Laboratório de Programação 1

Aula 05

Mário Hozano
professor@hozano.com

Ciência da Computação
UFAL - Arapiraca

Relembrando a aula anterior...

- O que são funções?
- O que são argumentos e parâmetros?
- Para que serve as funções *int()*, *float()* e *str()* ?
- O que é módulo? Como se importa um módulo?
- Para que serve as funções *print()*, *type()* e *input()*?
- Como se define uma função?
- O que são variáveis locais?

Roteiro da aula

- Expressões Lógicas
- Estruturas Condicionais
- Condicional Simples
- Condicional Composto
- Condicional Encadeado
- Condicional Aninhado
- Recursão

Expressões Lógicas

- Uma expressão lógica (ou booleana) é uma expressão que pode ter valor Verdadeiro ou Falso

- Em Python, os valores lógicos são determinados com o tipo *bool* pelos valores *True* e *False* (sem aspas)

- Abaixo seguem algumas expressões lógicas utilizando o operador que compara se dois valores são iguais (==)

```
>>> 5 == 5
True
>>> 5 == 6
False
>>> type(True)
<type 'bool'>
```

Expressões Lógicas – Operadores Relacionais

- Uma expressão lógica pode conter operadores relacionais e lógicos

- Os operadores relacionais abaixo permitem identificar uma relação lógica entre dois elementos

Operadores	Descrição
x == y	x é igual a y
x != y	x é diferente de y
x > y	x maior que y
x < y	x menor que y
x >= y	x maior ou igual a y
x <= y	x menor ou igual a y

Expressões Lógicas – Operadores Relacionais

- Considerando x = 7 e y = 9 teríamos

```
>>> x == y
False
>>> x != y
True
>>> x > y
False
>>> x < y
True
>>> x > 7
False
>>> x >= 7
True
```

Expressões Lógicas – Operadores Lógicos

- Além de operadoes relacionais, uma expressão lógica pode conter operadores lógicos
- Os operadores lógicos permitem agrupar condições lógicas para analisar um único resultado
- Em Python, os operadores lógicos são:

Operadores	Descrição
not	Negação (Não)
and	Conjunção (E)
or	Disjunção (Ou)

Expressões Lógicas – Operadores Relacionais

- Considerando x = 7 e y = 9 teríamos

```
>>> x > 5 and x < 9
True
>>> x == 9 or y == 7
False
>>> x == 9 or y == 9
True
>>> not (x > 7)
True
>>> (x > 7) and not (y == 7)
False
```

Estruturas Condicionais

- Em programas simples é comum a avaliação de expressões lógicas (condição) para alterar o fluxo de execução
- Em algoritmos e programação isto é obtido através das estruturas de seleção ou condicionais
- Em Python, estas estruturas são criadas utilizando o comando *if*, de acordo com a sintaxe a seguir

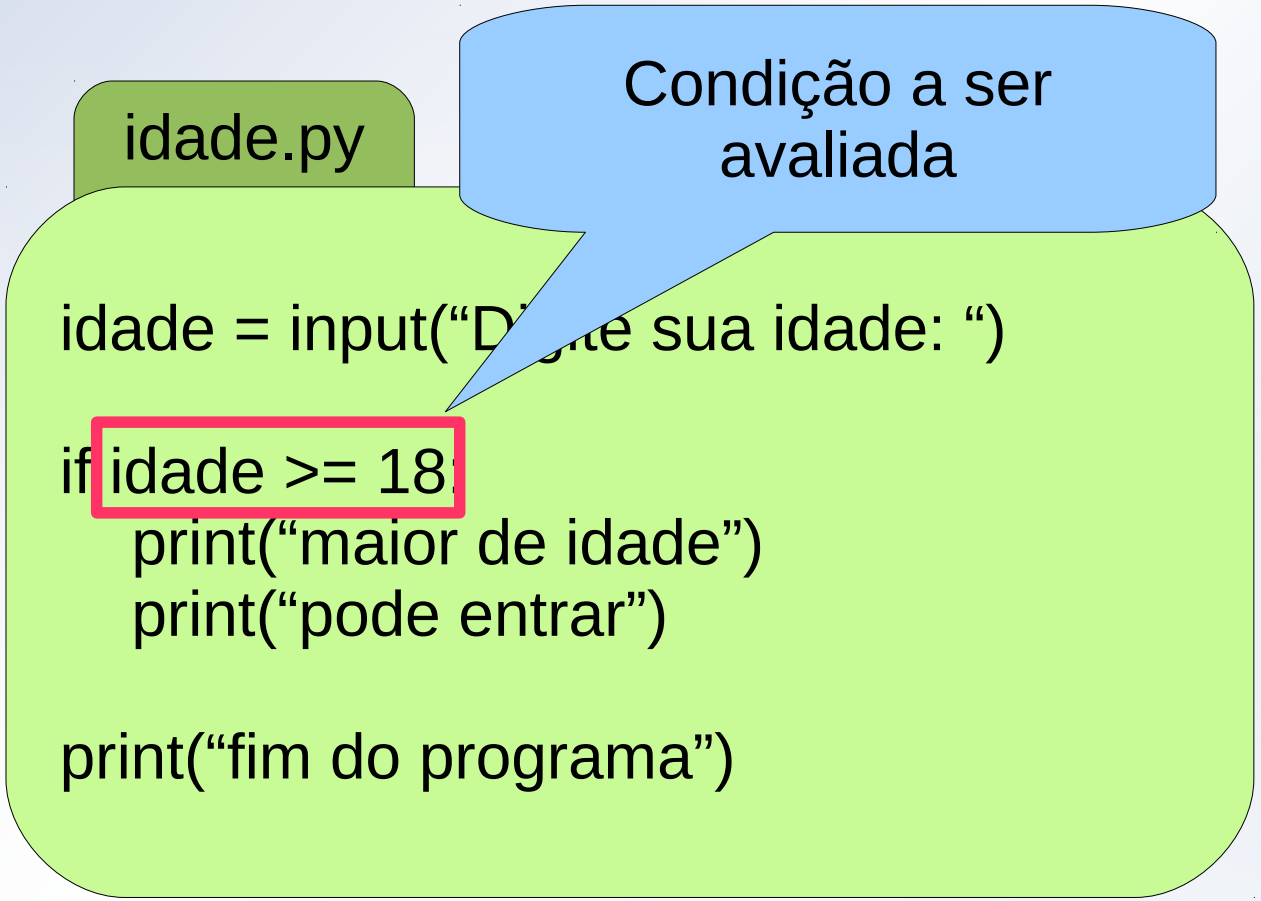
```
if <condição>:
    <fluxo condicional>
```

Condicional Simples

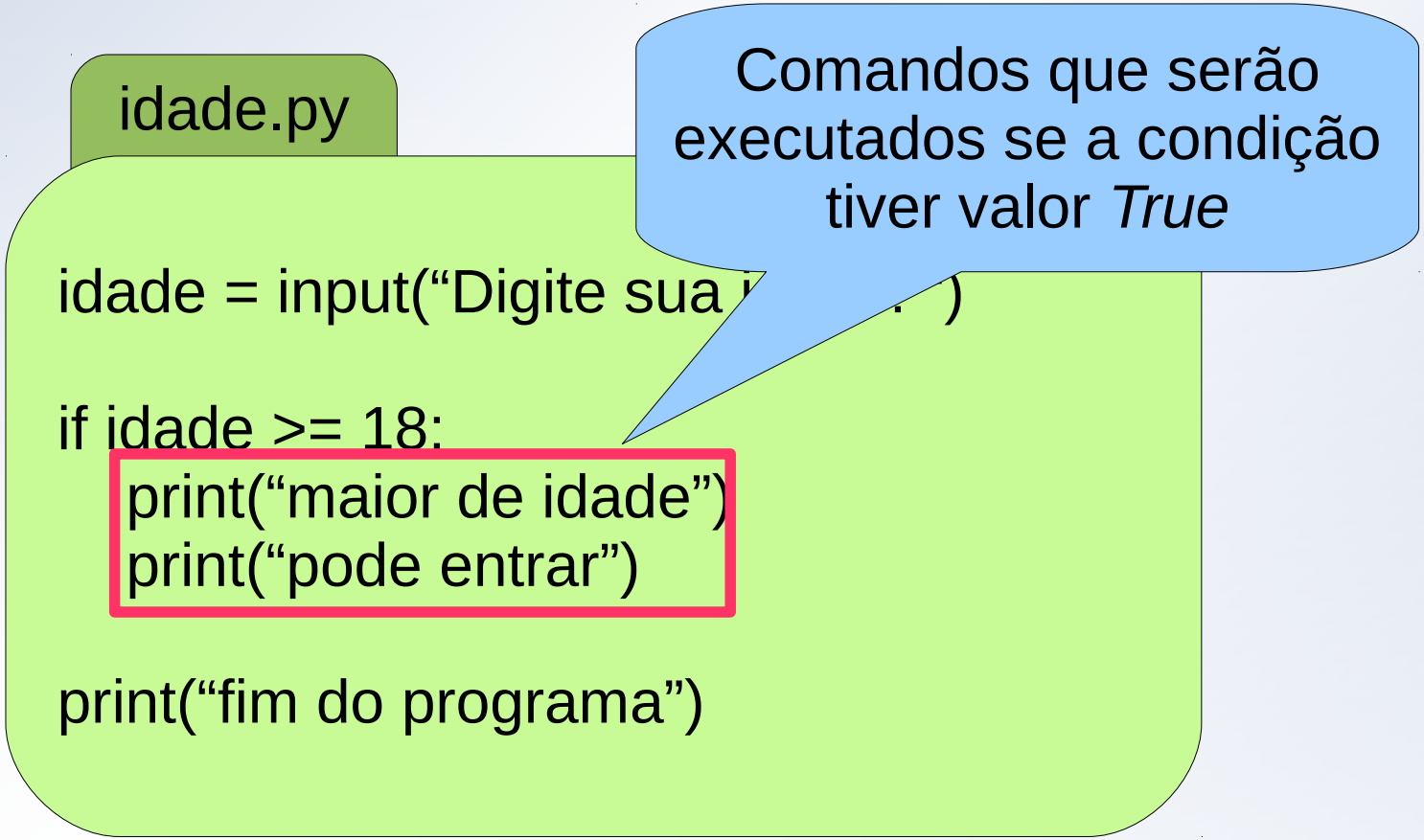
idade.py

```
idade = input(“Digite sua idade: “)

if idade >= 18:
    print(“maior de idade”)
    print(“pode entrar”)

print(“fim do programa”)
```

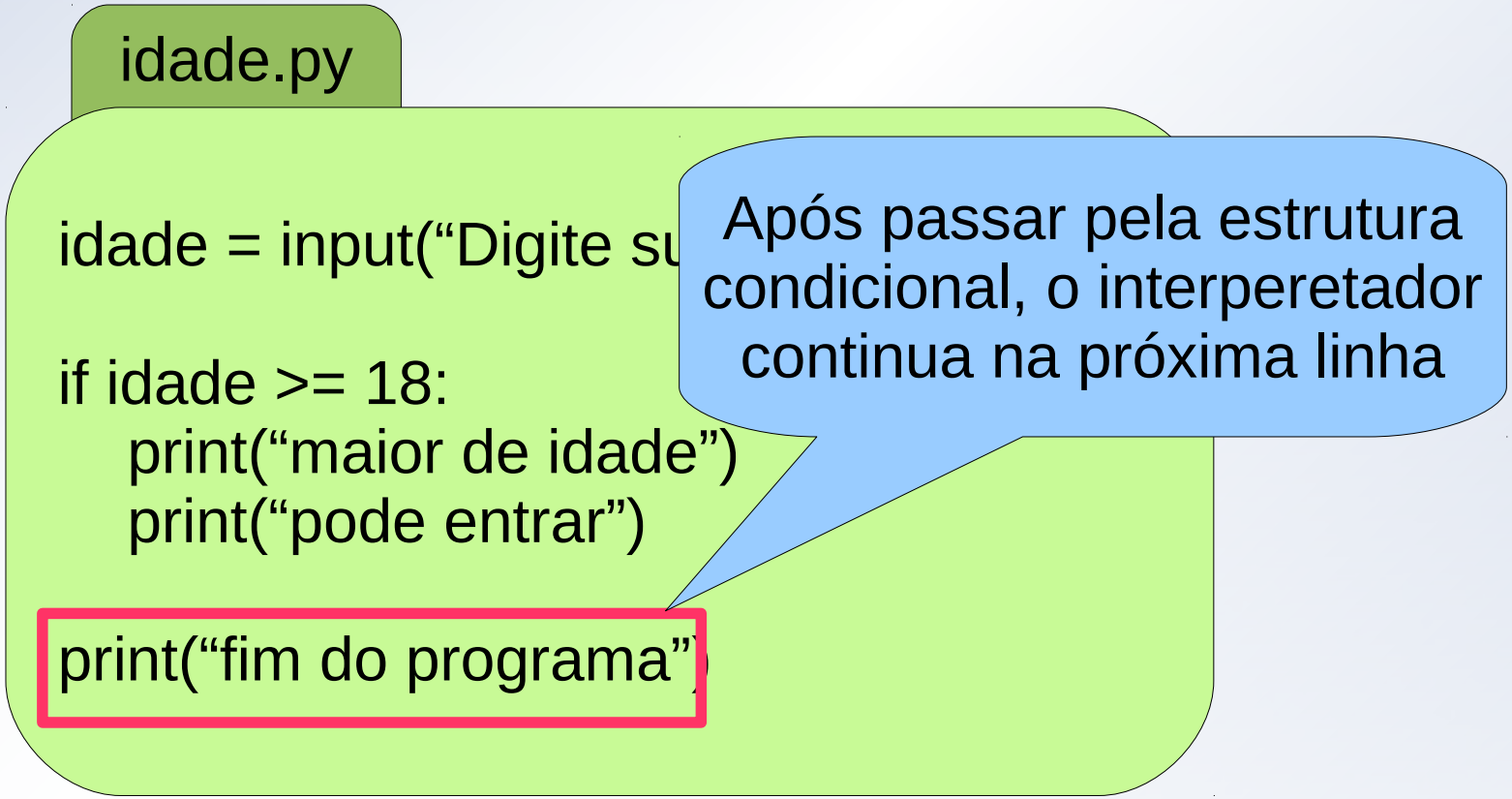
Condicional Simples



Condicional Simples



Condicional Simples



Condicional Simples

Como alterar o programa para ele também exibir mensagens caso *idade < 18* ?

idade.py

```
idade = input(“Digite sua idade: “)

if idade >= 18:
    print(“maior de idade”)
    print(“pode entrar”)

print(“fim do programa”)
```

Condicional Composto

- O Condicional Composto permite alterar o fluxo do programa com duas **execuções alternativas**
- Ele determina uma entre duas alternativas de fluxo dependendo da avaliação da condição dada
- Sua sintaxe acrescenta a palavra-chave *else* (se não), como descrito abaixo

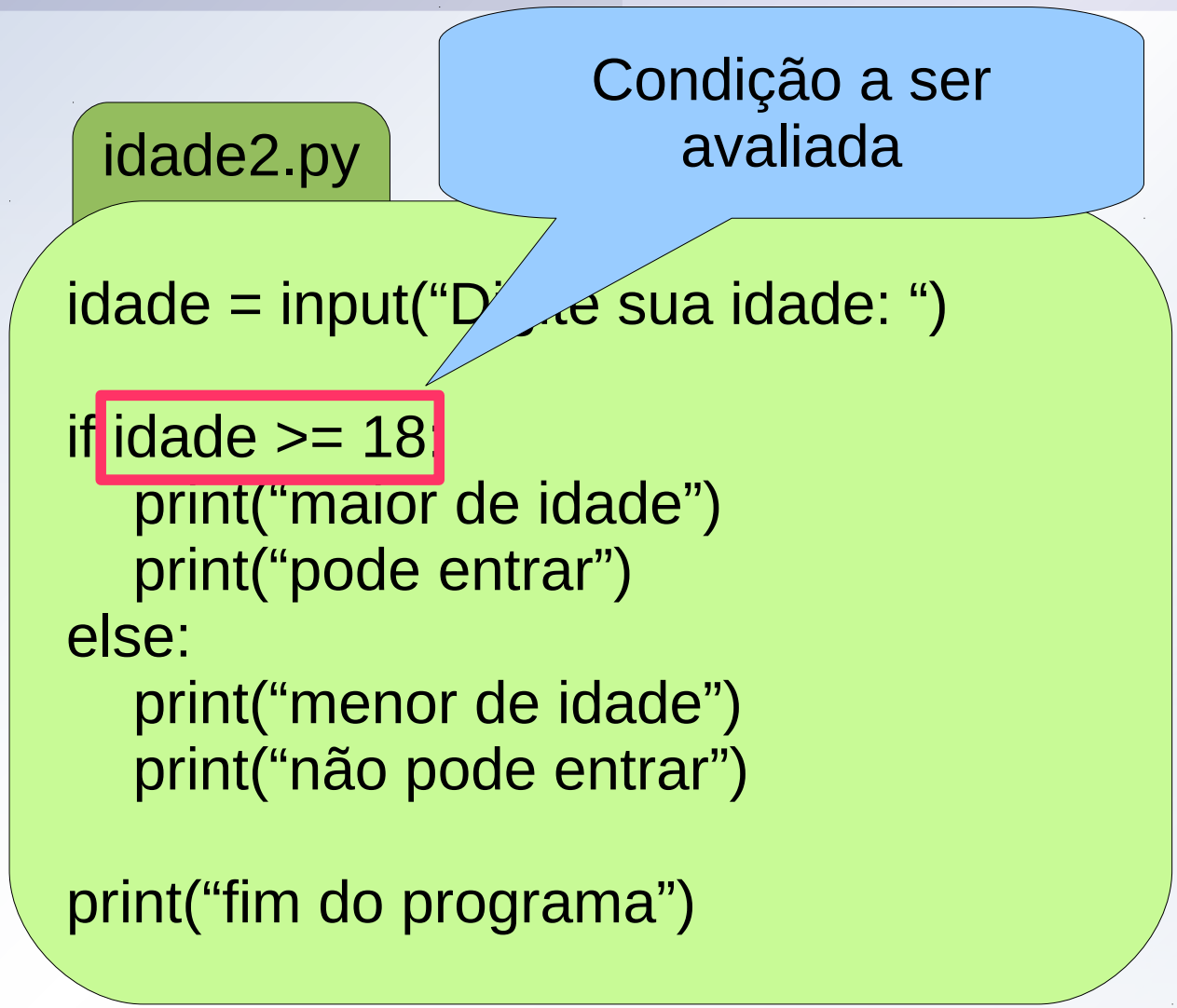
```
if <condição>:
    <alternativa 1>
else:
    <alternativa 2>
```

Condicional Composto

idade2.py

```
idade = input(“Digite sua idade: “)

if idade >= 18:
    print(“maior de idade”)
    print(“pode entrar”)
else:
    print(“menor de idade”)
    print(“não pode entrar”)

print(“fim do programa”)
```

Condicional Composto



Condicional Composto

idade2.py

```
idade = input("Digite sua idad

if idade >= 18:
    print("maior de idade")
    print("pode entrar")
else:
    print("menor de idade")
    print("não pode entrar")

print("fim do programa")
```

Comandos que serão executados se a condição tiver valor *True*

Condicional Composto

idade2.py

```
idade = input("Digite sua idad

if idade >= 18:
    print("maior de idade")
    print("pode entrar")
else:
    print("menor de idade")
    print("não pode entrar")

print("fim do programa")
```

Comandos que serão executados se a condição tiver valor *True*

Comandos que serão executados se a condição tiver valor *False*

Condicional Composto

idade2.py

```
idade = input(“Digite sua idade: “)

if idade >= 18:
    print(“maior de idade
    print(“pode entrar”)
else:
    print(“menor de idade”)
    print(“não pode entrar”)

print(“fim do programa”)
```

Após passar pela estrutura condicional, o interperetador continua na próxima linha

Condicional Composto

Como alterar o programa para ele também exibir mensagens caso 16 < *idade* < *18* ?

idade2.py

```
idade = input(“Digite sua idade: “)

if idade >= 18:
    print(“maior de idade”)
    print(“pode entrar”)
else:
    print(“menor de idade”)
    print(“não pode entrar”)

print(“fim do programa”)
```

Condicional Encadeado

- O Condicional Encadeado permite alterar o fluxo do programa com **várias execuções alternativas**
- Ele determina **uma** entre as alterativas de fluxo dependendo da avaliação das condições dadas
- Sua sintaxe acrescenta a palavra-chave *elif* (contração de *else if*), como descrito abaixo

```
if <condição 1>:
    <alternativa 1>
elif <condição 2>:
    <alternativa 2>
elif <condição 3>:
    <alternativa 3>
else:
    <alternativa 4>
```

Condicional Encadeado

- O Condicional Encadeado permite alterar o fluxo do programa com **várias execuções alternativas**
- Ele determina **uma** entre as alterativas de fluxo dependendo da avaliação das condições dadas
- Sua sintaxe acrescenta a palavra-chave *elif* (contração de *else if*), como descrito abaixo

```
if <condição 1>:
    <alternativa 1>
elif <condição 2>:
    <alternativa 2>
elif <condição 3>:
    <alternativa 3>
else:
    <alternativa 4>
```

O bloco *else* é opcional

Condicional Encadeado

idade3.py

```
idade = input(“Digite sua idade: “)

if idade >= 18:
    print(“maior de idade”)
    print(“pode entrar”)
elif idade >= 16:
    print(“menor de idade”)
    print(“pode entrar com acompanhante maior”)
else:
    print(“menor de idade”)
    print(“não pode entrar”)

print(“fim do programa”)
```

Condicional Encadeado

idade3.py

Condição 1

```
idade = input("Digite sua idade: ")

if idade >= 18:
    print("maior de idade")
    print("pode entrar")
elif idade >= 16:
    print("menor de idade")
    print("pode entrar com acompanhante maior")
else:
    print("menor de idade")
    print("não pode entrar")

print("fim do programa")
```

Condicional Encadeado

idade3.py

Condição 1

Comandos que serão executados se a Condição 1 tiver valor *True*

```
idade = input("Digite sua ida

if idade >= 18:
    print("maior de idade")
    print("pode entrar")
elif idade >= 16:
    print("menor de idade")
    print("pode entrar com acompanhante maior")
else:
    print("menor de idade")
    print("não pode entrar")

print("fim do programa")
```

Condicional Encadeado

idade3.py

```
idade = input(“Digite sua idade: “)

if idade >= 18:
    print(“maior de idade”)
    print(“pode entrar”)
elif idade >= 16:
    print(“menor de idade”)
    print(“pode entrar com acompanhante maior”)
else:
    print(“menor de idade”)
    print(“não pode entrar”)

print(“fim do programa”)
```

Condição 2

Condicional Encadeado

idade3.py

```
idade = input(“Digite sua idade: “)

if idade >= 18:
    print(“maior de idade”)
    print(“pode entrar”)
elif idade >= 16:
    print(“menor de idade”)
    print(“pode entrar com acompanhante maior”)
else:
    print(“menor de idade”)
    print(“não pode entrar”)

print(“fim do programa”)
```

Condição 2

Comandos que serão executados se a Condição 2 tiver valor *True*

Condicional Encadeado

idade3.py

```
idade = input(“Digite sua idade: “)

if idade >= 18:
    print(“maior de idade”)
    print(“pode entrar”)
elif idade >= 16:
    print(“menor de ida
    print(“pode entrar com a
else:
    print(“menor de idade”)
    print(“não pode entrar”)

print(“fim do programa”)
```

Comandos que serão executados caso nenhum condição seja *True*

Condicional Aninhado

- Estruturas condicionais podem ser utilizadas dentro de outros condicionais

- Quando isto acontece diz-se que utilizou-se condicionais (ou *if*s) aninhandos

```
if x == y:
    print(“x é igual a y”)
else:
    if x > y:
        print(“x é maior que y”)
    else:
        print(“x é menor que y”)
```

Condicional Aninhado

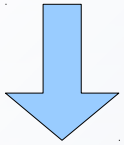
- Por ser de difícil compreensão, os condicionais aninhandos devem ser evitados, sempre que possível
- Condições com operadores lógicos podem evitar condicionais aninhados como no exemplo abaixo

```
if media < 7:
    if media >= 5:
        print(“Pode fazer prova final”)
```

Condicional Aninhado

- Por ser de difícil compreensão, os condicionais aninhandos devem ser evitados, sempre que possível
- Condições com operadores lógicos podem evitar condicionais aninhados como no exemplo abaixo

```
if media < 7:
    if media >= 5:
        print(“Pode fazer prova final”)
```



```
if media >= 5 and media < 7:
    print(“Pode fazer prova final”)
```

Recursão

- Sabe-se que uma função pode chamar outra
- Em programação também é possível que uma função contenha comandos que chame-a novamente
- Apesar de parecer estranho, esta prática pode ser bastante útil em algumas situações
- Este processo é chamdo de recursão e as funções deste tipo são chamadas de **funções recursivas**
- Funções recursivas apresentam condicionais em seu corpo que indicam uma situação de parada

Recursão - Exemplo

- A função abaixo realiza uma contagem regressiva a partir de um número inteiro dado como argumento

```
def contagem_regressiva(n):
    if n <= 0:
        print(“BOOOMMM!!!”)
    else:
        print(n)
        contagem_regressiva(n - 1)
```

Recursão - Exemplo

- A função abaixo realiza uma contagem regressiva a partir de um número inteiro dado como argumento

Condição de Parada: quando for chamada deixa de chamar a função recursivamente

```
def contagem_r
    if n <= 0:
        print("BOOOMMM!!!")
    else:
        print(n)
        contagem_regressiva(n - 1)
```

Recursão - Exemplo

- A função abaixo realiza uma contagem regressiva a partir de um número inteiro dado como argumento

Condição de Parada: quando for chamada deixa de chamar a função recursivamente

```
def contagem_r
    if n <= 0:
        print("BOOOMMM!!!")
    else:
        print(n)
        contagem_regressiva(n - 1)
```

O que acontece se chamarmos o comando abaixo ?
>>> contagem_regressiva(3)

Recursão - Exemplo

```
def contagem_regressiva(n):
    if n <= 0:
        print(“BOOOMMM!!!”)
    else:
        print(n)
        contagem_regressiva(n – 1)

contagem_regressiva(3)
```

Recursão - Exemplo

O interpretador lê o nome da função e passa para a próxima linha após a função

```
def contagem_regressiva(n):
    if n <= 0:
        print(“BOOOMMM!!!”)
    else:
        print(n)
        contagem_regressiva(n – 1)

contagem_regressiva(3)
```

Recursão - Exemplo

```
def contagem_regressiva(n):
    if n <= 0:
        print("BOOOMMM
    else:
        print(n)
        contagem_regressiva
                          - 1)

contagem_regressiva(3)
```

A função é chamada com argumento igual a 3

Recursão - Exemplo

A função é iniciada com n = 3

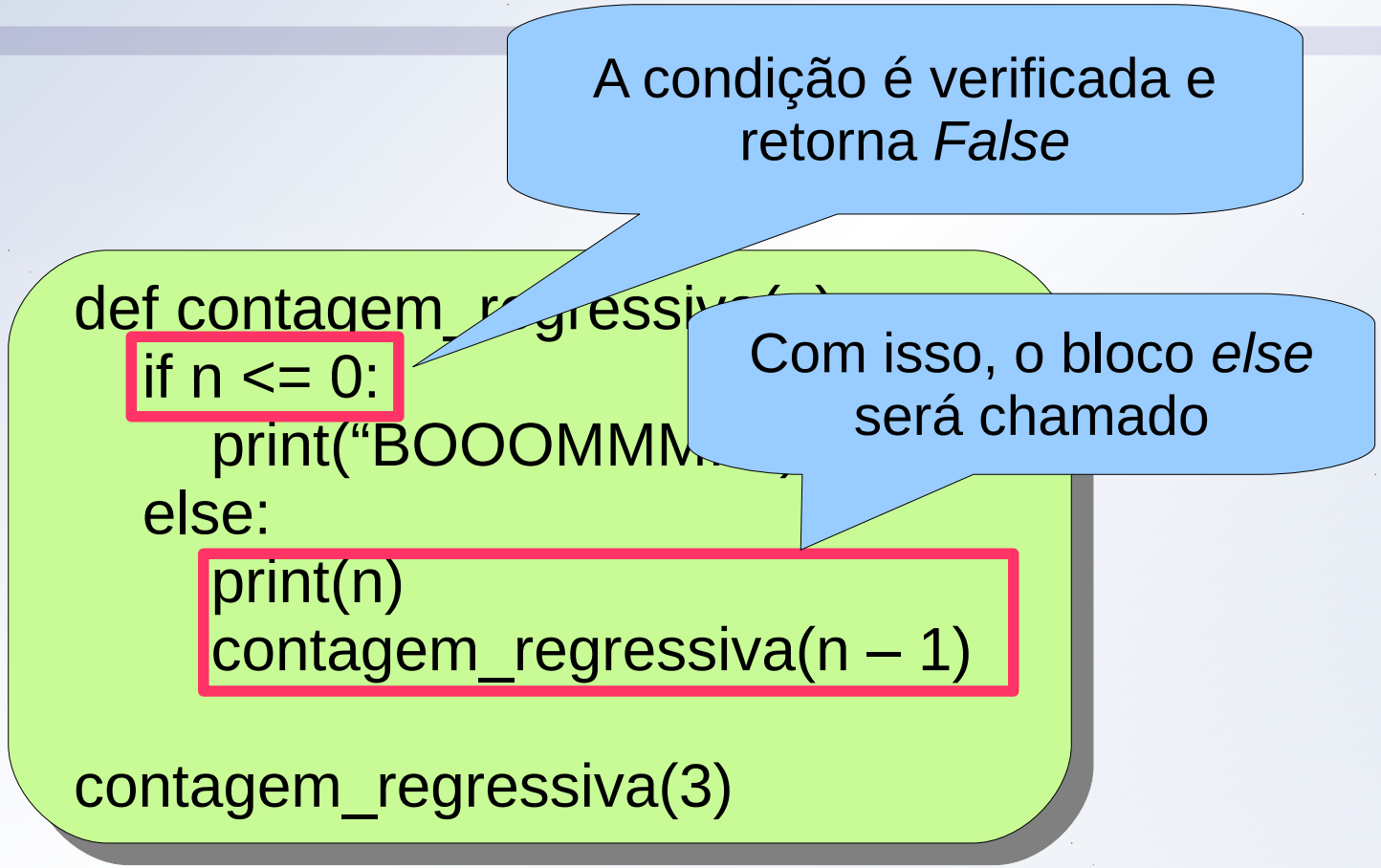
```
def contagem_regressiva(n):
    if n <= 0:
        print(“BOOOMMM!!!”)
    else:
        print(n)
        contagem_regressiva(n – 1)

contagem_regressiva(3)
```

Recursão - Exemplo

A condição é verificada e retorna *False*

```
def contagem_regressiva(n):
    if n <= 0:
        print(“BOOOMMM!!!”)
    else:
        print(n)
        contagem_regressiva(n – 1)

contagem_regressiva(3)
```

Recursão - Exemplo



Recursão - Exemplo

```
def contagem_r
    if n <= 0:
        print("BOOOM
    else:
        print(n)
        contagem_regressiva(n – 1)

contagem_regressiva(3)
```

O valor de *n* é impresso na tela (3)

Recursão - Exemplo

```
def contagem_r
    if n <= 0:
        print("BOO
    else:
        print(n)
        contagem_regressiva(n – 1)

contagem_regressiva(3)
```

A função é chamada com argumento igual a 2

Recursão - Exemplo

A função é iniciada novamente com n = 2

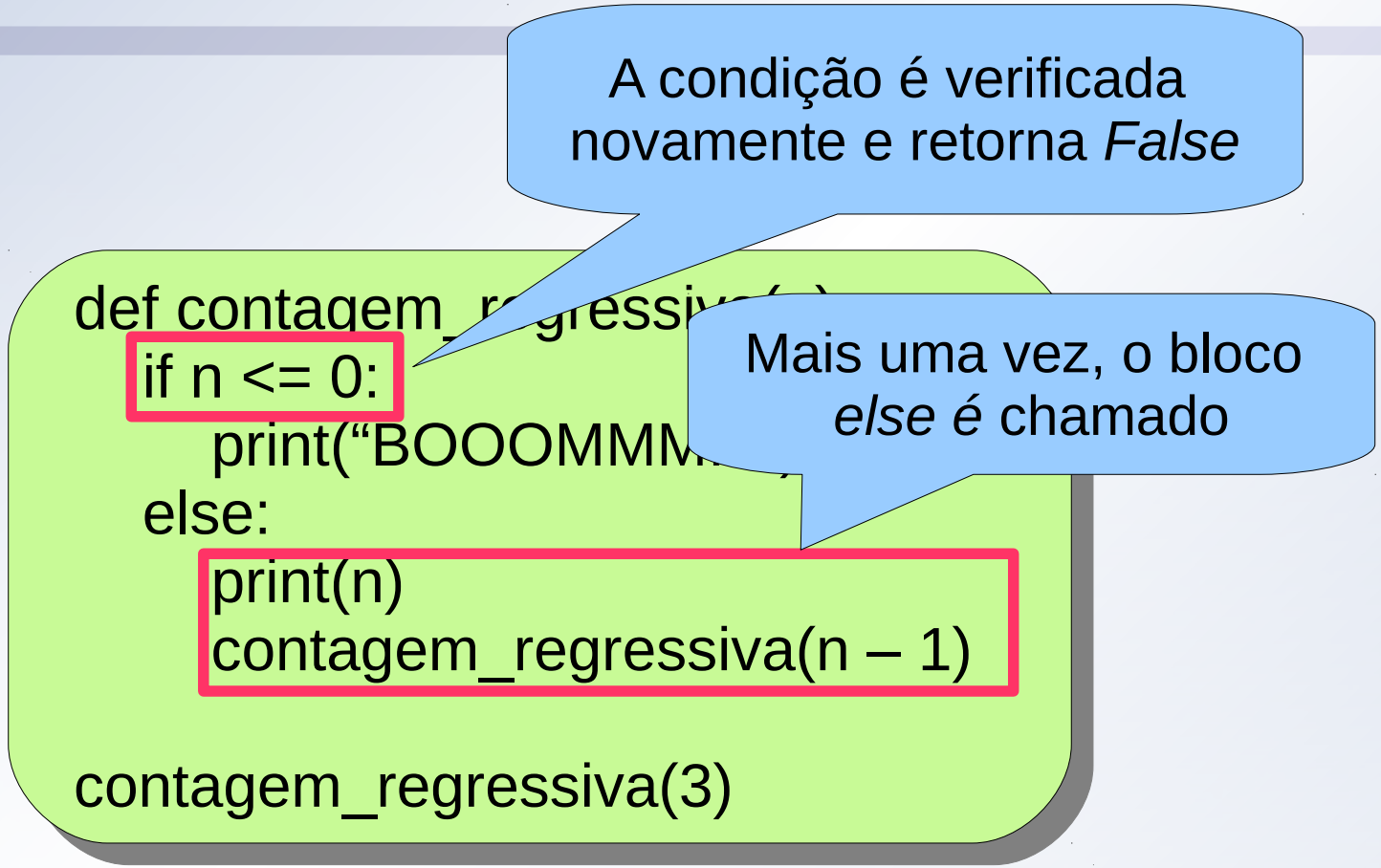
```
def contagem_regressiva(n):
    if n <= 0:
        print(“BOOOMMM!!!”)
    else:
        print(n)
        contagem_regressiva(n – 1)

contagem_regressiva(3)
```

Recursão - Exemplo

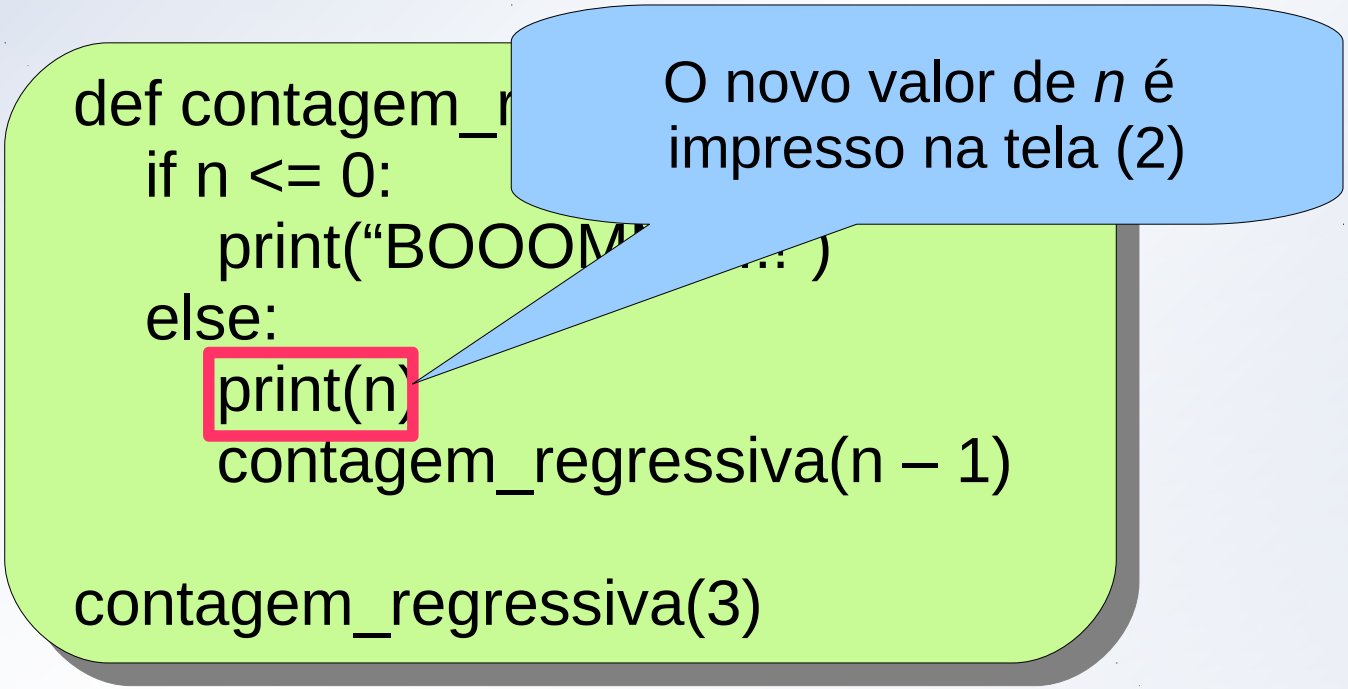
A condição é verificada novamente e retorna *False*

```
def contagem_regressiva(n):
    if n <= 0:
        print(“BOOOMMM!!!”)
    else:
        print(n)
        contagem_regressiva(n – 1)

contagem_regressiva(3)
```

Recursão - Exemplo



Recursão - Exemplo



Recursão - Exemplo

```
def contagem_r
    if n <= 0:
        print("BOO
    else:
        print(n)
        contagem_regressiva(n – 1)

contagem_regressiva(3)
```

A função é chamada novamente com argumento igual a 1

Recursão - Exemplo

A função é iniciada novamente com n = 1

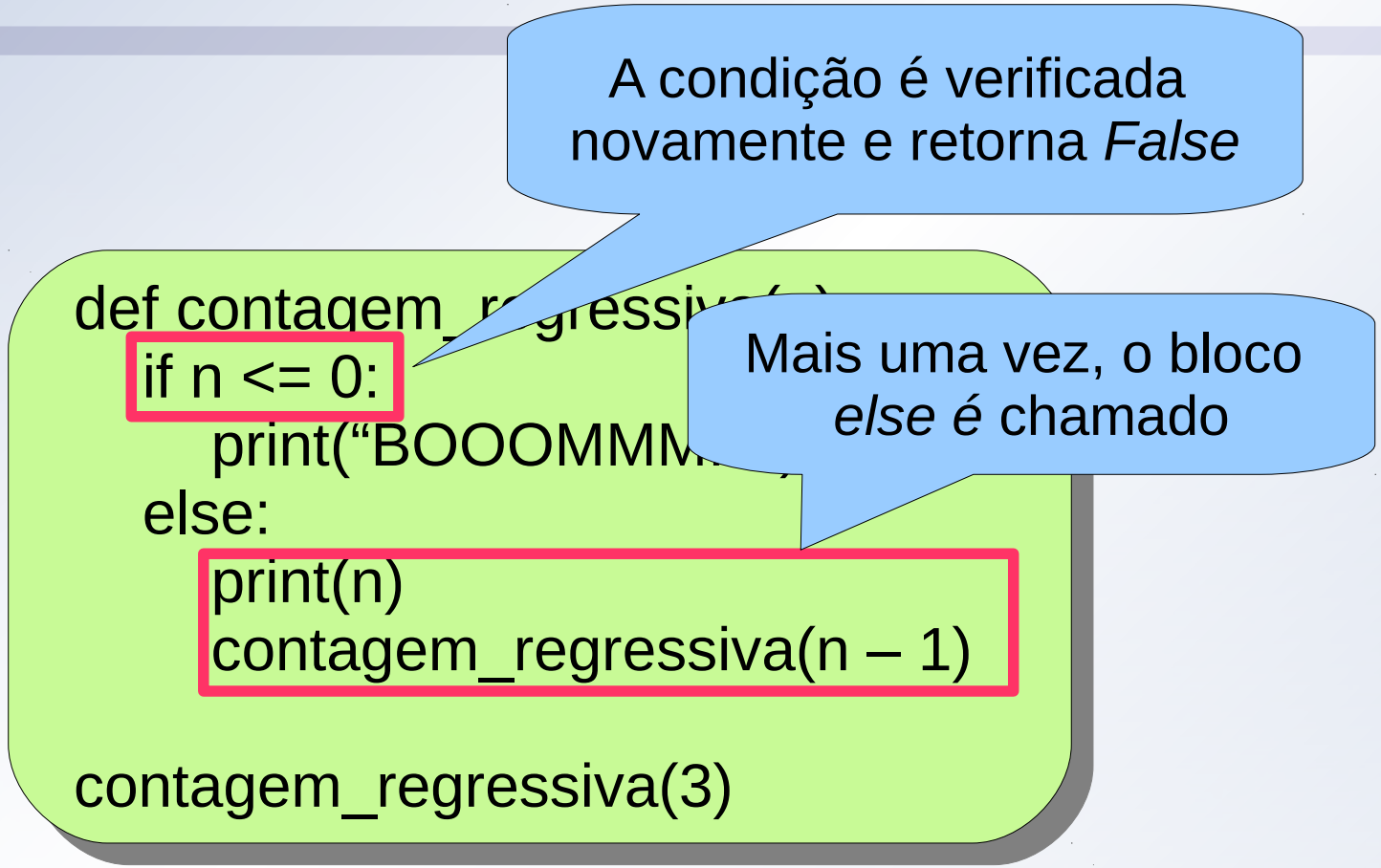
```
def contagem_regressiva(n):
    if n <= 0:
        print(“BOOOMMM!!!”)
    else:
        print(n)
        contagem_regressiva(n – 1)

contagem_regressiva(3)
```

Recursão - Exemplo

A condição é verificada novamente e retorna *False*

```
def contagem_regressiva(n):
    if n <= 0:
        print("BOOOMMM!!!")
    else:
        print(n)
        contagem_regressiva(n – 1)

contagem_regressiva(3)
```

Recursão - Exemplo



Recursão - Exemplo

```
def contagem_r
    if n <= 0:
        print("BOOOM
    else:
        print(n)
        contagem_regressiva(n – 1)

contagem_regressiva(3)
```

O novo valor de *n* é impresso na tela (1)

Recursão - Exemplo

```
def contagem_r
    if n <= 0:
        print("BOO
    else:
        print(n)
        contagem_regressiva(n – 1)

contagem_regressiva(3)
```

A função é chamada novamente com argumento igual a 0

Recursão - Exemplo

A função é iniciada novamente com n = 0

```
def contagem_regressiva(n):
    if n <= 0:
        print(“BOOOMMM!!!”)
    else:
        print(n)
        contagem_regressiva(n – 1)

contagem_regressiva(3)
```

Recursão - Exemplo

A condição é verificada novamente e desta vez retorna *True*

```
def contagem_regressiva(n):
    if n <= 0:
        print(“BOOOMMM!!!”)
    else:
        print(n)
        contagem_regressiva(n – 1)

contagem_regressiva(3)
```

Recursão - Exemplo

A condição é verificada novamente e desta vez retorna *True*

```
def contagem_regressiva(n):
    if n <= 0:
        print("BOOOMMM!!!")
    else:
        print(n)
        contagem_regressiva(n

contagem_regressiva(3)
```

Desta vez, o bloco *if será* chamado

Recursão - Exemplo

A mensagem é impressa na tela e o fluxo desta chamada da função é encerrado

```
def contagem_regressiva(n):
    if n <= 0:
        print(“BOOOMMM!!!”)
    else:
        print(n)
        contagem_regressiva(n – 1)

contagem_regressiva(3)
```

Recursão - Exemplo

```
def contagem_regressiva(n):
    if n <= 0:
        print(“BOOOMMM!!!”)
    else:
        print(n)
        contagem_regressiva(n – 1)

contagem_regressiva(3)
```

O Fluxo de execução das chamadas com n = 1, n=2 e n=3 são encerrados sucessivamente e o programa termina

Recursão - Cuidados

- Cuidado para não criar funções recursivas que não tenham condições de parada
- Certifique-se de que estas as condições de parada sejam chamadas em algum momento
- Nos casos em que as condições de parada não são acessadas, a função entra em chamadas infinitas (*loop* infinito)
- Tente chamar a função abaixo :)

```
def recursao():
    recursao()
```

Exercícios

- Escreva um programa que receba 3 números inteiros e os imprima em ordem crescente.
- Escreva um programa que receba as notas de um aluno e verifique se foi aprovado por média em uma disciplina da UFAL. Inclua funcionalidades para tratar as provas de reposição e final.
- Escreva funções que:
  - verifique se um número dado é um quadrado perfeito.
  - que receba 3 números inteiros e verifique se um triangulo pode ser formado com os lados de acordo com os valores dados.
- Escreva um programa que calcule o fatorial de um número dado com função recursiva.