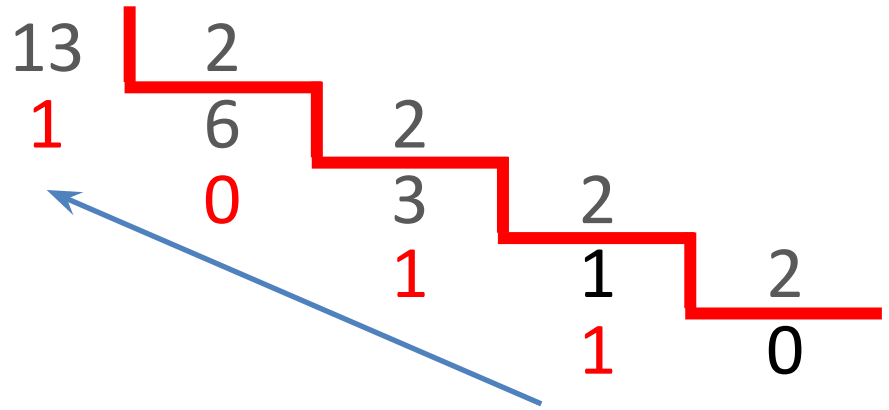
- Exemplo: calcular o fatorial de um número, e.g., $4! = 4 \times 3 \times 2 \times 1$
- Solução trivial: $1! = 1$

```
def fatorial(n):
    if n == 1:
        return 1; # solução trivial
    else:
        return n * fatorial(n-1); # recursão

n = input(“Entre um número: ")
print(“Fatorial: ", fatorial(n))
```

Conversão entre Bases

- Para converter um número da base decimal para qualquer outra base: **divisões sucessivas**.
- Para converter o decimal 13_{10} em binário: $1\,1\,0\,1_2$



Tratamento de Exceção

- É usado para controlar o que acontece se houver algum erro de execução no programa.
- Sintaxe:

```
try:
    bloco_instrucoes1
except:
   bloco_instrucoes2
finally:
   bloco_instrucoes3
```

Tratamento de Exceção

- O bloco dentro do comando ***try*** (tente) contém instruções que o programador imagina que possam dar erro.
- Sintaxe:

```
try:
    bloco_instrucoes1
except:
   bloco_instrucoes2
finally:
   bloco_instrucoes3
```

Tratamento de Exceção

- O bloco dentro do comando ***except*** (exceção) contém instruções que são executadas se houver algum erro no bloco do **try**.
- Sintaxe:

```
try:
    bloco_instrucoes1
except:
   bloco_instrucoes2
finally:
   bloco_instrucoes3
```

Tratamento de Exceção

- O comando ***finally*** (finalmente) é opcional, o respectivo bloco de instruções será sempre executado.
- Sintaxe:

```
try:
    bloco_instrucoes1
except:
   bloco_instrucoes2
finally:
   bloco_instrucoes3
```

Tratamento de Exceção

Exemplo:

```
try:
    num = int(input('Entre com um número: '))
except:
  print('Número inválido!')
```

Tratamento de Exceção

- O comando ***raise*** (levantar exceção) pode ser usado para provocar uma exceção.
- Exemplo:

```
try:
    num = int(input('Entre com um número menor que 10: '))
    if num>=10:
        raise
except:
   print('Número inválido!')
```

Tratamento de Exceção

Exemplo: programa para ler dois números e imprimir a sua divisão.

```
try:
    x = float(input("Entre com um num.: "))
    y = float(input("Entre com outro num.: "))
    divisao = x/y
    print("Divisão: ", divisao)
except:
    print("Erro!")
```

Tratamento de Exceção

Exemplo: programa para ler dois números e imprimir a sua divisão.

```
try:
    x = float(input("Entre com um num.: "))
    y = float(input("Entre com outro num.: "))
    try:
        divisao = x/y
        print("Divisão: ", divisao)
    except:
        print("Erro na divisão!")
except:
    print("Número inválido!")
```

Introdução ao Processamento de Dados
Turma 3 (2020.1)

Recursividade e Tratamento de Erros

Gilson. A. O. P. Costa (IME/UERJ)

gilson.costa@ime.uerj.br