Algoritmos e Programação de Computadores

Comandos para execução repetitiva: While e For

Ref.: material original (1o S., T. KLMN). por **Profa. Sandra Avila**, Instituto de Computação (IC/ Unicamp)

MC102-Z, 21 Agosto, 2018

Agenda

- Comando `while`
- Comando `for`
- Variável acumuladora
- Comando `continue` and `break`

Comandos Repetitivos*

- Até agora vimos como escrever programas capazes de executar comandos de **forma linear (i.e. sequencial)**, 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, de forma repetitiva (ou ***iterativa***), para obter o(s) resultado(s) esperado(s).

* Comandos para **execução** repetitiva/iterativa, 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?

```
# 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)
# repete 95 vezes a linha acima
print(100)
```

Comando `while`

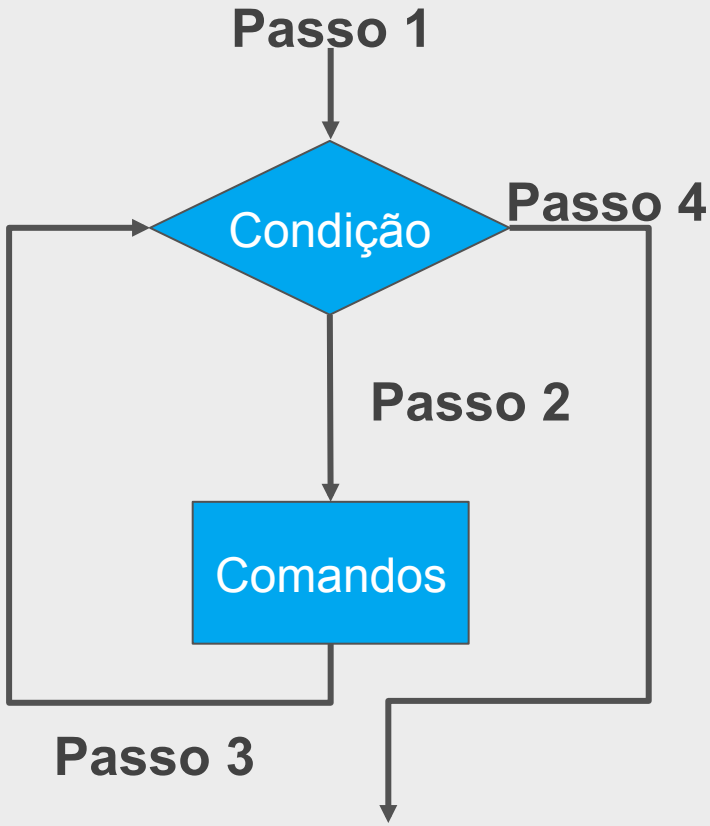
Comando `while`

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

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

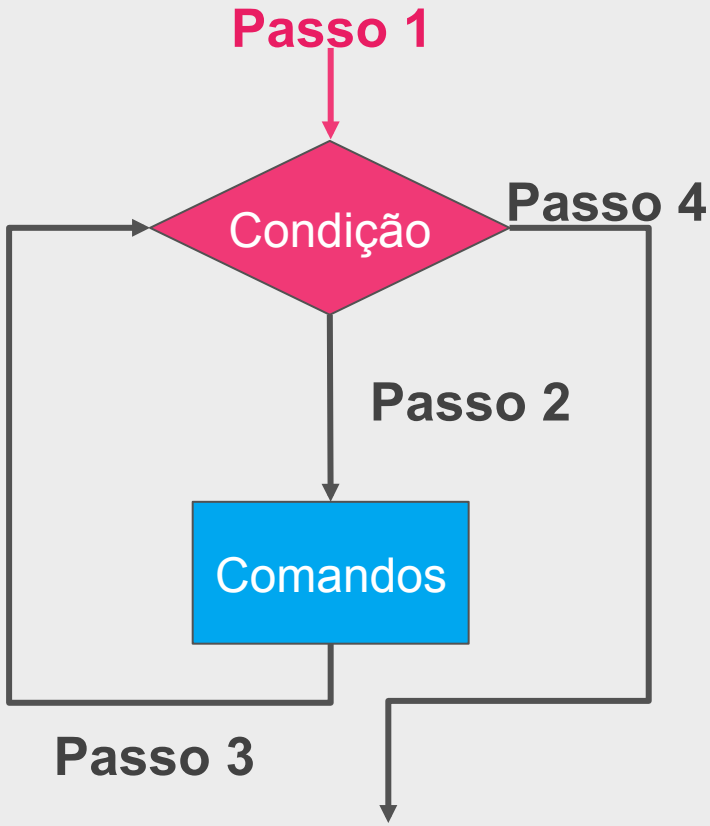
Comando `while`

- **Passo 1**: Testa condição.
  - Se condição for verdadeira, vai para o **Passo 2**
  - Senão, vai para **Passo 4**
- **Passo 2**: Executa comandos
- **Passo 3**: Volta para **Passo 1**



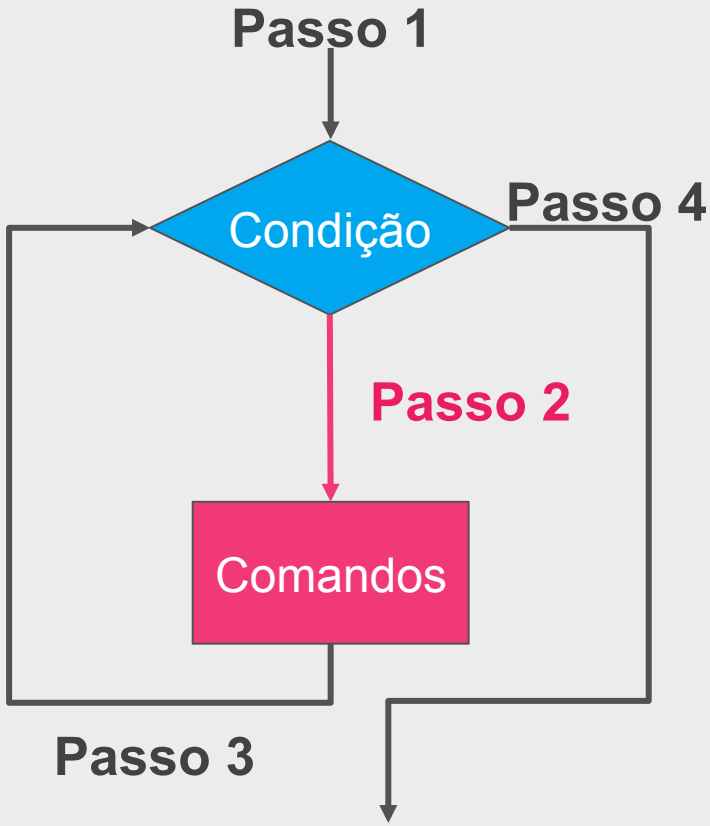
Comando `while`

- **Passo 1**: Testa condição.
  - Se condição for verdadeira, vai para o **Passo 2**
  - Senão, vai para **Passo 4**
- **Passo 2**: Executa comandos
- **Passo 3**: Volta para **Passo 1**



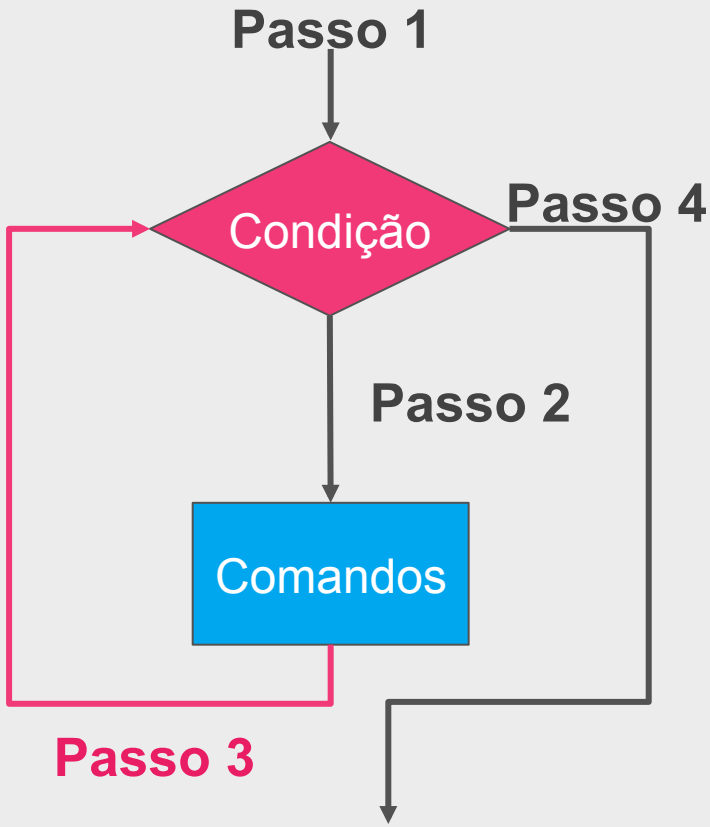
Comando `while`

- **Passo 1**: Testa condição.
  - Se condição for verdadeira, vai para o **Passo 2**
  - Senão, vai para **Passo 4**
- **Passo 2**: Executa comandos
- **Passo 3**: Volta para **Passo 1**



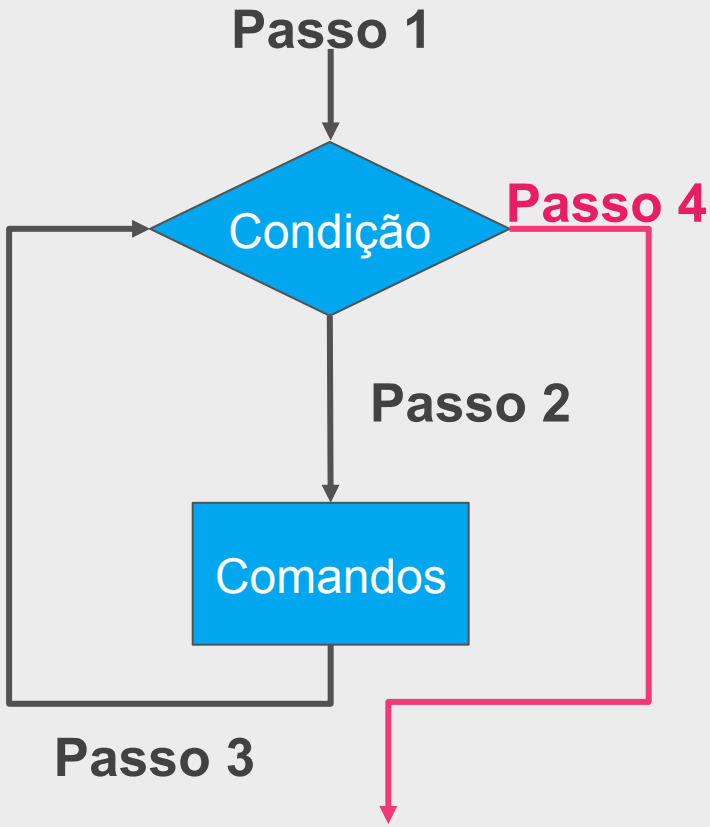
Comando `while`

- **Passo 1**: Testa condição.
  - Se condição for verdadeira, vai para o **Passo 2**
  - Senão, vai para **Passo 4**
- **Passo 2**: Executa comandos
- **Passo 3**: Volta para **Passo 1**



Comando `while`

- **Passo 1**: Testa condição.
  - Se condição for verdadeira, vai para o **Passo 2**
  - Senão, vai para **Passo 4**
- **Passo 2**: Executa comandos
- **Passo 3**: Volta para **Passo 1**



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.

```
# 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? Ele nunca entrar na repetição (no laço/*loop*)

```
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/*loop* infinito).

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

Comando `while-else`

- Ao final do `while` podemos utilizar a instrução `else`.

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

Comando `while-else`

- 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
else:
        print("Fim.")
```

Atenção: Nem sempre faz sentido ter o `else`.

Comando `while-else`

- Programa que imprime os *n* primeiros números.

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

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

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`

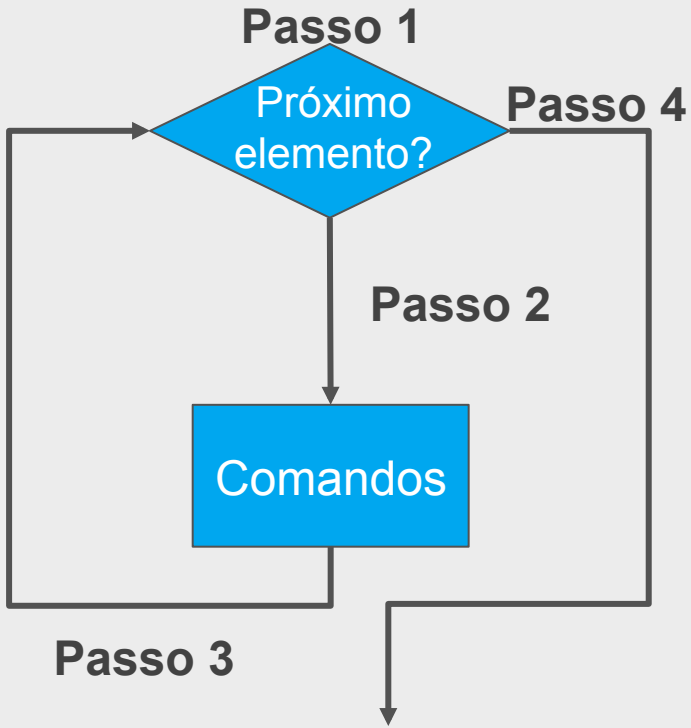
Comando `for`

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

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

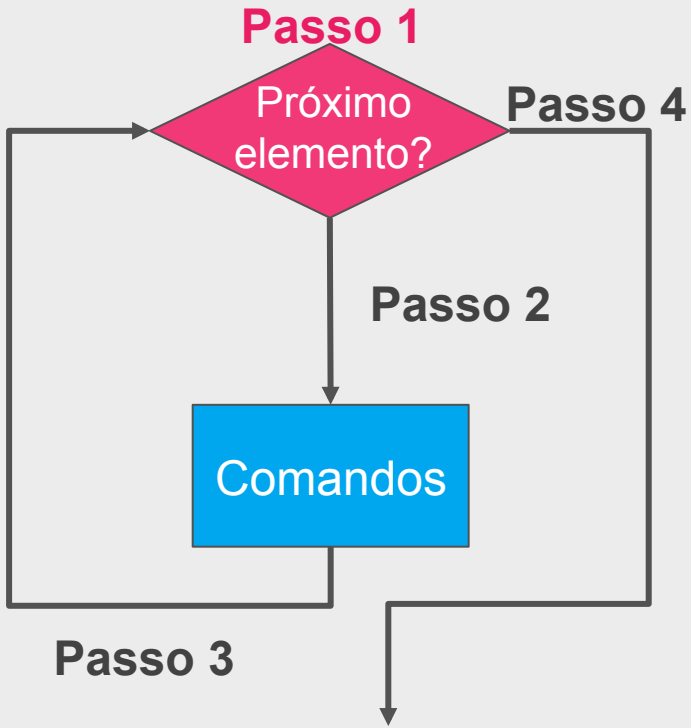
Comando `for`

- **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**



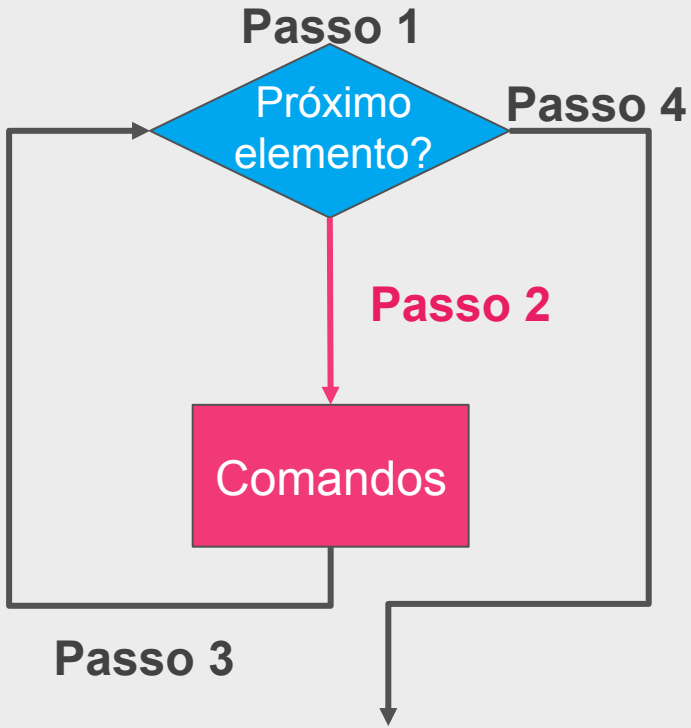
Comando `for`

- **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**



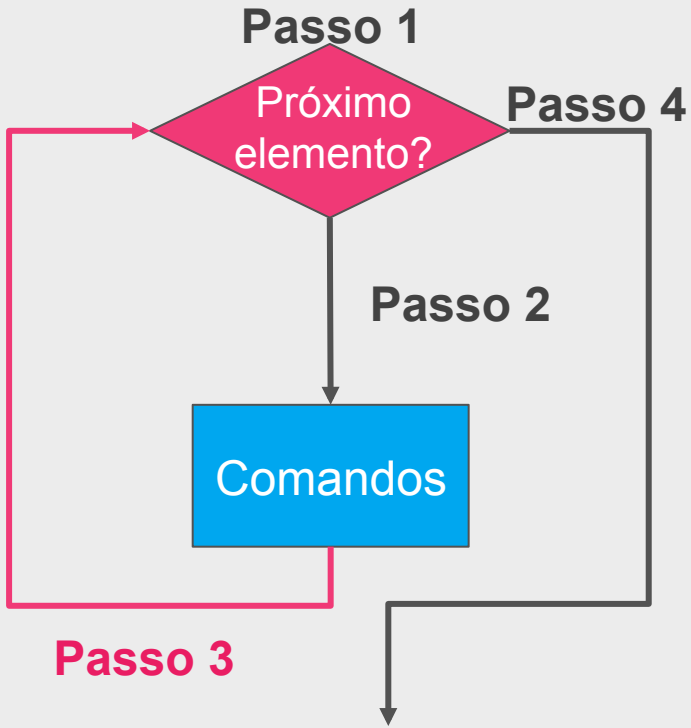
Comando `for`

- **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**



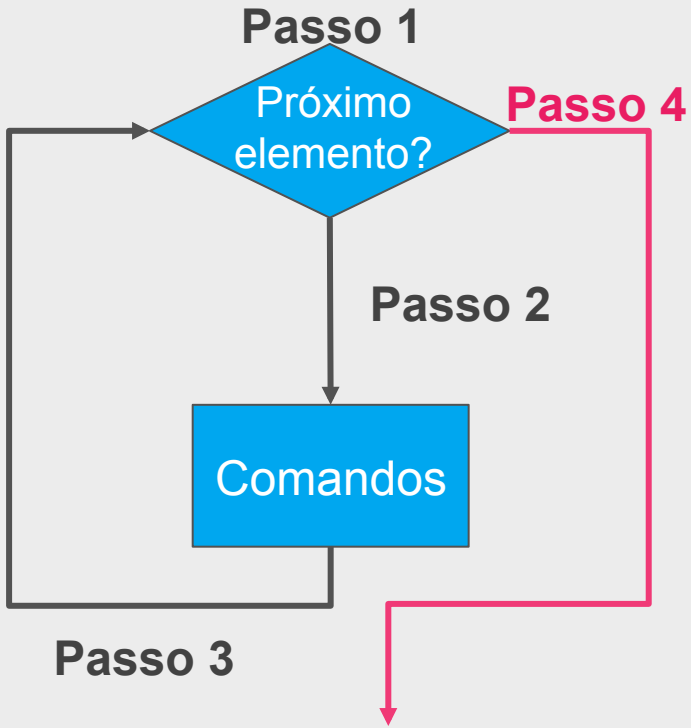
Comando `for`

- **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 1 a 100.

A Função `range`

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

A Função `range`

- O laço/loop **for** permite iterar sobre valores numéricos, a variável de controle é conhecida como **índice** (*index*). Ex. `numero`
- 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 loop/laço `while` se você precisa repetir alguma computação até que alguma condição seja atendida, e você não pode calcular antecipadamente quando isso acontecerá.
  - `for` : “iteração definida”
  - `while` : “iteração indefinida”, 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

palpites = 0
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!")
```

Variável Acumuladora

Variável Acumuladora

- Vamos ver alguns exemplos de problemas que são resolvidos utilizando laços (*loops*).
- 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("Digite o valor de n: "))
acumuladora = 0
for numero in range(n):
    aux = int(input())
    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

palpites = 0
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!")
```

Laços (Loops) e os comandos `break` e `continue`

Laços (Loop) 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(s1)
    break
comando(s2)
```

```
for variável in lista:
    comando(s1)
    break
comando(s2)
```

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 (loop).")
```

```
1
2
3
4
"Terminou o laço (loop)."
```

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.")
```

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

- O que será impresso?

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.

Referências

- Os slides dessa aula foram baseados no material de MC102 da Prof. Sandra Avila (IC/Unicamp)

- Comandos para execução repetitiva (iterativa)
  - https://panda.ime.usp.br/pensepy/static/pensepy/07-Iteracao/maisiteracao.html#o-comando-while
  - https://runestone.academy/runestone/static/thinkcspy/MoreAboutIteration/toctree.html
  - https://github.com/iviarcio/mc102/blob/master/05.Controle%20de%20Fluxo%20-%20Itera%C3%A7%C3%B5es.ipynb