Algoritmos e Programação de Computadores

Comandos Repetitivos: while e for

Prof. Edson Borin
Instituto de Computação (IC/Unicamp)

Agenda

- Comando `while`
- Comando `for`
- Variável acumuladora

Aula anterior

- Comando `continue` and `break`

Comandos Repetitivos*

- Até agora vimos como escrever programas capazes de executar comandos de **forma linear**, e, se necessário, tomar decisões com relação a executar ou não um bloco de comandos.

- Entretanto, eventualmente faz-se necessário executar um bloco de comandos várias vezes para obter o resultado esperado.

* Comandos repetitivos, iterativos, laços, *loops*, ...

Comandos Repetitivos

- Programa que imprime todos os números inteiros de 1 a 4.
- Será que dá pra fazer com o que já sabemos?

Comandos Repetitivos

- Programa que imprime todos os números inteiros de 1 a 4.
- Será que dá pra fazer com o que já sabemos?

```
# Imprime todos os números inteiros de 1 a 4
print(1)
print(2)
print(3)
print(4)
```

Comandos Repetitivos

- Programa que imprime todos os números inteiros de 1 a 100.

```
# Imprime todos os números inteiros de 1 a 100
print(1)
print(2)
print(3)
print(4)
# TODO: Adicionar 95 prints, um para cada número entre 4 e 100
print(100)
```

Comando `while`

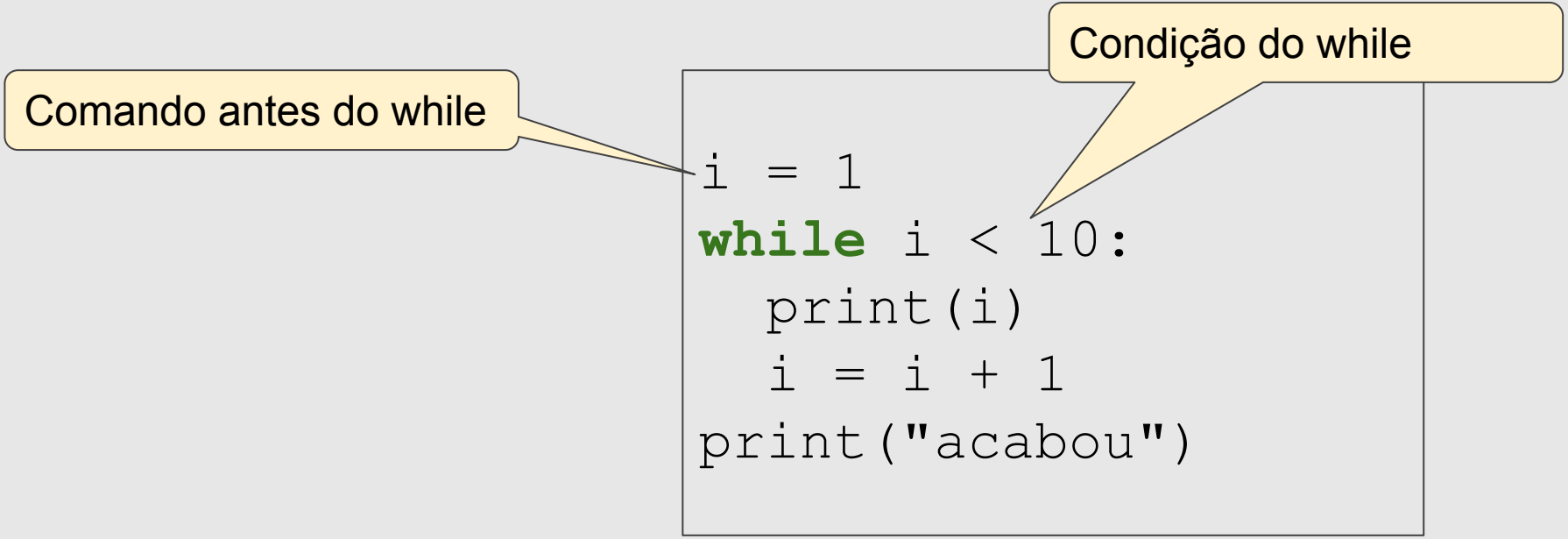
Comando `while`

- Executa um bloco de comando(s) enquanto a condição é verdadeira (`True`).

```
while condicao:
   comando1
   …
   comandoN
```

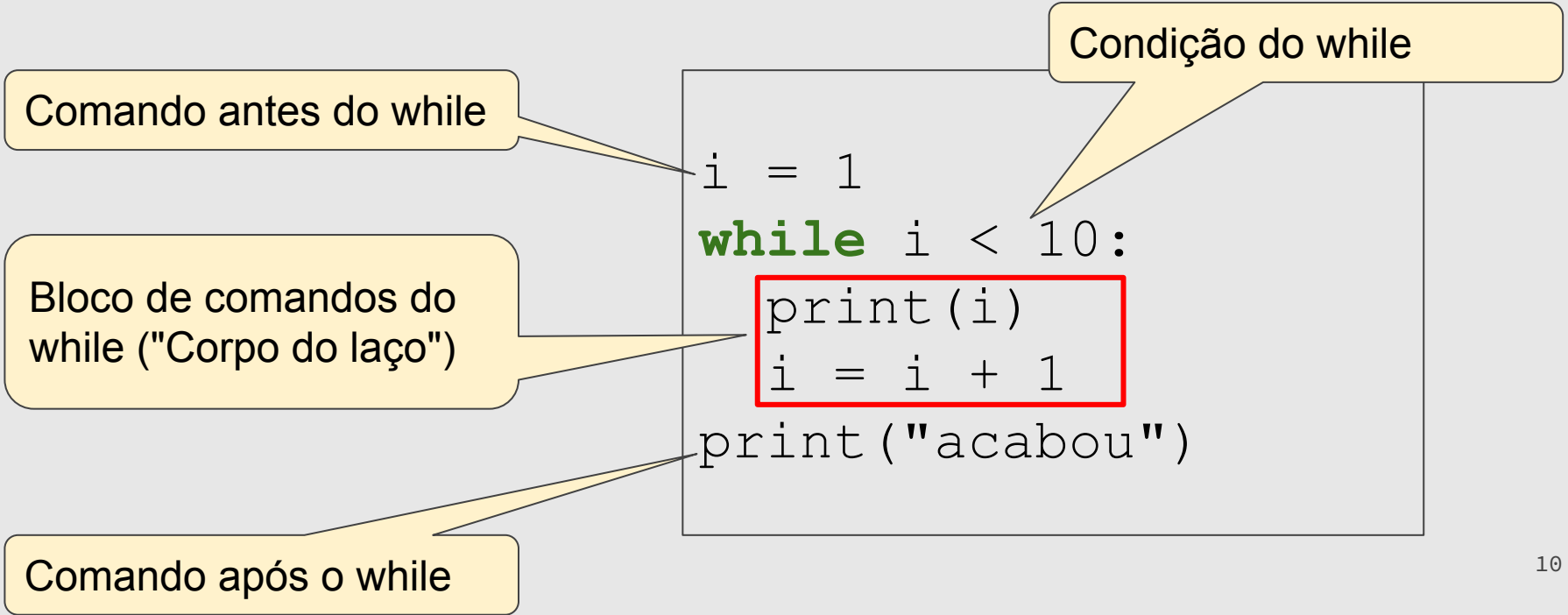
Comando `while`

- Exemplo:



Comando `while`

- Exemplo:



Comando `while`

- Processo de execução do while

Passo 1: teste da condição de parada

```
i = 1
while i < 10:
   print(i)
   i = i + 1
print("acabou")
```

Comando `while`

- Processo de execução do while

Passo 1: teste da condição de parada

Passo 2: Caso a condição for verdadeira, execute os comandos do bloco do while e volte para o Passo 1

```
i = 1
while i < 10:
    print(i)
    i = i + 1
print("acabou")
```

Comando `while`

- Processo de execução do while

Passo 1: teste da condição de parada

Passo 2: Caso a condição for verdadeira, execute os comandos do bloco do while e volte para o Passo 1

Passo 3: Caso a condição for falsa, continue a execução com comandos após o while

```
i = 1
while i < 10:
    print(i)
    i = i + 1
print("acabou")
```

Comando `while`

- Qual será a saída produzida por este programa?

```
i = 1
while i < 10:
  print(i)
  i = i + 1
print("acabou")
```

Comando `while`

- Qual será a saída produzida por este programa?

```
$python3 while-1.py
1
2
3
4
5
6
7
8
9
acabou
```

```
i = 1
while i < 10:
  print(i)
  i = i + 1
print("acabou")
```

Comando `while`

- Programa que imprime todos os números de 1 a 100.

```
# Imprime todos os números de 1 a 100
numero = 1
while numero <= 100:
    print(numero)
    numero = numero + 1
```

Comando `while`

- Programa que imprime os *n* primeiros números inteiros.

```
# Imprime os n primeiros números
n = int(input("Digite um número: "))
numero = 1
while numero <= n:
    print(numero)
    numero = numero + 1
```

Comando `while`

- O que acontece se a condição no comando `while` for falsa na **primeira** vez?

```
while a != a:
    a = a + 1
```

Comando `while`

- O que acontece se a condição no comando `while` for falsa na **primeira** vez? Ele nunca entra na repetição (no laço)

```
while a != a:
    a = a + 1
```

Comando `while`

- O que acontece se a condição no comando `while` for **sempre** verdadeira?

```
while a == a:
    a = a + 1
```

Comando `while`

- O que acontece se a condição no comando `while` for **sempre** verdadeira? Ele entra na repetição e nunca sai (laço infinito).

```
while a == a:
    a = a + 1
```

Listas

Listas (Breve Introdução)

- Uma lista em Python é uma estrutura que armazena vários dados, que podem ser de um mesmo tipo ou não.
- Uma lista é criada como a construção: $[dado_1, dado_2, ..., dado_n]$

```
lista1 = [10, 20, 30, 40]
lista2 = ["programação", "mc102", "python"]
lista3 = ["oi", 2.0, 5, [10, 20]]
```

Listas (Breve Introdução)

- O acesso a um dado específico da lista ocorre por indicação do seu índice.

```
lista3 = ["oi", 2.0, 5, [10, 20]]
print(lista3[1])   # O índice do primeiro elemento é 0.
2.0
print(lista3[2])
5
print(lista3[3])
[10, 20]
print(lista3[4])
IndexError: list index out of range
```

Comando `for`

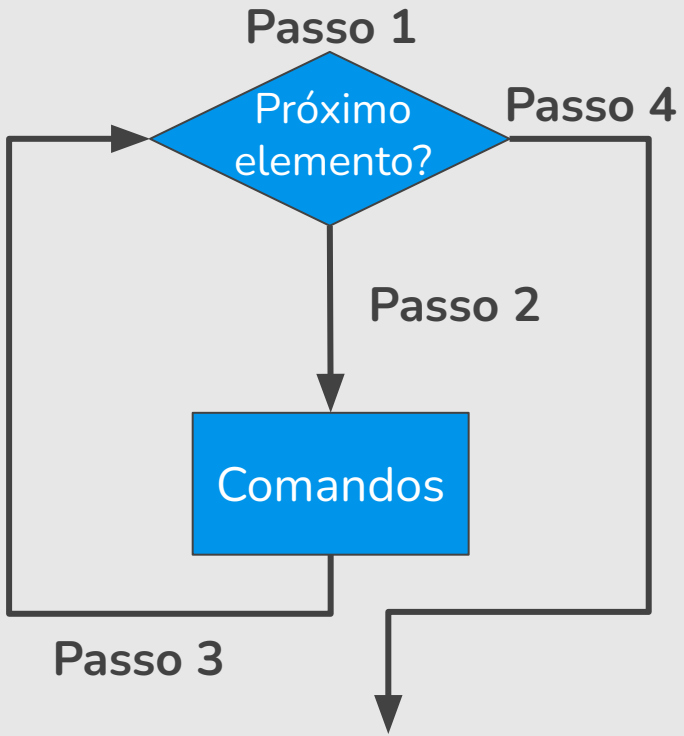
Comando `for`

- É a estrutura de repetição mais usada no Python.
- Para cada elemento da lista, em ordem de ocorrência, é atribuído este elemento à variável e então é executado o(s) comando(s).

```
for variável in lista:
    comando(s)
```

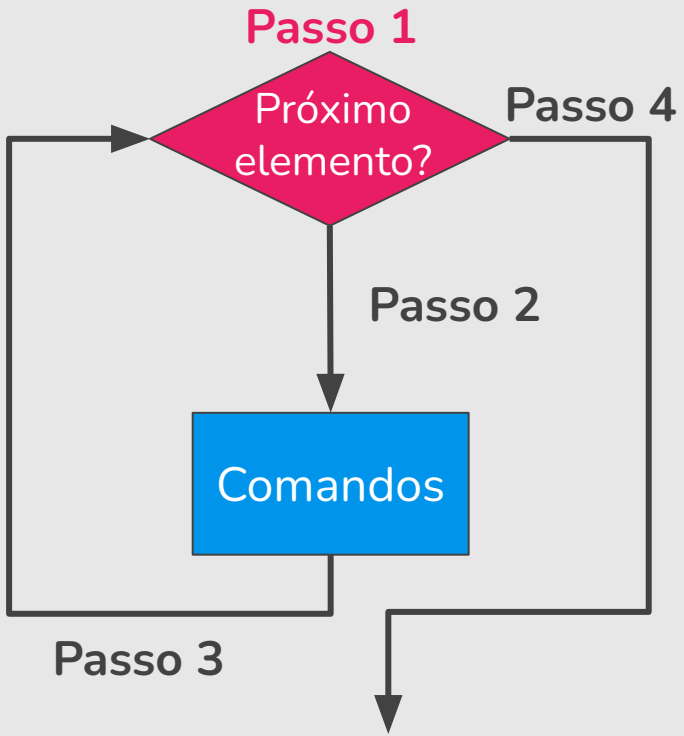
Comando `for`

- **Passo 1**: Verifica se percorreu toda a lista.
  - Se não percorreu, atribui-se o próximo elemento da lista para a variável.
  - Se percorreu, vai para **Passo 4**
- **Passo 2**: Executa comandos
- **Passo 3**: Volta para **Passo 1**



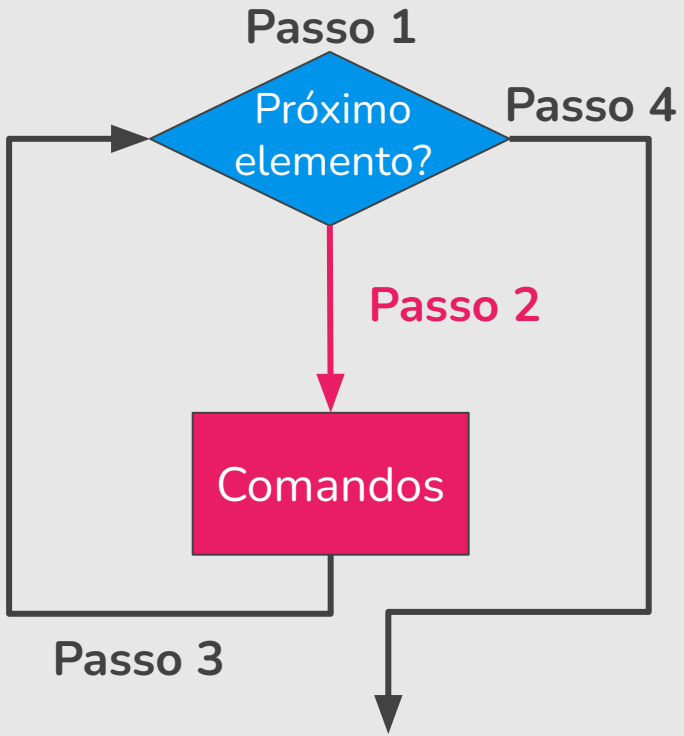
Comando `for`

- **Passo 1**: Verifica se percorreu toda a lista.
  - Se não percorreu, atribui-se o próximo elemento da lista para a variável.
  - Se percorreu, vai para **Passo 4**
- **Passo 2**: Executa comandos
- **Passo 3**: Volta para **Passo 1**



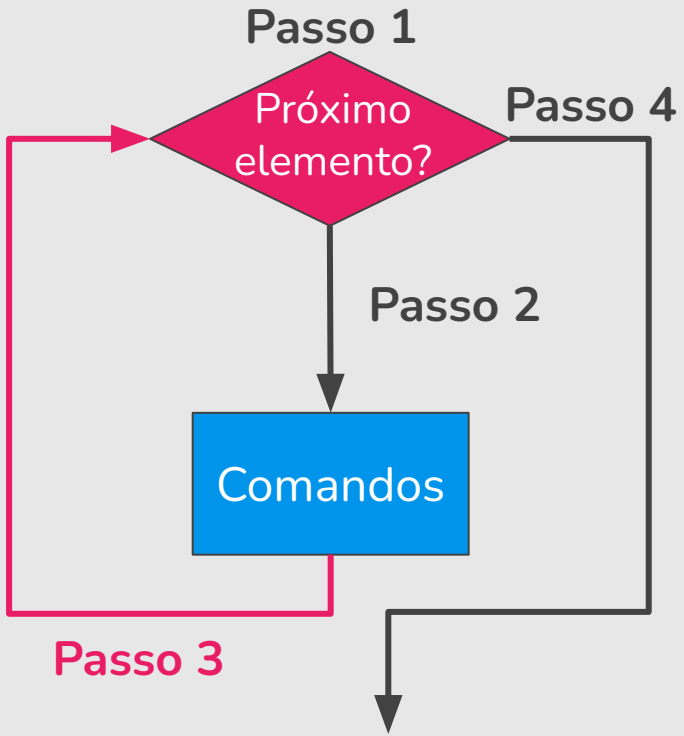
Comando `for`

- **Passo 1**: Verifica se percorreu toda a lista.
  - Se não percorreu, atribui-se o próximo elemento da lista para a variável.
  - Se percorreu, vai para **Passo 4**
- **Passo 2**: Executa comandos
- **Passo 3**: Volta para **Passo 1**



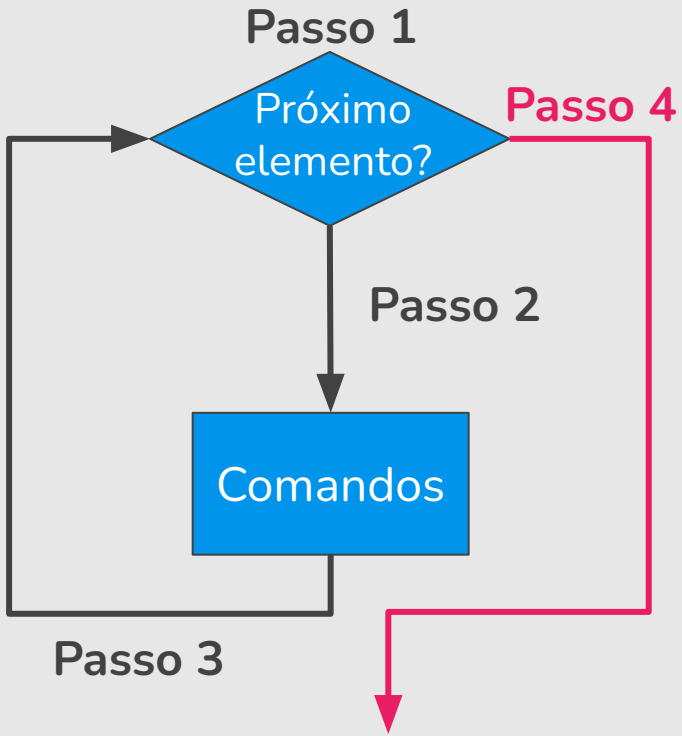
Comando `for`

- **Passo 1**: Verifica se percorreu toda a lista.
  - Se não percorreu, atribui-se o próximo elemento da lista para a variável.
  - Se percorreu, vai para **Passo 4**
- **Passo 2**: Executa comandos
- **Passo 3**: Volta para **Passo 1**



Comando `for`

- **Passo 1**: Verifica se percorreu toda a lista.
  - Se não percorreu, atribui-se o próximo elemento da lista para a variável.
  - Se percorreu, vai para **Passo 4**
- **Passo 2**: Executa comandos
- **Passo 3**: Volta para **Passo 1**



Comando `for`

- Programa que imprime todos os números de uma lista.

```
# Imprime todos os números de uma lista
lista_numeros = [1, 2, 3, 4, 5]
for numero in lista_numeros:
    print(numero)
```

```
1
2
3
4
5
```

Comando `for`

- Programa que imprime todos os números de uma lista.

```
# Imprime todos os números de uma lista

for numero in [1, 2, 3, 4, 5]:
    print(numero)
```

```
1
2
3
4
5
```

Comando `for`

- Programa que imprime todos os números de 1 a 100.

A Função `range`

- É comum fazermos um laço `for` iterar sobre valores numéricos.
- Em Python, a função `range(n)` gera uma lista com valores de `0` até n`-1`.

A Função `range`

- É comum fazermos um laço `for` iterar sobre valores numéricos.
- Em Python, a função `range(n)` gera uma lista com valores de `0` até $n-1$.
- Programa que imprime todos os números de 0 a 9.

```
# Imprime todos os números de 0 a 9
for numero in range(10):
    print(numero)
```

A Função `range`

- Podemos especificar um intervalo de valores na função `range(n)`
  - `range(inicio,fim)`: gera-se números de `inicio` até *`fim`*`-1`.
- Programa que imprime todos os números de 5 a 9.

```
# Imprime todos os números de 5 a 9
for numero in range(5,10):
    print(numero)
```

A Função `range`

- Programa que imprime todos os números de 1 a 100.

```
# Imprime todos os números de 1 a 100
for numero in range(1,101):
    print(numero)
```

A Função `range`

- Podemos especificar um passo a ser considerado no intervalo de valores na função `range(n)`
  - `range(inicio,fim,passo)`: gera-se números de `inicio` com incremento de `passo` até *`fim`*`-1`.

A Função `range`

- Programa que imprime todos os números pares entre 0 e 13.

```
# Imprime todos os números pares entre 0 e 13
for numero in range(0,13,2):
    print(numero)
```

```
0
2
4
6
8
10
12
```

while e for

- Programa que imprime os *n* primeiros números.

```
# Imprime os n primeiros números
n = int(input("Digite um número: "))
numero = 1
while numero <= n:
    print(numero)
    numero = numero + 1
```

```
# Imprime os n primeiros números
n = int(input("Digite um número: "))
for numero in range(1,n+1):
    print(numero)
```

`while` ou `for`?

- Use um laço `for`, se você souber, antes de iniciar o laço, o número máximo de vezes que você precisará executar o corpo do laço.
- Por exemplo, se você estiver percorrendo uma lista de elementos, você sabe que o número máximo de iterações do laço que você pode precisar é “todos os elementos da lista”.

`while` ou `for`?

- Use um laço `while` se você precisa repetir alguma computação até que alguma condição seja atendida, e você não pode calcular antecipadamente quando isso acontecerá.
    - `for` : “número de iterações definido”
    - `while` : “número de iterações indefinido”, não temos certeza de quantas iterações precisamos nem podemos estabelecer um limite superior.

Jogo de Adivinhação

```
import random # módulo random
numero = random.randrange(1, 101) # número entre 1 e 100

meu_palpite = int(input("Adivinhe meu número entre 1 e 100: "))
palpites = 1

while meu_palpite != numero:
    if meu_palpite > numero:
        print(meu_palpite, "está acima.")
    elif meu_palpite < numero:
        print(meu_palpite, "está abaixo.")
    meu_palpite = int(input("tente novamente: "))
    palpites = palpites + 1
print("\nÓtimo, você acertou em", palpites, "tentativas!")
```

Variável Acumuladora

Variável Acumuladora

- Vamos ver alguns exemplos de problemas que são resolvidos utilizando laços.
- Há alguns padrões de solução que são bem conhecidos, e são úteis em diversas situações.
- O primeiro padrão deles é o uso de uma “variável acumuladora”.

Ler um inteiro positivo *n*, em seguida ler *n* números do teclado e apresentar a soma destes.

Soma de Números

- Como *n* não é definido a priori, não podemos criar *n* variáveis e depois somá-las.
- A ideia é criar uma variável acumuladora que a cada iteração de um laço **acumula** a soma de todos os números lidos até então.

```
acumuladora = 0
repita n vezes
    leia um número aux
    acumuladora = acumuladora + aux
```

Soma de Números

- Programa que soma *n* números.

```
# Soma n números
n = int(input("Quantos números deseja somar? "))
acumuladora = 0
for numero in range(n):
    aux = int(input("Digite um número: "))
    acumuladora = acumuladora + aux  # Acumula a soma
print("A soma é:", acumuladora)
```

Jogo de Adivinhação

```
import random # módulo random
numero = random.randrange(1, 101) # número entre 1 e 100

meu_palpite = int(input("Adivinhe meu número entre 1 e 100: "))
palpites = 1

while meu_palpite != numero:
    if meu_palpite > numero:
        print(meu_palpite, "está acima.")
    elif meu_palpite < numero:
        print(meu_palpite, "está abaixo.")
    meu_palpite = int(input("tente novamente: "))
    palpites = palpites + 1
print("\nÓtimo, você acertou em", palpites, "tentativas!")
```

Laços e os comandos `break` e `continue`

Laços e o Comando `break`

- O comando `break` faz com que a execução de um laço seja terminada, passando a execução para o próximo comando depois do final do laço.

```
while condicao:
    comando(s)
    break
comando(s)
```

```
for variável in lista:
    comando(s)
    break
comando(s)
```

Laços e o Comando `break`

- O que será impresso?

```
for numero in range(1,11):
    if (numero >= 5):
        break
    print(numero)
print("Terminou o laço.")
```

Laços e o Comando `break`

- O que será impresso?

```
for numero in range(1,11):
    if (numero >= 5):
        break
    print(numero)
print("Terminou o laço.")
```

```
1
2
3
4
"Terminou o laço."
```

Laços e o Comando `continue`

- O comando `continue` faz com que a execução de um laço seja alterada para o final do laço.

```
numero = 1
while numero <= 10:
    if (numero == 5):
        numero = numero + 1
        continue
    print(numero)
    numero = numero + 1
print("Terminou o laço.")
```

- O que será impresso?

Laços e o Comando `continue`

- O comando `continue` faz com que a execução de um laço seja alterada para o final do laço.

```
numero = 1
while numero <= 10:
    if (numero == 5):
        numero = numero + 1
        continue
    print(numero)
    numero = numero + 1
print("Terminou o laço.")
```

- O que será impresso?

```
1
2
3
4
6
7
8
9
10
"Terminou o laço."
```

Exercícios

1. Faça um programa que lê dois números inteiros positivos *a* e *b*. Utilizando laços, o seu programa deve calcular e imprimir o valor a^b.
2. Faça um programa que lê um número *n* e imprima os valores entre 2 e *n*, que são divisores de *n*.
3. Repita o Jogo de Adivinhação dando a opção do jogador de desistir, por exemplo, escolhendo o número 0.

Exercícios

1. Faça um programa que lê dois números inteiros positivos *a* e *b*. Utilizando laços, o seu programa deve calcular e imprimir o valor a^b.
2. Faça um programa que lê um número *n* e imprima os valores entre 2 e *n*, que são divisores de *n*.
3. Repita o Jogo de Adivinhação dando a opção do jogador de desistir, por exemplo, escolhendo o número 0.

Exercício 1: Usando `while`

Faça um programa que lê dois números inteiros positivos *a* e *b*. Utilizando laços, o seu programa deve calcular e imprimir o valor a^b.

Exercício 1: Usando `while`

Faça um programa que lê dois números inteiros positivos *a* e *b*. Utilizando laços, o seu programa deve calcular e imprimir o valor a^b.

```
base = int(input("Digite a base: ")) # base a
expoente = int(input("Digite o expoente: ")) # expoente b

contador = 0
resultado = 1

while (contador < expoente):
    # base ** expoente  = base * base (expoente vezes)
    resultado = resultado * base
    contador = contador + 1
print(base, "elevado a", expoente, "=", resultado)
```

Exercício 1: Usando `while`

Faça um programa que lê dois números inteiros positivos *a* e *b*. Utilizando laços, o seu programa deve calcular e imprimir o valor a^b.

```
base = int(input("Digite a base: ")) # base a
expoente = int(input("Digite o expoente: ")) # expoente b

contador = 0
resultado = 1

while (contador != expoente):
    # base ** expoente  = base * base (expoente vezes)
    resultado = resultado * base
    contador = contador + 1
print(base, "elevado a", expoente, "=", resultado)
```

Exercício 1: Usando `for`

Faça um programa que lê dois números inteiros positivos *a* e *b*. Utilizando laços, o seu programa deve calcular e imprimir o valor a^b.

Exercício 1: Usando `for`

Faça um programa que lê dois números inteiros positivos *a* e *b*. Utilizando laços, o seu programa deve calcular e imprimir o valor a^b.

```
base = int(input("Digite a base: ")) # base a
expoente = int(input("Digite o expoente: ")) # expoente b

resultado = 1

for numero in range(1,expoente+1):
    # base ** expoente  = base * base (expoente vezes)
    resultado = resultado * base
print(base, "elevado a", expoente, "=", resultado)
```

Exercício 1

Faça um programa que lê dois números inteiros ~~positivos~~ *a* e *b*. Utilizando laços, o seu programa deve calcular e imprimir o valor a^b.

```
base = int(input("Digite a base: ")) # base a
expoente = int(input("Digite o expoente: ")) # expoente b

contador = 0
resultado = 1

if (expoente < 0):
    while contador > expoente:
        # base ** expoente = 1 / (base * base) (expoente vezes)
        resultado = resultado / base
        contador = contador - 1
else:
    while (contador < expoente):
        # base ** expoente  = base * base (expoente vezes)
        resultado = resultado * base
        contador = contador + 1
print(base, "elevado a", expoente, "=", resultado)
```

```
base = int(input("Digite a base: ")) # base a
expoente = int(input("Digite o expoente: ")) # expoente b

contador = 0
resultado = 1

if (expoente < 0):
    while contador > expoente:
        # base ** expoente = 1 / (base * base) (expoente vezes)
        resultado = resultado / base
        contador = contador - 1
    print(base, "elevado a", expoente, "=", format(resultado, ".2f"))
else:
    while (contador < expoente):
        # base ** expoente  = base * base (expoente vezes)
        resultado = resultado * base
        contador = contador + 1
    print(base, "elevado a", expoente, "=", resultado)
```

Exercícios

1. Faça um programa que lê dois números inteiros positivos *a* e *b*. Utilizando laços, o seu programa deve calcular e imprimir o valor a^b.
2. Faça um programa que lê um número *n* e imprima os valores entre 2 e *n*, que são divisores de *n*.
3. Repita o Jogo de Adivinhação dando a opção do jogador de desistir, por exemplo, escolhendo o número 0.

Exercício 2: Usando `for`

Faça um programa que lê um número *n* e imprima os valores entre 2 e *n*, que são divisores de *n*.

```
n = int(input("Digite um número inteiro positivo: "))

for numero in range(2,n+1):
    if (n % numero == 0): # se n é divisível por numero
        print(numero, end=" ")
```

Exercício 2: Usando `while`

Faça um programa que lê um número *n* e imprima os valores entre 2 e *n*, que são divisores de *n*.

```
n = int(input("Digite um número inteiro positivo: "))

numero = 2
while numero <= n:
    if (n % numero == 0): # se n é divisível por numero
        print(numero, end=" ")
    numero = numero + 1
```

Exercícios

1. Faça um programa que lê dois números inteiros positivos *a* e *b*. Utilizando laços, o seu programa deve calcular e imprimir o valor a^b.
2. Faça um programa que lê um número *n* e imprima os valores entre 2 e *n*, que são divisores de *n*.
3. Repita o Jogo de Adivinhação dando a opção do jogador de desistir, por exemplo, escolhendo o número 0.

Exercício 3: Jogo de Adivinhação

```
import random # módulo random
numero = random.randrange(1, 101) # número entre 1 e 100

palpites = 1
meu_palpite = int(input("Adivinhe meu número entre 1 e 100: "))

while meu_palpite != numero:
    palpites = palpites + 1
    if meu_palpite > numero:
        print(meu_palpite, "está acima.")
    elif meu_palpite < numero:
        print(meu_palpite, "está abaixo.")
    meu_palpite = int(input("tente novamente: "))
print("\nÓtimo, você acertou em", palpites, "tentativas!")
```

```
import random # módulo random
numero = random.randrange(1, 101) # número entre 1 e 100

palpites = 1
meu_palpite = int(input("Adivinhe meu número entre 1 e 100: "))

while meu_palpite != numero:
    if meu_palpite == 0:
        print("Ah. Você desistiu do jogo.")
        break
    else:
        palpites = palpites + 1
        if meu_palpite > numero:
            print(meu_palpite, "está acima.")
        elif meu_palpite < numero:
            print(meu_palpite, "está abaixo.")
        meu_palpite = int(input("tente novamente: "))

if meu_palpite != 0:
    print("\nÓtimo, você acertou em", palpites, "tentativas!")
```

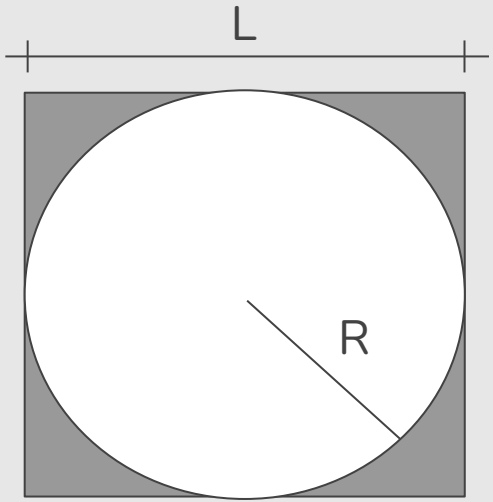
Exercício 3: Jogo de Adivinhação

```
import random # módulo random
numero = random.randrange(1, 101) # número entre 1 e 100

palpites = 1
meu_palpite = int(input("Adivinhe meu número entre 1 e 100: "))

while meu_palpite != numero:
    palpites = palpites + 1
    if meu_palpite > numero:
        print(meu_palpite, "está acima.")
    elif meu_palpite < numero:
        print(meu_palpite, "está abaixo.")
    meu_palpite = int(input("tente novamente: "))
print("\nÓtimo, você acertou em", palpites, "tentativas!")
```

Exercício 4: Estimando PI com Monte Carlo

Área do quadrado = L x L

Área do círculo = PI x R^2



* L = 2 x R

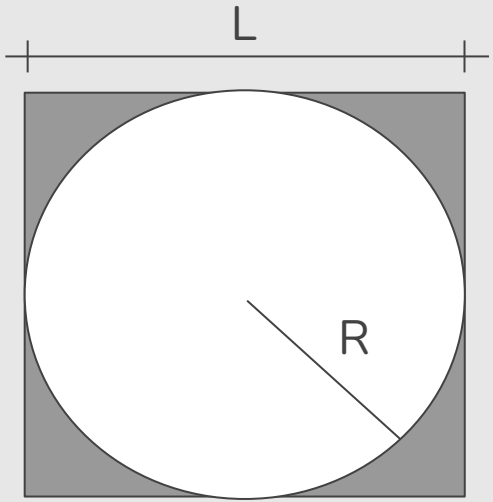
Exercício 4: Estimando PI com Monte Carlo

Área do quadrado = L x L

Área do círculo = PI x R^2

$$\frac{\text{Área do círculo}}{\text{Área do quadrado}} = \frac{\text{Chutes dentro}}{\text{Total de chutes}}$$

Método de Monte Carlo



* L = 2 x R

Exercício 4: Estimando PI com Monte Carlo

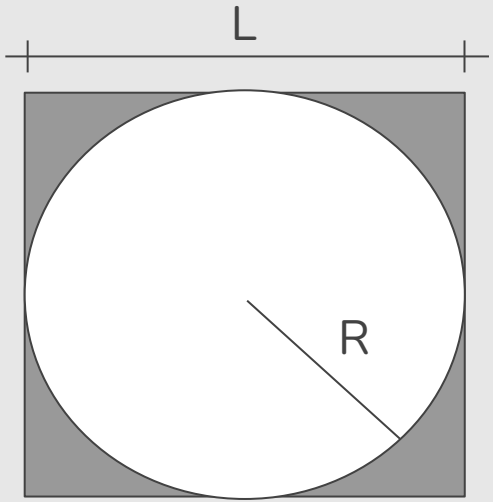
– – –

Área do quadrado = L x L

Área do círculo = PI x R^2

$$\frac{\text{Área do círculo}}{\text{Área do quadrado}} = \frac{\text{Chutes dentro}}{\text{Total de chutes}}$$

$$\frac{PI \times R^2}{L \times L} = \frac{\text{Chutes dentro}}{\text{Total de chutes}}$$



* L = 2 x R

Exercício 4: Estimando PI com Monte Carlo

– – –

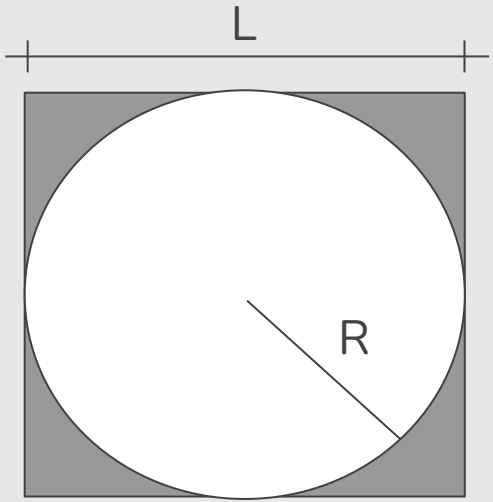
Área do quadrado = L x L

Área do círculo = PI x R^2

$$\frac{\text{Área do círculo}}{\text{Área do quadrado}} = \frac{\text{Chutes dentro}}{\text{Total de chutes}}$$

$$\frac{PI \times R^2}{L \times L} = \frac{\text{Chutes dentro}}{\text{Total de chutes}}$$

$$PI = \frac{\text{Chutes dentro} \times L \times L}{\text{Total de chutes} \times R^2} =$$



* L = 2 x R

Exercício 4: Estimando PI com Monte Carlo

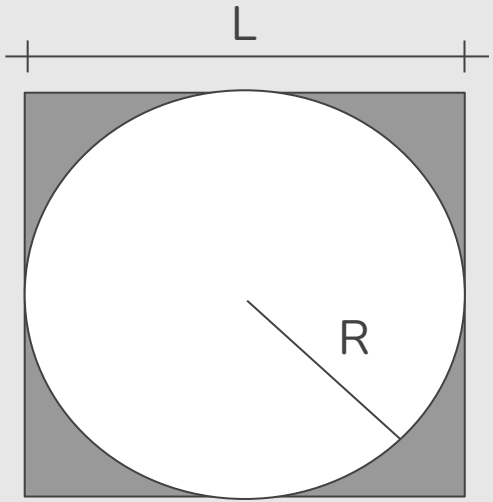
Área do quadrado = L x L

Área do círculo = PI x R^2

$$\frac{\text{Área do círculo}}{\text{Área do quadrado}} = \frac{\text{Chutes dentro}}{\text{Total de chutes}}$$

$$\frac{PI \times R^2}{L \times L} = \frac{\text{Chutes dentro}}{\text{Total de chutes}}$$

$$PI = \frac{\text{Chutes dentro} \times L \times L}{\text{Total de chutes} \times R^2} = \frac{\text{Chutes dentro} \times 4 \times R^2}{\text{Total de chutes} \times R^2}$$



* L = 2 x R

Exercício 4: Estimando PI com Monte Carlo

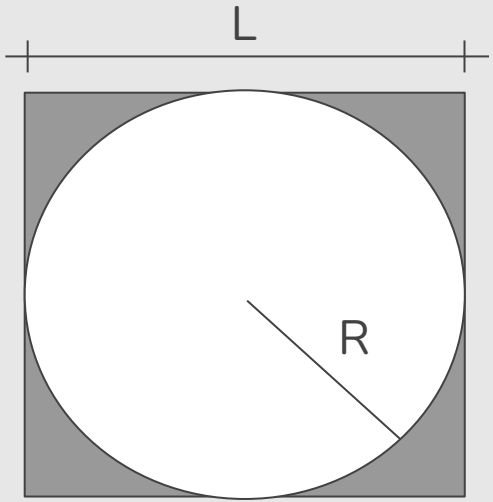
– – –

Área do quadrado = L x L

Área do círculo = PI x R^2

$$\frac{\text{Área do círculo}}{\text{Área do quadrado}} = \frac{\text{Chutes dentro}}{\text{Total de chutes}}$$

$$\frac{PI \times R^2}{L \times L} = \frac{\text{Chutes dentro}}{\text{Total de chutes}}$$



* L = 2 x R

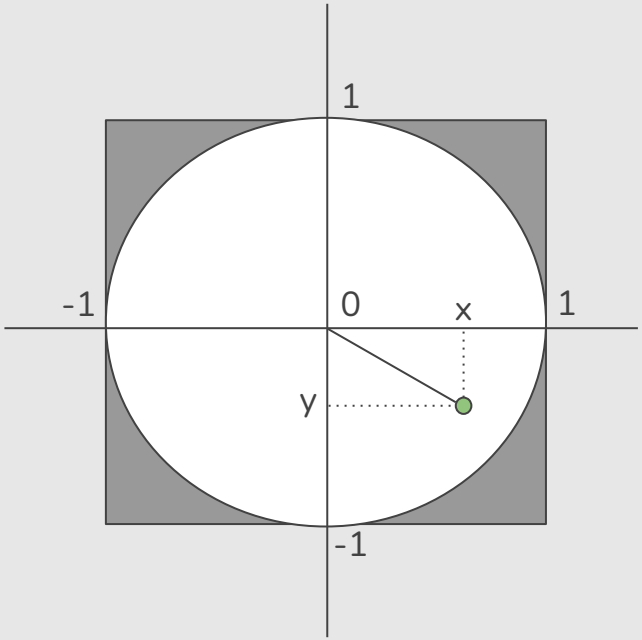
$$PI = \frac{\text{Chutes dentro} \times L \times L}{\text{Total de chutes} \times R^2} = \frac{\text{Chutes dentro} \times 4 \times R^2}{\text{Total de chutes} \times R^2} = \frac{\text{Chutes dentro} \times 4}{\text{Total de chutes}}$$

Exercício 4: Estimando PI com Monte Carlo

Plano cartesiano

Chute: ponto com coordenada (x,y)

-1 <= x <= 1
-1 <= y <= 1



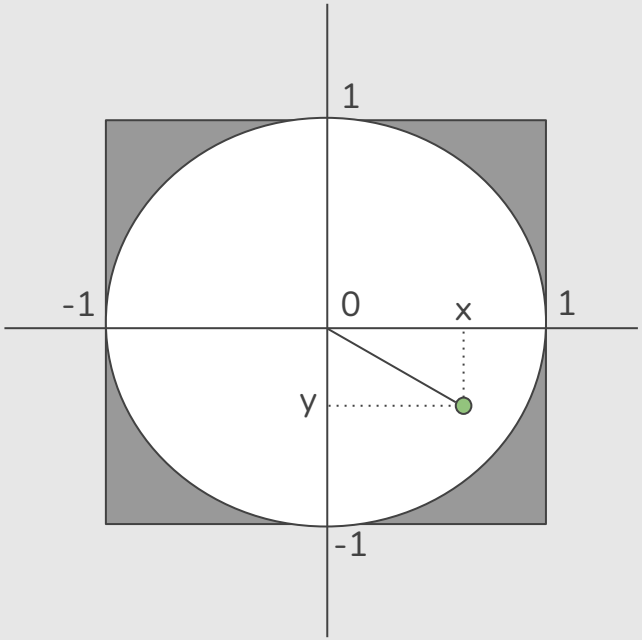
Exercício 4: Estimando PI com Monte Carlo

Plano cartesiano

Chute: ponto com coordenada (x,y)

-1 <= x <= 1
-1 <= y <= 1

$x^2 + y^2 <= 1$ ⇔ Chute dentro



Exercício 4: Estimando PI com Monte Carlo

```
import random # módulo random

N  = 10000
dentro = 0

for i in range(N) :
    x = random.uniform(-1,1) # número real (float) entre -1 e 1
    y = random.uniform(-1,1) # número real (float) entre -1 e 1
    if (x*x+y*y) <= 1 :
        dentro = dentro + 1

print("Pi = ", (4*dentro)/N)
```

Mais Exercícios =)

- https://wiki.python.org.br/EstruturaDeRepeticao: 51 exercícios \o/

- Curso de Python:
  - https://www.codecademy.com/learn/learn-python

Créditos

Os *slides* deste curso foram baseados nos slides produzidos e cedidos gentilmente pela Professora Sandra Ávila, do Instituto de Computação da Unicamp.