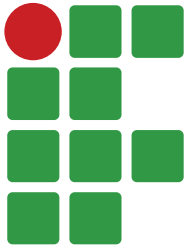
Laços de Repetição
Parte 1: comando **For**

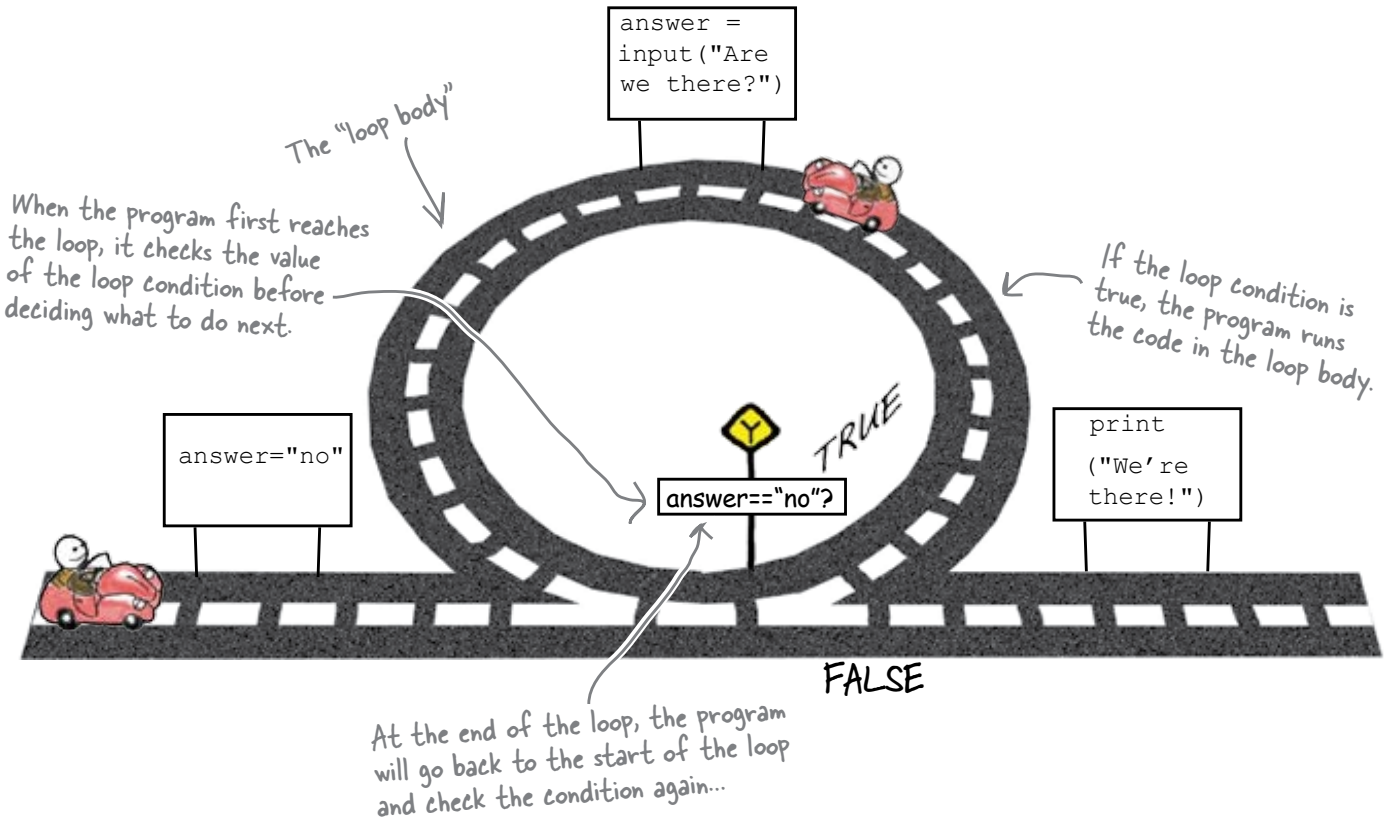
Lógica de Programação - Téc. Informática
Prof. Dr. Paulo César Rodacki Gomes
paulo.gomes@ifc.edu.br



INSTITUTO FEDERAL
Catarinense
Campus Blumenau

Introdução

- Laços de repetição permitem que você execute o mesmo trecho de código repetidas vezes



We want to make sure the loop runs the first time.

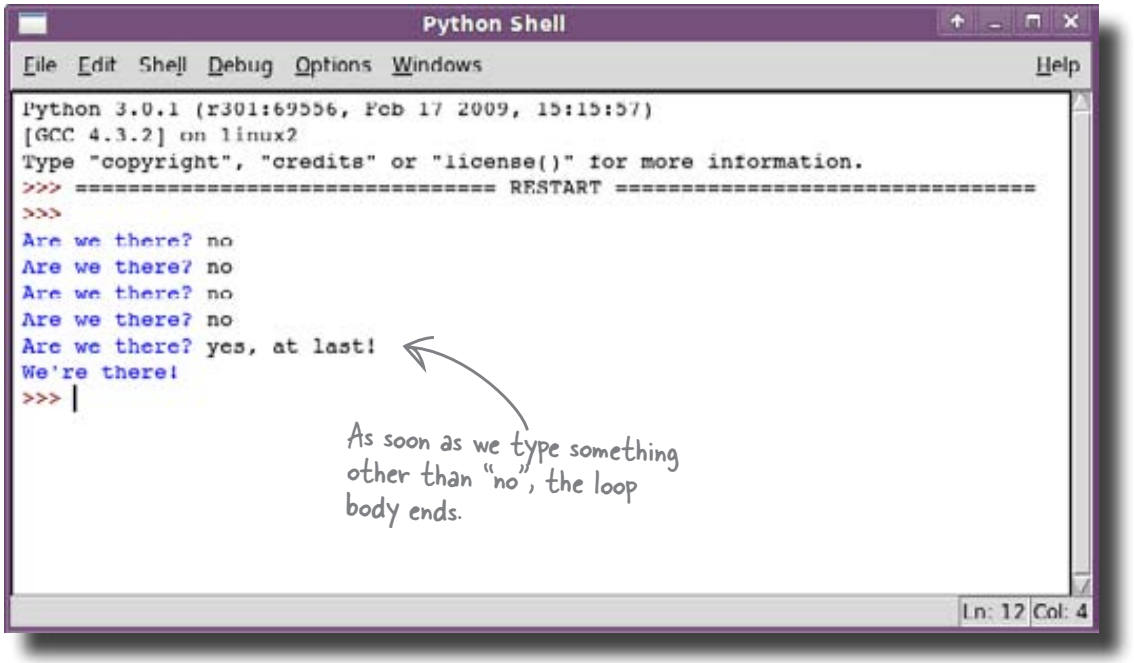
The loop condition

The loop body is the indented code following the "while" line.

```
answer = "no"
while answer == "no":
    answer = input("Are we there? ")
print("We're there!")
```

The loop body is just one line of code in this example, but the loop body can be many lines of code. It might even include branches and other loops.

This is what the loop looks like when you write it as a Python while loop. The code keeps asking the question "Are we there?" until the user types something other than `no`. This is what it looks like when it runs:



As soon as we type something other than "no", the loop body ends.

O laço **for**

O laço for nos permite percorrer os itens de uma coleção e, para cada um deles, executar o bloco de código declarado no loop.

Sua sintaxe é a seguinte:

```
for variavel in lista:
    bloco de comandos
```

No código ao lado percorremos uma lista nomes e imprimimos cada elemento.

```
nomes = ["Pedro", "Joao", "Leticia"]
for n in nomes:
    print(n)
```

```
Pedro
Joao
Leticia
```

A função range()

- a função range() gera uma lista de números inteiros entre um determinado número inicial e um número final
- a lista de números pode ser em ordem crescente ou decrescente
- a lista pode conter números consecutivos (ex.: 1, 2, 3, 4...) ou também pode pular intervalos (ex.: 1, 3, 5, 7,...)

A função range()

- a função aceita um argumento inteiro e retorna um objeto “range" (i.e. a lista de números inteiros)
- Exemplo:

```
print("range function example")
print("Printing range function result")
for i in range(6):
    print(i, end=', ')
```

```
range function example
Printing range function result
0, 1, 2, 3, 4, 5,
```

A função range()

- a função pode receber até 3 argumentos, desses, 2 são opcionais
- sintaxe:

```
range (start, stop[, step])
```

- start: é o primeiro número da lista, por default, inicia com 0, se não for especificado
- stop: é o limite final, a função não inclui este número na lista
- step: é a diferença entre cada número e seu próximo da lista. Por default o valor é 1

Exemplo

- usando somente um argumento em range()
- Exemplo:

```
print("Print first 5 numbers using range function")
for i in range(5):
    print(i, ', ')
```

- Resultado:

```
Print first 5 numbers using range function
0, 1, 2, 3, 4,
```

Exemplo

- usando dois argumentos em range()
- Exemplo:

```
print("Imprime inteiros entre start e stop")
for i in range(5, 10):
    print(i, ', ')
```

- Resultado:

```
Imprime inteiros entre start e stop
5, 6, 7, 8, 9,
```

Exemplo

- usando todos os três argumentos em range()
- Exemplo:

```
print("Usando argumentos start, stop e step")
print("Imprimindo números ímpares entre 1 e 10")
for i in range(1, 10, 2):
    print(i, ', ')
```

- Resultado:

```
Usando argumentos start, stop e step
Imprimindo números ímpares entre 1 e 10
1, 3, 5, 7, 9,
```

Exemplo

- seqüência decrescente com step negativo
- Exemplo:

```
start = -2
stop = -10
step = -2
print("Range decrescente com números negativos")
for number in range(start, stop, step):
    print(number, ', ')
```

- Resultado:

```
Range decrescente com números negativos
-2, -4, -6, -8,
```

Exemplo

- range de número negativo até número positivo
- Exemplo:

```
print ("Imprimindo do negativo para o positivo")
for num in range(-2,5,1):
    print(num, ", ")
```

- Resultado:

```
Imprimindo do negativo para o positivo
-2, -1, 0, 1, 2, 3, 4,
```

Exemplo

- range de número positivo até número negativo
- Exemplo:

```
print ("Imprimindo do positivo para o negativo")
for num in range(2,-5,-1):
    print(num, ", ")
```

- Resultado:

```
Imprimindo do positivo para o negativo
2, 1, 0, -1, -2, -3, -4,
```

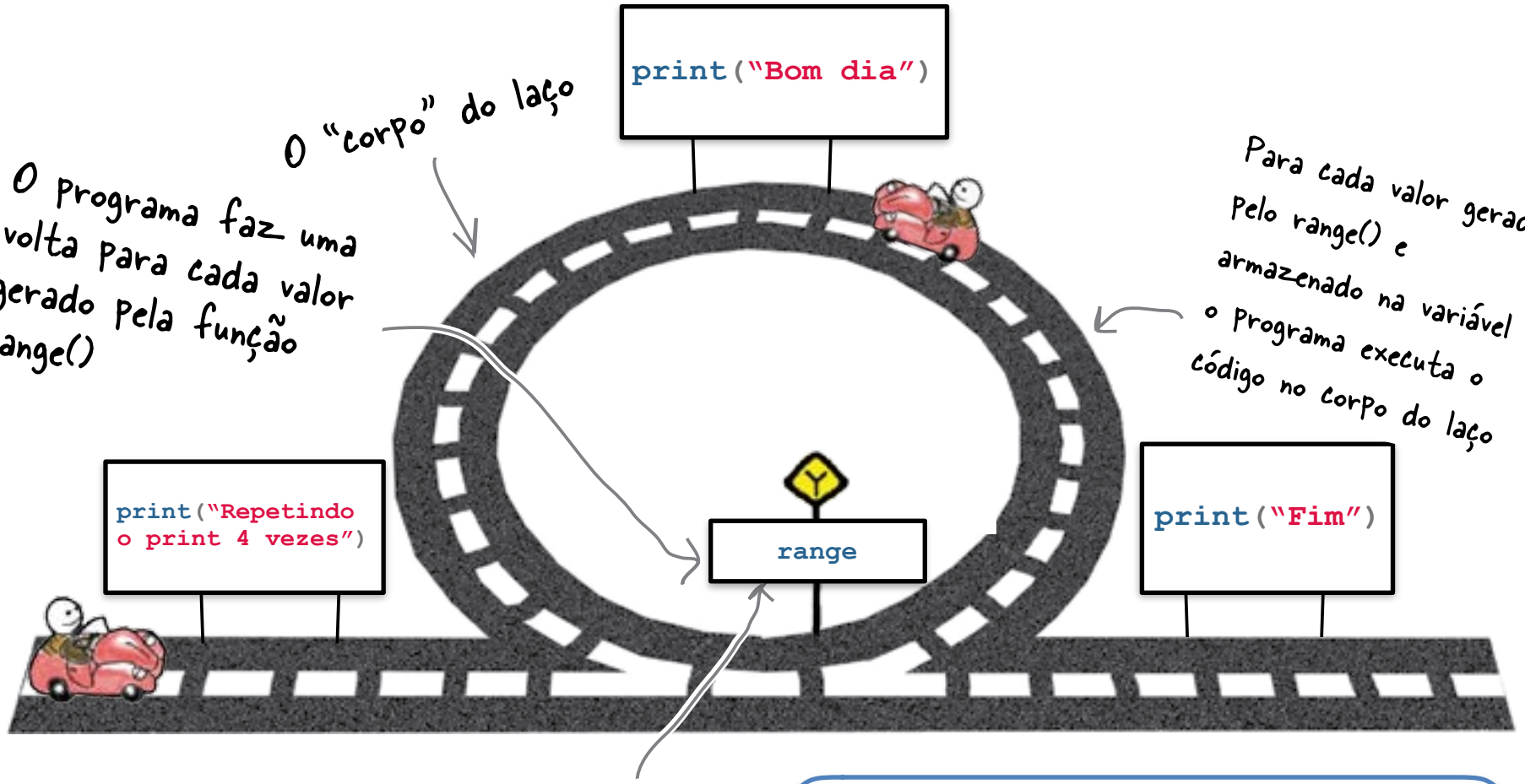
Exemplo

- Comando **for** junto com função **range()**
- Exemplo:

```
print ("Repetindo o print 4 vezes")
for i in range(4):
    print("Bom dia")
print ("Fim")
```

- Resultado:

```
Repetindo o print 4 vezes
Bom dia
Bom dia
Bom dia
Bom dia
Fim
```




```
print ("Repetindo o print 4 vezes")
for i in range(4):
    print("Bom dia")
print ("Fim")
```

Fazendo cálculos iterativos

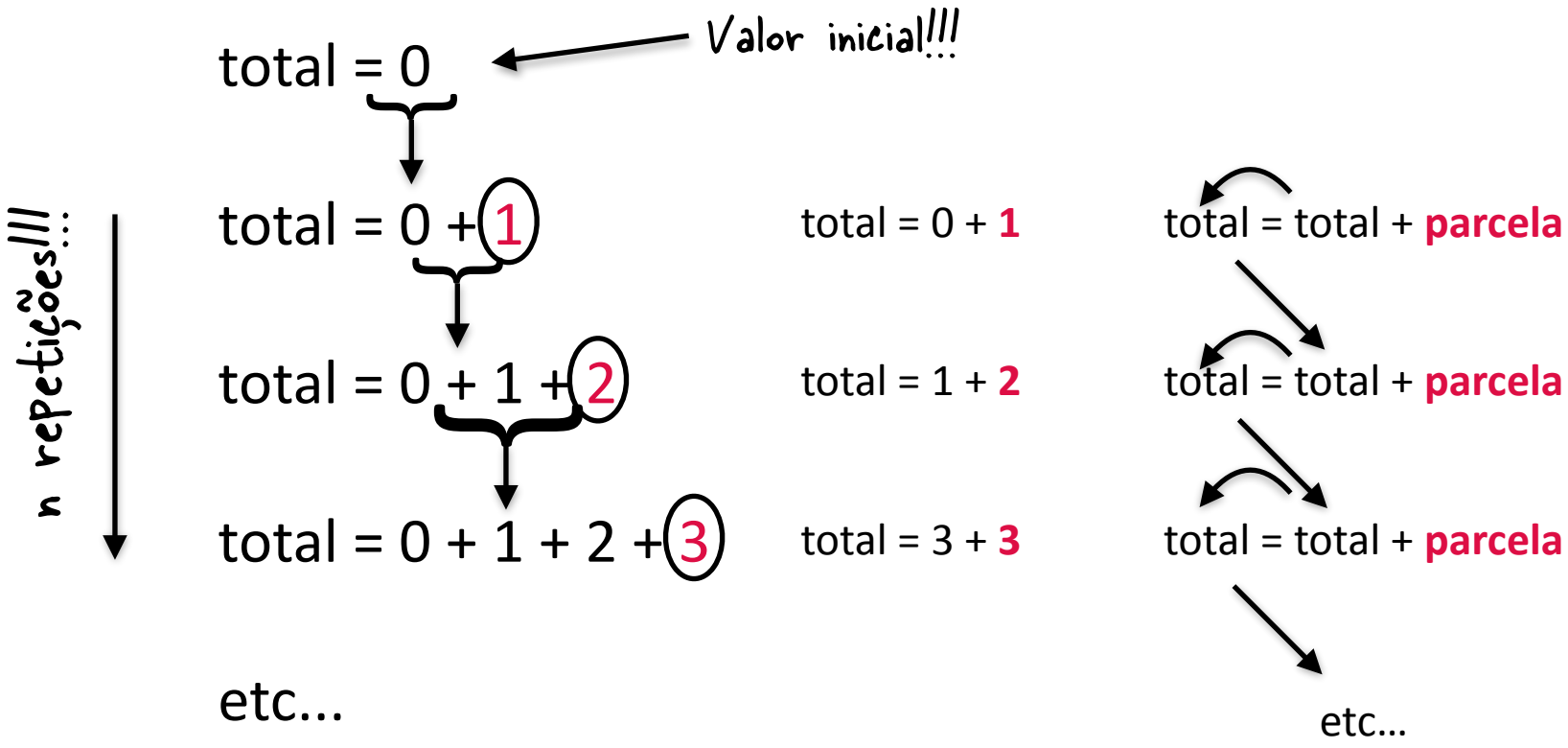
- podemos usar laços de repetição para fazer cálculos os quais, no momento de criação do programa, não sabemos exatamente qual "tamanho" terão
- Exemplo: calcule a soma de todos os números inteiros de 1 a n, sendo o valor n fornecido pelo usuário do programa.

Fazendo cálculos iterativos

- Exemplo: calcule a soma de todos os números inteiros de 1 a n, sendo o valor n fornecido pelo usuário do programa.
- Se o usuário digitar **5**, precisamos calcular **total = 1+2+3+4+5**
- Se o usuário digitar **1000**, precisamos calcular **total = 1+2+3+4+5+6+7+….+1000**
- Ou seja, a quantidade de parcelas da soma depende do valor que o usuário digitar!!!

Fazendo cálculos iterativos

- Podemos usar a expressão: **total = total + parcela**, e montar cada parcela num laço de repetição para fazer o seguinte cálculo:



Fazendo cálculos iterativos

- Exemplo:

```
n = int(input("Digite um número: "))
total = 0
for parcela in range(1, n+1):
    total = total + parcela
print("Soma =", total)
```

- Resultado:

```
Digite um número: 5
Soma = 15
```

Fazendo cálculos iterativos

- Exemplo 2:

```
n = int(input("Digite um número: "))
total = 0
for parcela in range(1, n+1):
    print("total =", total, "+", parcela)
    total = total + parcela
print("Soma =", total)
```

- Resultado:

```
Digite um número: 5
total = 0 + 1
total = 1 + 2
total = 3 + 3
total = 6 + 4
total = 10 + 5
Soma = 15
```