ESTRUTURA DE REPETIÇÃO

AULA 09

PROFESSOR: EDUARDO KAZENSKI

CONTEÚDO

- Comandos de REPETIÇÃO;
  - Faça enquanto
  - Enquanto Faça
  - Para... Faça

OBJETIVOS

- Conseguir aplicar as estruturas corretas conforme o algoritmo exigir em sua lógica.
- Entender como as repetições são representadas nos algoritmos.

ESTRUTURAS DE REPETIÇÃO OU LOOP

- Uma estrutura de repetição, como o próprio nome indica, permite a execução de uma sequência de instruções repetidas vezes até que determinada condição seja avaliada como verdadeira, por um número específico de vezes ou para cada item (elemento) em uma coleção, ou seja, utilizando acumuladores e contadores.

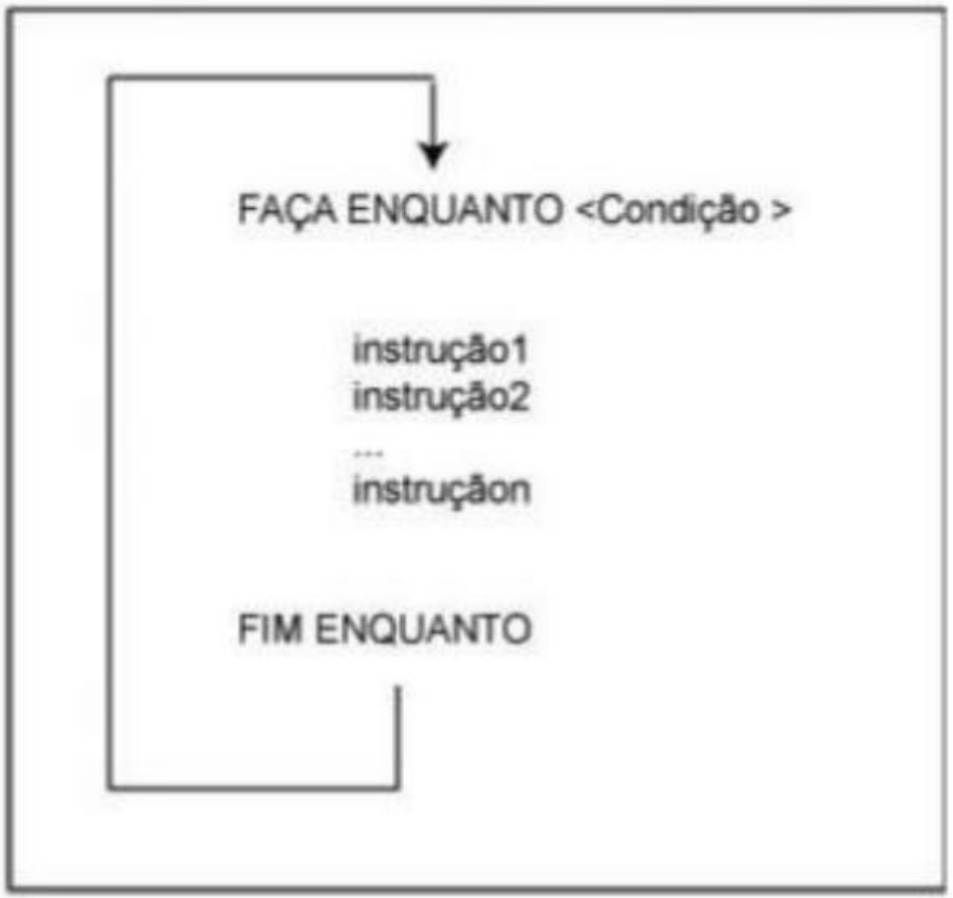
- Tais estruturas também são conhecidas como de loop ou de laço.

ESTRUTURAS DE REPETIÇÃO OU LOOP

- A maior parte das linguagens de programação costuma definir três tipos de estruturas de repetição:
  - **Faça enquanto**
  - **Enquanto Faça**
  - **Para ... Passo ... Faça**

FAÇA ENQUANTO ...

- Nesta estrutura, a rotina só será executada caso a condição estipulada seja avaliada como verdadeira.
- Após entrar na estrutura, o sistema executará todas as instruções contidas na sequência existente dentro do **Faça Enquanto e Fim Enquanto.**
- Sua sintaxe é:



FAÇA ENQUANTO ...

- Nesta estrutura, a condição deverá ser avaliada em seu início. Ao localizar a instrução Fim Enquanto a rotina retornará ao início da estrutura para avaliar novamente a condição. Se avaliada novamente como verdadeira, continuará a repetição.

- Esse tipo de estrutura é recomendado quando o número de repetições das instruções é desconhecido, pois, como já verificamos, a condição será avaliada sempre no início, permanecendo em um loop até que seja avaliada como falsa.

ENQUANTO ... FAÇA (VALIDAÇÃO NO INÍCIO)

```
1.  valor = 1
2.  while valor <= 5:
3.    valor = valor + 1
4.    print(valor)
5.  print(valor)
```

TESTE DE MESA

valor = 1

print (linha 4) =

print (linha 5) =

ENQUANTO ... FAÇA (VALIDAÇÃO NO INÍCIO)

```
1. valor = 1
2. while valor <= 5:
3.     valor = valor + 1
4.     print(valor)
5. print(valor)
```

1<=5?

TESTE DE MESA

valor = 1

print (linha 4) =

print (linha 5) =

ENQUANTO ... FAÇA (VALIDAÇÃO NO INÍCIO)

1. valor = 1
2. while valor <= 5:
3. valor = valor + 1
4. print(valor)
5. print(valor)

1<=5? TRUE

TESTE DE MESA

valor = 1

print (linha 4) =

print (linha 5) =

ENQUANTO ... FAÇA (VALIDAÇÃO NO INÍCIO)

```
1. valor = 1
2. while valor <= 5:
3.     valor = valor + 1
4.     print(valor)
5. print(valor)
```

1 + 1 = 2

TESTE DE MESA

valor = 2

print (linha 4) =

print (linha 5) =

ENQUANTO ... FAÇA (VALIDAÇÃO NO INÍCIO)

```
1. valor = 1
2. while valor <= 5:
3.     valor = valor + 1
4.     print(valor)
5. print(valor)
```

Repete LOOP ↑

TESTE DE MESA

valor = 2

print (linha 4) = 2

print (linha 5) =

ENQUANTO ... FAÇA (VALIDAÇÃO NO INÍCIO)

```
1. valor = 1
2. while valor <= 5:
3.     valor = valor + 1
4.     print(valor)
5. print(valor)
```

Reiniciando

TESTE DE MESA

valor = 2

print (linha 4) = 2

print (linha 5) =

ENQUANTO ... FAÇA (VALIDAÇÃO NO INÍCIO)

```
1. valor = 1
2. while valor <= 5:
3.     valor = valor + 1
4.     print(valor)
5. print(valor)
```

2<=5?

TESTE DE MESA

valor = 2

print (linha 4) = 2

print (linha 5) =

ENQUANTO ... FAÇA (VALIDAÇÃO NO INÍCIO)

```
1. valor = 1
2. while valor <= 5:
3.     valor = valor + 1
4.     print(valor)
5. print(valor)
```

2<=5? TRUE

TESTE DE MESA

valor = 2

print (linha 4) = 2

print (linha 5) =

ENQUANTO ... FAÇA (VALIDAÇÃO NO INÍCIO)

```
1. valor = 1
2. while valor <= 5:
3.     valor = valor + 1
4.     print(valor)
5. print(valor)
```

2 + 1 = 3

TESTE DE MESA

valor = 3

print (linha 4) = 2

print (linha 5) =

ENQUANTO ... FAÇA (VALIDAÇÃO NO INÍCIO)

```
1. valor = 1
2. while valor <= 5:
3.     valor = valor + 1
4.     print(valor)
5. print(valor)
```

Repete LOOP ↑

TESTE DE MESA

valor = 3

print (linha 4) = 3

print (linha 5) =

ENQUANTO ... FAÇA (VALIDAÇÃO NO INÍCIO)

```
1. valor = 1
2. while valor <= 5:
3.     valor = valor + 1
4.     print(valor)
5. print(valor)
```

Reiniciando

TESTE DE MESA

valor = 3

print (linha 4) = 3

print (linha 5) =

ENQUANTO ... FAÇA (VALIDAÇÃO NO INÍCIO)

```
1. valor = 1
2. while valor <= 5:
3.     valor = valor + 1
4.     print(valor)
5. print(valor)
```

3<=5?

TESTE DE MESA

valor = 3

print (linha 4) = 3

print (linha 5) =

ENQUANTO ... FAÇA (VALIDAÇÃO NO INÍCIO)

```
1. valor = 1
2. while valor <= 5:
3.     valor = valor + 1
4.     print(valor)
5. print(valor)
```

3<=5? TRUE

TESTE DE MESA

valor = 3

print (linha 4) = 3

print (linha 5) =

ENQUANTO ... FAÇA (VALIDAÇÃO NO INÍCIO)

```
1. valor = 1
2. while valor <= 5:
3.     valor = valor + 1
4.     print(valor)
5. print(valor)
```

3 + 1 = 4

TESTE DE MESA

valor = 4

print (linha 4) = 3

print (linha 5) =

ENQUANTO ... FAÇA (VALIDAÇÃO NO INÍCIO)

```
1. valor = 1
2. while valor <= 5:
3.     valor = valor + 1
4.     print(valor)
5. print(valor)
```

Repete LOOP ↑

TESTE DE MESA

valor = 4

print (linha 4) = 4

print (linha 5) =

ENQUANTO ... FAÇA (VALIDAÇÃO NO INÍCIO)

```
1. valor = 1
2. while valor <= 5:
3.     valor = valor + 1
4.     print(valor)
5. print(valor)
```

Reiniciando

TESTE DE MESA

valor = 4

print (linha 4) = 4

print (linha 5) =

ENQUANTO ... FAÇA (VALIDAÇÃO NO INÍCIO)

```
1. valor = 1
2. while valor <= 5:
3.     valor = valor + 1
4.     print(valor)
5. print(valor)
```

4<=5?

TESTE DE MESA

valor = 4

print (linha 4) = 4

print (linha 5) =

ENQUANTO ... FAÇA (VALIDAÇÃO NO INÍCIO)

```
1. valor = 1
2. while valor <= 5:
3.     valor = valor + 1
4.     print(valor)
5. print(valor)
```

4<=5? TRUE

TESTE DE MESA

valor = 4

print (linha 4) = 4

print (linha 5) =

ENQUANTO ... FAÇA (VALIDAÇÃO NO INÍCIO)

```
1. valor = 1
2. while valor <= 5:
3.     valor = valor + 1
4.     print(valor)
5. print(valor)
```

4 + 1 = 5

TESTE DE MESA

valor = 5

print (linha 4) = 4

print (linha 5) =

ENQUANTO ... FAÇA (VALIDAÇÃO NO INÍCIO)

```
1. valor = 1
2. while valor <= 5:
3.     valor = valor + 1
4.     print(valor)
5. print(valor)
```

Repete LOOP ↑

TESTE DE MESA

valor = 5

print (linha 4) = 5

print (linha 5) =

ENQUANTO ... FAÇA (VALIDAÇÃO NO INÍCIO)

1. valor = 1 **Reiniciando**
2. while valor <= 5:
3. valor = valor + 1
4. print(valor)
5. print(valor)

TESTE DE MESA

valor = 5

print (linha 4) = 5

print (linha 5) =

ENQUANTO ... FAÇA (VALIDAÇÃO NO INÍCIO)

```
1. valor = 1
2. while valor <= 5:
3.     valor = valor + 1
4.     print(valor)
5. print(valor)
```

5<=5?

TESTE DE MESA

valor = 5

print (linha 4) = 5

print (linha 5) =

ENQUANTO ... FAÇA (VALIDAÇÃO NO INÍCIO)

```
1. valor = 1
2. while valor <= 5:
3.     valor = valor + 1
4.     print(valor)
5. print(valor)
```

5<=5? TRUE

TESTE DE MESA

valor = 5

print (linha 4) = 5

print (linha 5) =

ENQUANTO ... FAÇA (VALIDAÇÃO NO INÍCIO)

```
1. valor = 1
2. while valor <= 5:
3.     valor = valor + 1
4.     print(valor)
5. print(valor)
```

5 + 1 = 6

TESTE DE MESA

valor = 6

print (linha 4) = 5

print (linha 5) =

ENQUANTO ... FAÇA (VALIDAÇÃO NO INÍCIO)

```
1. valor = 1
2. while valor <= 5:
3.     valor = valor + 1
4.     print(valor)
5. print(valor)
```

Repete LOOP ↑

TESTE DE MESA

valor = 6

print (linha 4) = 6

print (linha 5) =

ENQUANTO ... FAÇA (VALIDAÇÃO NO INÍCIO)

1. valor = 1 **Reiniciando**
2. while valor <= 5:
3. valor = valor + 1
4. print(valor)
5. print(valor)

TESTE DE MESA

valor = 6

print (linha 4) = 6

print (linha 5) =

ENQUANTO ... FAÇA (VALIDAÇÃO NO INÍCIO)

```
1. valor = 1
2. while valor <= 5:
3.     valor = valor + 1
4.     print(valor)
5. print(valor)
```

6<=5?

TESTE DE MESA

valor = 6

print (linha 4) = 6

print (linha 5) =

ENQUANTO ... FAÇA (VALIDAÇÃO NO INÍCIO)

```
1. valor = 1
2. while valor <= 5:
3.     valor = valor + 1
4.     print(valor)
5. print(valor)
```

6<=5? FALSE

TESTE DE MESA

valor = 6

print (linha 4) = 6

print (linha 5) =

ENQUANTO ... FAÇA (VALIDAÇÃO NO INÍCIO)

```
1. valor = 1
2. while valor <= 5:
3.     valor = valor + 1
4.     print(valor)
5. print(valor)
```

LOOP ENCERRA

TESTE DE MESA

valor = 6

print (linha 4) = 6

print (linha 5) =

ENQUANTO ... FAÇA (VALIDAÇÃO NO INÍCIO)

```
1. valor = 1
2. while valor <= 5:
3.     valor = valor + 1
4.     print(valor)
5. print(valor)
```

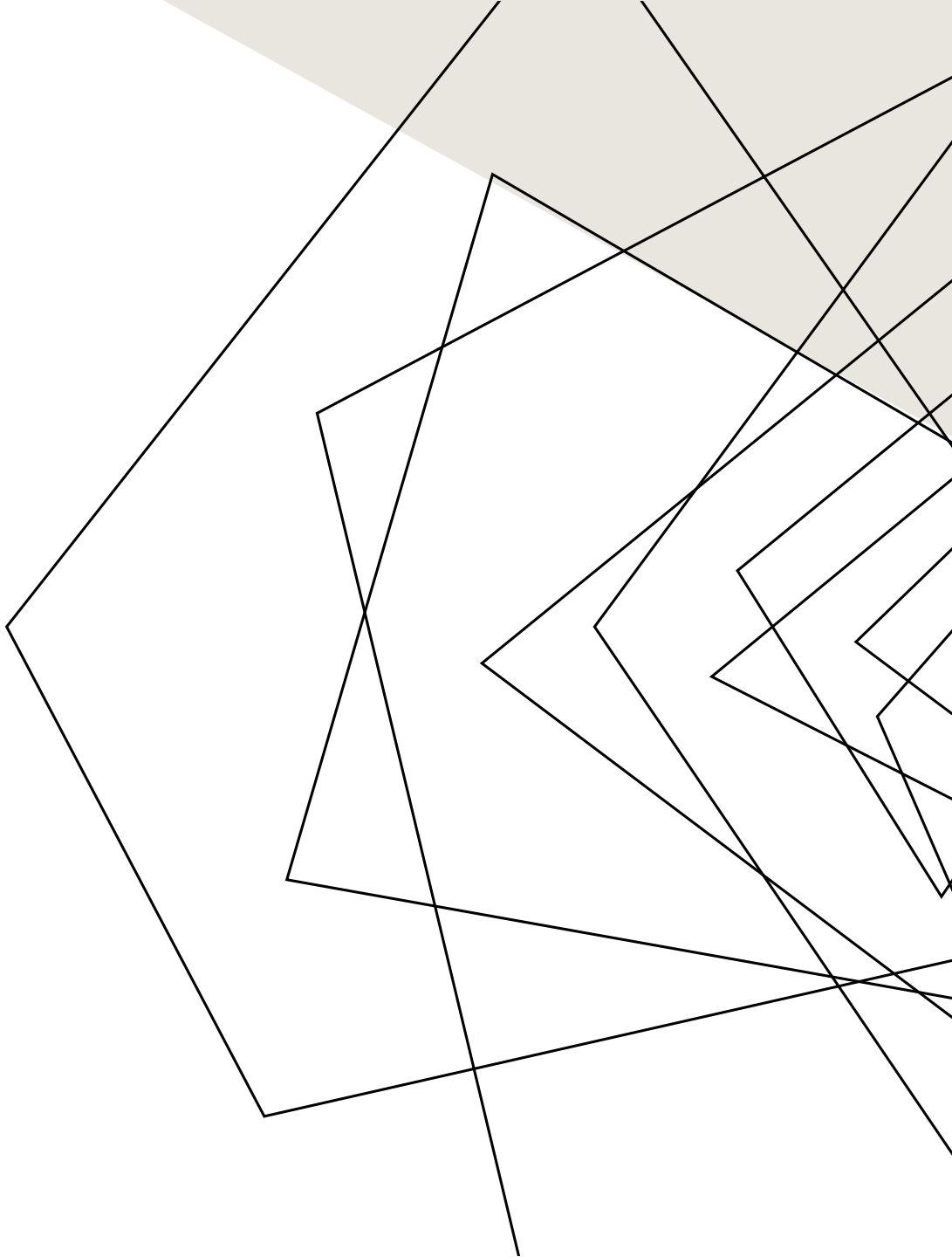
Segue o código

TESTE DE MESA

valor = 6

print (linha 4) = 6

print (linha 5) = 6



FAÇA ... ENQUANTO (VALIDAÇÃO NO FIM)

```
1. valor = 1
2. while True:
3.     valor = valor + 1
4.     print(valor)
5.     if valor > 3:
6.         break
7. print(valor)
```

TESTE DE MESA

valor = 1

print (linha 4) =

print (linha 7) =

FAÇA ... ENQUANTO (VALIDAÇÃO NO FIM)

```
1. valor = 1
2. while True:
3.     valor = valor + 1
4.     print(valor)
5.     if valor > 3:
6.         break
7. print(valor)
```

Inicia o LOOP

TESTE DE MESA

valor = 1

print (linha 4) =

print (linha 7) =

FAÇA ... ENQUANTO (VALIDAÇÃO NO FIM)

```
1. valor = 1
2. while True:
3.     valor = valor + 1
4.     print(valor)
5.     if valor > 3:
6.         break
7. print(valor)
```

1 + 1 = 2

TESTE DE MESA

valor = 2

print (linha 4) =

print (linha 7) =

FAÇA ... ENQUANTO (VALIDAÇÃO NO FIM)

```
1. valor = 1
2. while True:
3.     valor = valor + 1
4.     print(valor)
5.     if valor > 3:
6.         break
7. print(valor)
```

TESTE DE MESA

valor = 2

print (linha 4) = 2

print (linha 7) =

FAÇA ... ENQUANTO (VALIDAÇÃO NO FIM)

```
1. valor = 1
2. while True:
3.     valor = valor + 1
4.     print(valor)
5.     if valor > 3:
6.         break
7. print(valor)
```

2>3?

TESTE DE MESA

valor = 2

print (linha 4) = 2

print (linha 7) =

FAÇA ... ENQUANTO (VALIDAÇÃO NO FIM)

```
1. valor = 1
2. while True:
3.     valor = valor + 1
4.     print(valor)
5.     if valor > 3:
6.         break
7. print(valor)
```

2>3? FALSE

TESTE DE MESA

valor = 2

print (linha 4) = 2

print (linha 7) =

FAÇA ... ENQUANTO (VALIDAÇÃO NO FIM)

```
1. valor = 1
2. while True:
3.     valor = valor + 1
4.     print(valor)
5.     if valor > 3:
6.         break
7. print(valor)
```

Não entra no IF

TESTE DE MESA

valor = 2

print (linha 4) = 2

print (linha 7) =

FAÇA ... ENQUANTO (VALIDAÇÃO NO FIM)

```
1. valor = 1
2. while True:
3.     valor = valor + 1
4.     print(valor)
5.     if valor > 3:
6.         break
7. print(valor)
```

Repete LOOP ↑

TESTE DE MESA

valor = 2

print (linha 4) = 2

print (linha 7) =

FAÇA ... ENQUANTO (VALIDAÇÃO NO FIM)

```
1. valor = 1
2. while True:
3.     valor = valor + 1
4.     print(valor)
5.     if valor > 3:
6.         break
7. print(valor)
```

Inicia o LOOP

TESTE DE MESA

valor = 2

print (linha 4) = 2

print (linha 7) =

FAÇA ... ENQUANTO (VALIDAÇÃO NO FIM)

```
1. valor = 1
2. while True:
3.     valor = valor + 1
4.     print(valor)
5.     if valor > 3:
6.         break
7. print(valor)
```

2 + 1 = 3

TESTE DE MESA

valor = 3

print (linha 4) = 2

print (linha 7) =

FAÇA ... ENQUANTO (VALIDAÇÃO NO FIM)

```
1. valor = 1
2. while True:
3.     valor = valor + 1
4.     print(valor)
5.     if valor > 3:
6.         break
7. print(valor)
```

TESTE DE MESA

valor = 3

print (linha 4) = 3

print (linha 7) =

FAÇA ... ENQUANTO (VALIDAÇÃO NO FIM)

```
1. valor = 1
2. while True:
3.     valor = valor + 1
4.     print(valor)
5.     if valor > 3:
6.         break
7. print(valor)
```

3>3?

TESTE DE MESA

valor = 3

print (linha 4) = 3

print (linha 7) =

FAÇA ... ENQUANTO (VALIDAÇÃO NO FIM)

```
1. valor = 1
2. while True:
3.     valor = valor + 1
4.     print(valor)
5.     if valor > 3:
6.         break
7. print(valor)
```

3>3? FALSE

TESTE DE MESA

valor = 3

print (linha 4) = 3

print (linha 7) =

FAÇA ... ENQUANTO (VALIDAÇÃO NO FIM)

```
1. valor = 1
2. while True:
3.     valor = valor + 1
4.     print(valor)
5.     if valor > 3:
6.         break
7. print(valor)
```

Não entra no IF

TESTE DE MESA

valor = 3

print (linha 4) = 3

print (linha 7) =

FAÇA ... ENQUANTO (VALIDAÇÃO NO FIM)

```
1. valor = 1
2. while True:
3.     valor = valor + 1
4.     print(valor)
5.     if valor > 3:
6.         break
7. print(valor)
```

Repete LOOP ↑

TESTE DE MESA

valor = 3

print (linha 4) = 3

print (linha 7) =

FAÇA ... ENQUANTO (VALIDAÇÃO NO FIM)

```
1. valor = 1
2. while True:
3.     valor = valor + 1
4.     print(valor)
5.     if valor > 3:
6.         break
7. print(valor)
```

Inicia o LOOP

TESTE DE MESA

valor = 3

print (linha 4) = 3

print (linha 7) =

FAÇA ... ENQUANTO (VALIDAÇÃO NO FIM)

```
1. valor = 1
2. while True:
3.     valor = valor + 1
4.     print(valor)
5.     if valor > 3:
6.         break
7. print(valor)
```

2 + 1 = 3

TESTE DE MESA

valor = 4

print (linha 4) = 3

print (linha 7) =

FAÇA ... ENQUANTO (VALIDAÇÃO NO FIM)

```
1. valor = 1
2. while True:
3.     valor = valor + 1
4.     print(valor)
5.     if valor > 3:
6.         break
7. print(valor)
```

TESTE DE MESA

valor = 4

print (linha 4) = 4

print (linha 7) =

FAÇA ... ENQUANTO (VALIDAÇÃO NO FIM)

```
1. valor = 1
2. while True:
3.     valor = valor + 1
4.     print(valor)
5.     if valor > 3:
6.         break
7. print(valor)
```

4>3?

TESTE DE MESA

valor = 4

print (linha 4) = 4

print (linha 7) =

FAÇA ... ENQUANTO (VALIDAÇÃO NO FIM)

```
1. valor = 1
2. while True:
3.     valor = valor + 1
4.     print(valor)
5.     if valor > 3:
6.         break
7. print(valor)
```

4>3? TRUE

TESTE DE MESA

valor = 4

print (linha 4) = 4

print (linha 7) =

FAÇA ... ENQUANTO (VALIDAÇÃO NO FIM)

```
1. valor = 1
2. while True:
3.     valor = valor + 1
4.     print(valor)
5.     if valor > 3:
6.         break
7. print(valor)
```

Entra no IF

TESTE DE MESA

valor = 4

print (linha 4) = 4

print (linha 7) =

FAÇA ... ENQUANTO (VALIDAÇÃO NO FIM)

```
1. valor = 1
2. while True:
3.     valor = valor + 1
4.     print(valor)
5.     if valor > 3:
6.         break
7. print(valor)
```

Executa o break

TESTE DE MESA

valor = 4

print (linha 4) = 4

print (linha 7) =

FAÇA ... ENQUANTO (VALIDAÇÃO NO FIM)

```
1. valor = 1
2. while True:
3.     valor = valor + 1
4.     print(valor)
5.     if valor > 3:
6.         break
7. print(valor)
```

Fim IF e fim LOOP

TESTE DE MESA

valor = 4

print (linha 4) = 4

print (linha 7) =

FAÇA ... ENQUANTO (VALIDAÇÃO NO FIM)

```
1. valor = 1
2. while True:
3.     valor = valor + 1
4.     print(valor)
5.     if valor > 3:
6.         break
7. print(valor)
```

TESTE DE MESA

valor = 4

print (linha 4) = 4

print (linha 7) = 4

PARA ... PASSO ... FAÇA (FOR)

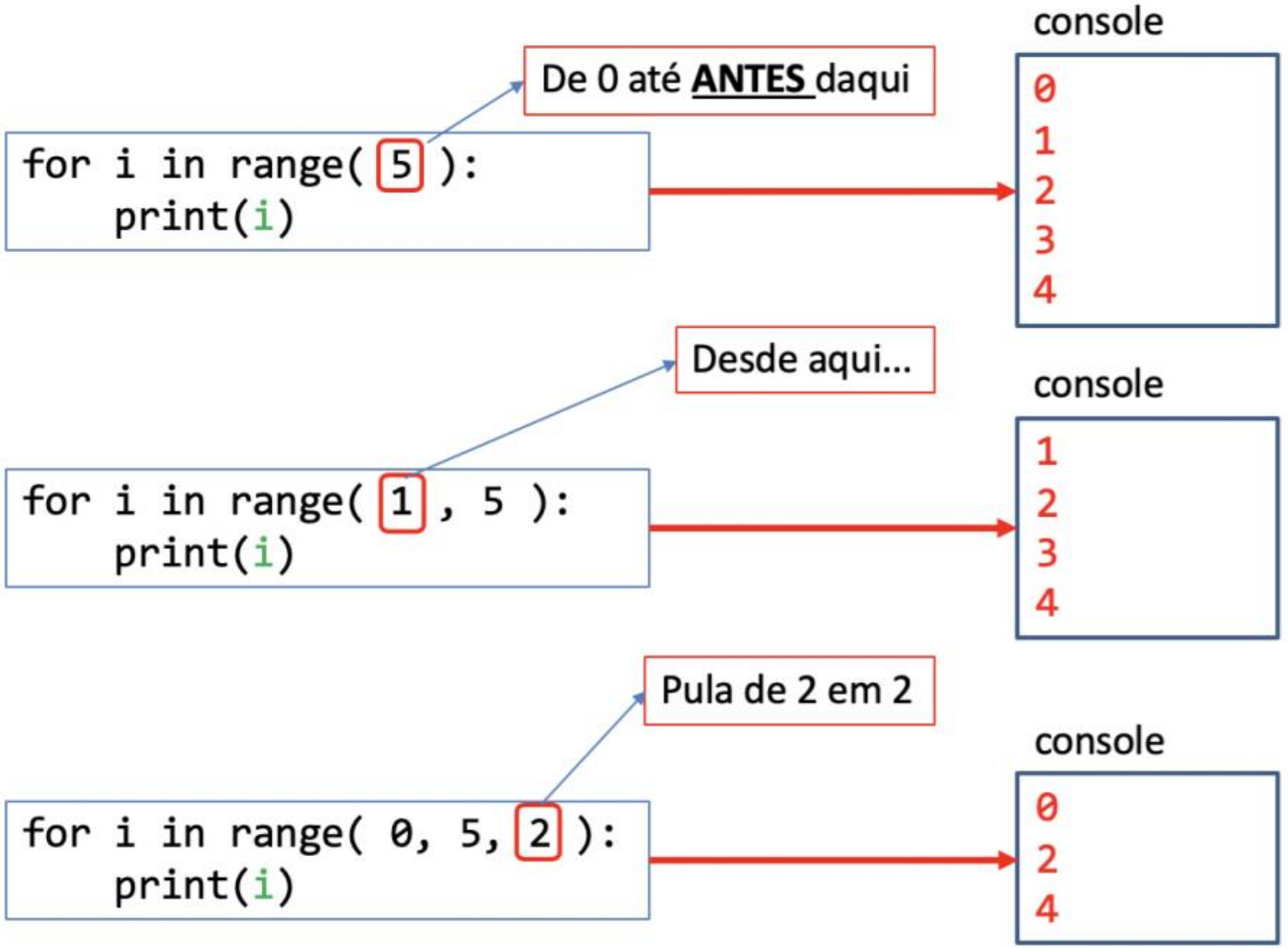
- Este tipo de estrutura permite controlar o número de vezes que as instruções devem ser repetidas. Portanto, sua principal característica é saber previamente o número de vezes que a estrutura será repetida, como, por exemplo, antes de bloquear um cartão magnético, o usuário terá a oportunidade de digitar sua senha três vezes.
- Sua sintaxe é:

```
PARA <variável> DE <valor inicial> ATÉ <valor final> PASSO <incremento> FAÇA

    instrução1
    instrução2
    ...
    instruçãon

FIM PARA
```

PARA ... PASSO ... FAÇA (FOR)

- Após a declaração da variável, um valor inicial é definido para ela. Esse valor inicial será comparado ao valor final.
- Caso a variável seja menor ou igual ao valor final, serão executadas as instruções existentes dentro da rotina de repetição.
    - incrementa-se o valor da variável em uma unidade;
    - verifica-se novamente se variável está dentro dos limites em que se repete a tarefa de comparação da variável, e se ela é verdadeira.
- Caso a variável contenha um valor maior que o declarado como valor final, será executada a instrução, ou conjunto de instruções, logo abaixo da instrução de finalização da repetição (FIM PARA).
- Caso não seja verificada a opção PASSO, o contador será incrementado automaticamente em uma unidade. Portanto, contador terá seu passo determinado com avanço de uma em uma unidade.

PARA ... PASSO ... FAÇA (FOR)



PARA ... PASSO ... FAÇA (FOR) CONTAR DE 0 A 4

```
1. for valor in range(5):
2.     print(valor)
```

"Eu preciso declarar o valor **?"**

Em Python, o próprio laço for se encarrega de criar a variável de controle (i ou qualquer outro nome que você usar) no momento em que o laço começa.

TESTE DE MESA

valor =

print (linha 2) =

PARA ... PASSO ... FAÇA (FOR) CONTAR DE 0 A 4

Início do LOOP

```
1. for valor in range(5):
2.     print(valor)
```

TESTE DE MESA

valor = 0

print (linha 2) =

PARA ... PASSO ... FAÇA (FOR) CONTAR DE 0 A 4

```
1. for valor in range(5):
2.     print(valor)
```

TESTE DE MESA

valor = 0

print (linha 2) = 0

PARA ... PASSO ... FAÇA (FOR) CONTAR DE 0 A 4

```
1. for valor in range(5):
2.     print(valor)
```

Repete LOOP ↑

Esse tipo de LOOP é mais robusto porque faz a soma automática, caso não seja definido.

TESTE DE MESA

valor = 0

print (linha 2) = 0

PARA ... PASSO ... FAÇA (FOR) CONTAR DE 0 A 4

```
1. for valor in range(5):
2.     print(valor)
```

TESTE DE MESA

valor = 1

print (linha 2) = 0

PARA ... PASSO ... FAÇA (FOR) CONTAR DE 0 A 4

```
1. for valor in range(5):
2.     print(valor)
```

TESTE DE MESA

valor = 1

print (linha 2) = 1

PARA ... PASSO ... FAÇA (FOR) CONTAR DE 0 A 4

```
1. for valor in range(5):
2.     print(valor)
```

Repete LOOP ↑

TESTE DE MESA

valor = 1

print (linha 2) = 1

PARA ... PASSO ... FAÇA (FOR) CONTAR DE 0 A 4

```
1. for valor in range(5):
2.     print(valor)
```

TESTE DE MESA

valor = 2

print (linha 2) = 1

PARA ... PASSO ... FAÇA (FOR) CONTAR DE 0 A 4

```
1. for valor in range(5):
2.     print(valor)
```

TESTE DE MESA

valor = 2

print (linha 2) = 2

PARA ... PASSO ... FAÇA (FOR) CONTAR DE 0 A 4

```
1. for valor in range(5):
2.     print(valor)
```

Repete LOOP ↑

TESTE DE MESA

valor = 2

print (linha 2) = 2

PARA ... PASSO ... FAÇA (FOR) CONTAR DE 0 A 4

```
1. for valor in range(5):
2.     print(valor)
```

TESTE DE MESA

valor = 3

print (linha 2) = 2

PARA ... PASSO ... FAÇA (FOR) CONTAR DE 0 A 4

```
1. for valor in range(5):
2.     print(valor)
```

TESTE DE MESA

valor = 3

print (linha 2) = 3

PARA ... PASSO ... FAÇA (FOR) CONTAR DE 0 A 4

```
1. for valor in range(5):
2.     print(valor)
```

Repete LOOP ↑

TESTE DE MESA

valor = 3

print (linha 2) = 3

PARA ... PASSO ... FAÇA (FOR) CONTAR DE 0 A 4

```
1. for valor in range(5):
2.     print(valor)
```

TESTE DE MESA

valor = 4

print (linha 2) = 3

PARA ... PASSO ... FAÇA (FOR) CONTAR DE 0 A 4

```
1. for valor in range(5):
2.     print(valor)
```

TESTE DE MESA

valor = 4

print (linha 2) = 4

PARA ... PASSO ... FAÇA (FOR) CONTAR DE 0 A 4

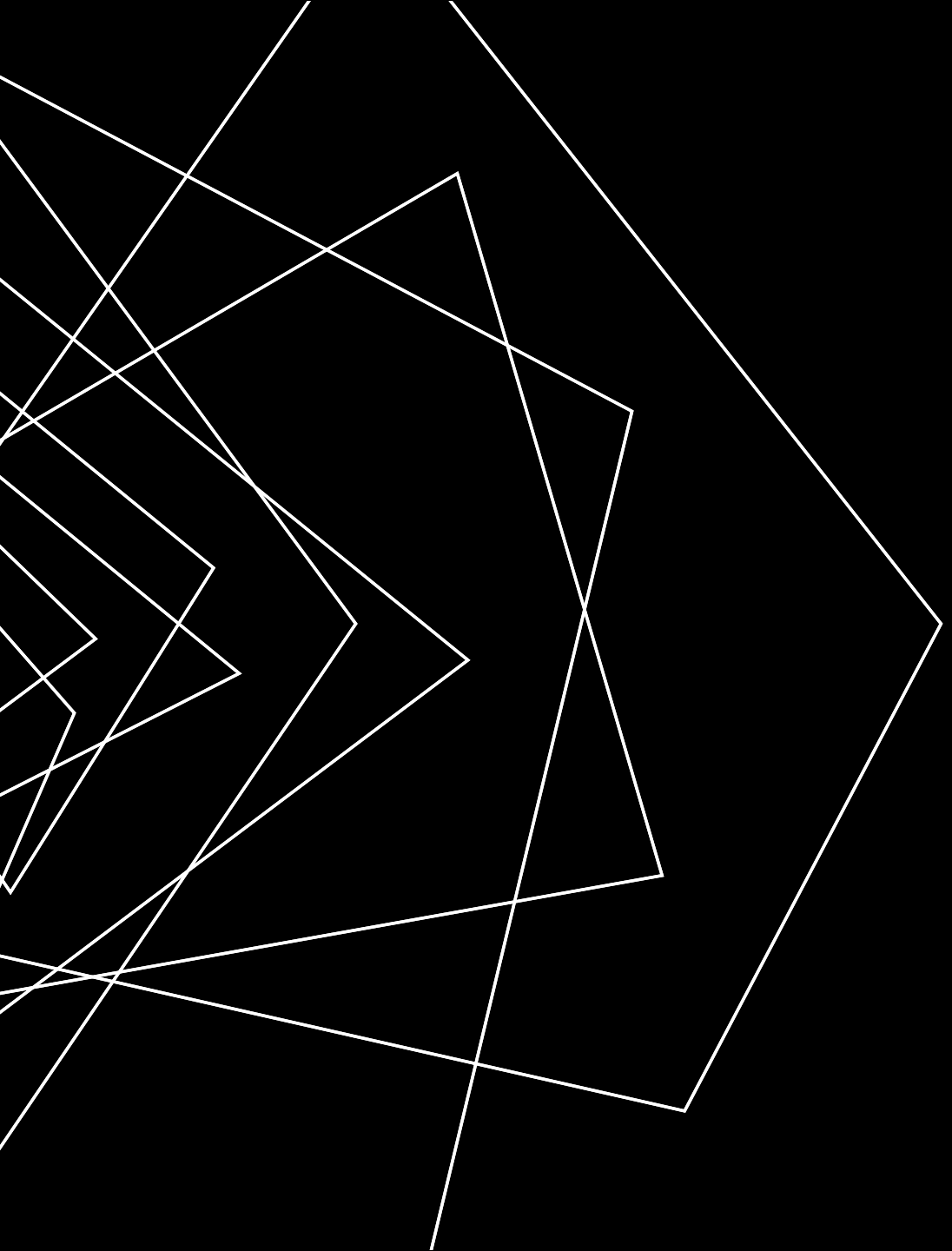
```
1. for valor in range(5):
2.     print(valor)
```

FIM

TESTE DE MESA

valor = 4

print (linha 2) = 4



FIM DE AULA

ESTRUTURAS DE LISTAS

AULA 10

PROFESSOR: EDUARDO KAZENSKI

CONTEÚDO

- Estrutura de Listas;

OBJETIVOS

- Conseguir aplicar as estruturas corretas conforme o algoritmo exigir em sua lógica.
- Entender como as repetições são representadas nos algoritmos.

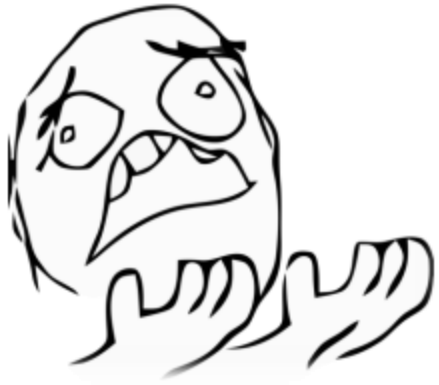
ESTRUTURAS DE LISTAS

- Construa um programa que registre o nome dos alunos de uma turma. A turma tem 3 alunos. O programa deve ler o nome de cada aluno e registrar.

```
aluno1 = input("Informe seu nome: ")
aluno2 = input("Informe seu nome: ")
aluno3 = input("Informe seu nome: ")

print("Os alunos da turma são: \n%s \n%s \n%s" %(aluno1,aluno2,aluno3))
```

ESTRUTURAS DE LISTAS

- Vamos alterar um pouco os requisitos do programa:
  - Construa um programa que registre o nome dos alunos de uma turma. A turma pode variar a quantidade de alunos. O programa deve ler a quantidade de alunos da turma e ler o nome de cada aluno e registrar.

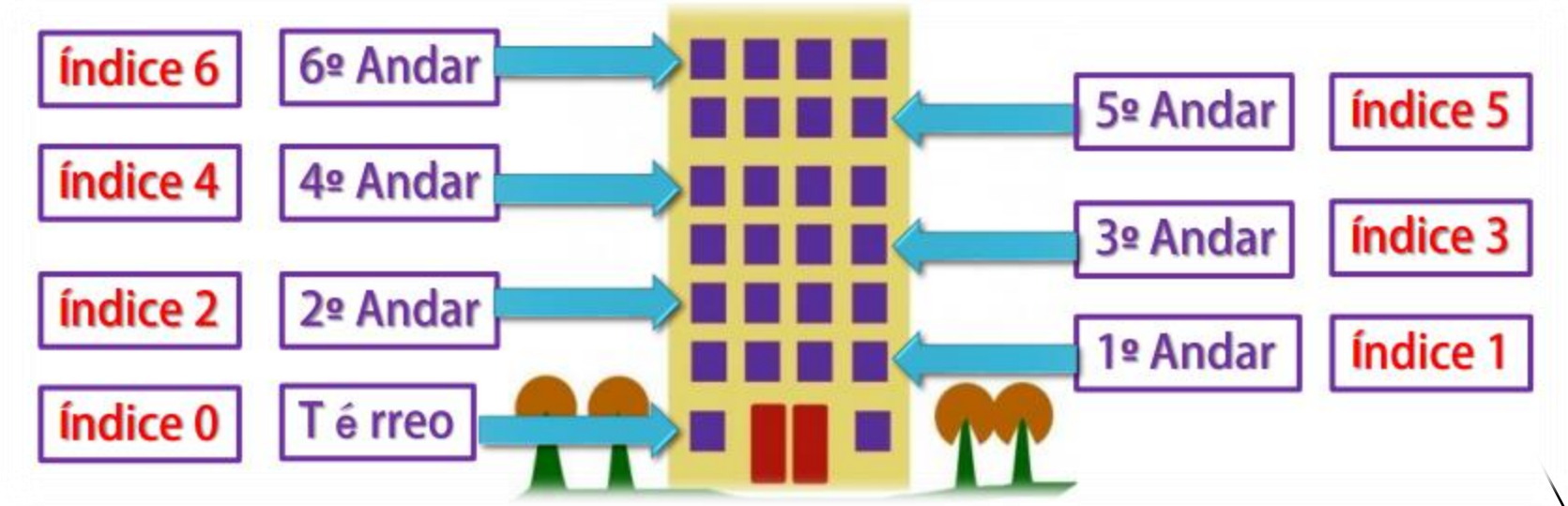


O QUE SÃO LISTAS?

- Uma lista é um conjunto de valores ordenados.
- Os valores são identificados por índices e chamamos de elementos.
- Uma lista pode conter zero ou mais elementos de qualquer tipo (em Python usa-se tipagem dinâmica)
- A medida que novos elementos são adicionados ou removidos o tamanho da lista cresce ou diminui.
- Exemplo: [10,20,30,40,50,60,70,80,90,100]

O QUE SÃO ÍNDICES?

- Índice é uma referência a posição do elemento dentro da lista.
- Os índices se iniciam em zero.
- Imagine uma lista como um edifício de apartamentos, onde o térreo é o andar zero, o primeiro andar é o 1, conforme o exemplo abaixo:



DECLARAÇÃO DE LISTAS

- Elementos envolvidos por colchetes [].
- Declaração e inicialização de uma lista vazia: exemplo = []
- Declaração e inicialização de uma lista com 5 elementos: exemplo = [10,50,5,2,100]
- Exemplo = [“algoritmo”, “é”, “muito”, “bom”, “e fácil”]
- Declaração e inicialização de uma lista sequencial com o range: **minhaLista = range(10)**
- Declaração e inicialização de uma lista ANINHADA: **suaLista = [10,50,5, [2, 100], “texto”, 3.0]**

ACESSO AOS ELEMENTOS DE UMA LISTA

- A forma de acesso a cada elemento de uma lista é através dos índices (endereços) de cada elemento:

```
suaLista = [10, 50, 5, 2, 100]

# Usamos o 'for' para percorrer cada item da lista
# O Python sabe quantos itens a lista tem e para quando a iteração acabar
print("--- Imprimindo com o laço FOR ---")
for item in suaLista:
    print(item)
```

ADICIONAR ELEMENTOS EM UMA LISTA

- Uma das maiores vantagens da utilização de listas é a possibilidade de adicionar elementos durante a execução do programa.
- Na prática o programador não precisa se preocupar com a quantidade de variáveis para os alunos da turma (lembra desse problema?)
- O método append():

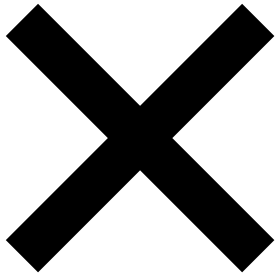
```
# Criamos uma lista completamente vazia
lista_numeros = []

print("Lista no início:", lista_numeros)

# O 'for' vai gerar os números de 1 até 5
for i in range(1, 6):
    # Usamos o .append() para adicionar o número atual (i) ao final da lista
    lista_numeros.append(i)
    print(f"Adicionando o número {i}. A lista agora é: {lista_numeros}")

print("\nProcesso concluído! A lista final é:", lista_numeros)
```

ADICIONAR ELEMENTOS EM UMA LISTA

- Uma das maiores vantagens da utilização de listas é a possibilidade de adicionar elementos durante a execução do programa.
- Na prática o programador não precisa se preocupar com a quantidade de variáveis para os alunos da turma (lembra desse problema?)
- O método append():

```
# Criamos uma lista completamente vazia
lista_numeros = []

print("Lista no início:", lista_numeros)

# O 'for' vai gerar os números de 1 até 5
for i in range(1, 6):
    # Usamos o .append() para adicionar o número atual (i) ao final da lista
    lista_numeros.append(i)
    print(f"Adicionando o número {i}. A lista agora é: {lista_numeros}")

print("\nProcesso concluído! A lista final é:", lista_numeros)
```

REMOVER ELEMENTOS DE UMA LISTA (1/2)



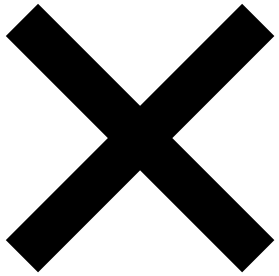
- Para remover elementos da lista há dois métodos:

  del: Apaga o elemento da lista.

  Exemplo:

```
# A nossa lista de números, já pronta
lista_numeros = [1, 2, 3, 4, 5]

print("Lista original:", lista_numeros)

# --- Removendo um item específico pelo seu índice ---
# O número 3 está na posição (índice) 2 da lista.
# (Lembre-se: a contagem começa do 0!)
del lista_numeros[2]

print(f"\nApós remover o item do índice 2 (o número 3):")
print(lista_numeros)

# --- Removendo o último item da lista ---
# Podemos usar o índice negativo -1 para acessar o último item.
del lista_numeros[-1]

print(f"\nApós remover o último item (o número 5):")
print(lista_numeros)
```

REMOVER ELEMENTOS DE UMA LISTA (2/2)



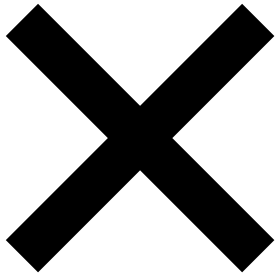
- Para remover elementos da lista há dois métodos:

  pop: Apaga o elemento da lista e retorna o elemento removido.

  Exemplo:

```python
# Nossa lista inicial de números
lista_numeros = [1, 2, 3, 4, 5]

print("Lista original:", lista_numeros)

# Usando .pop() para remover e guardar o último item
# Se você não informar o índice, o .pop() remove o último item por padrão.
item_removido = lista_numeros.pop()

print(f"\nItem removido (o último): {item_removido}")
print("Lista após o .pop():", lista_numeros)

# Usando .pop() para remover e guardar um item de um índice específico
# Vamos remover o item no índice 1 (que é o número 2).
item_removido_especifico = lista_numeros.pop(1)

print(f"\nItem removido do índice 1: {item_removido_especifico}")
print("Lista após o segundo .pop():", lista_numeros)
```

REMOVER ELEMENTOS DE UMA LISTA (2/2)

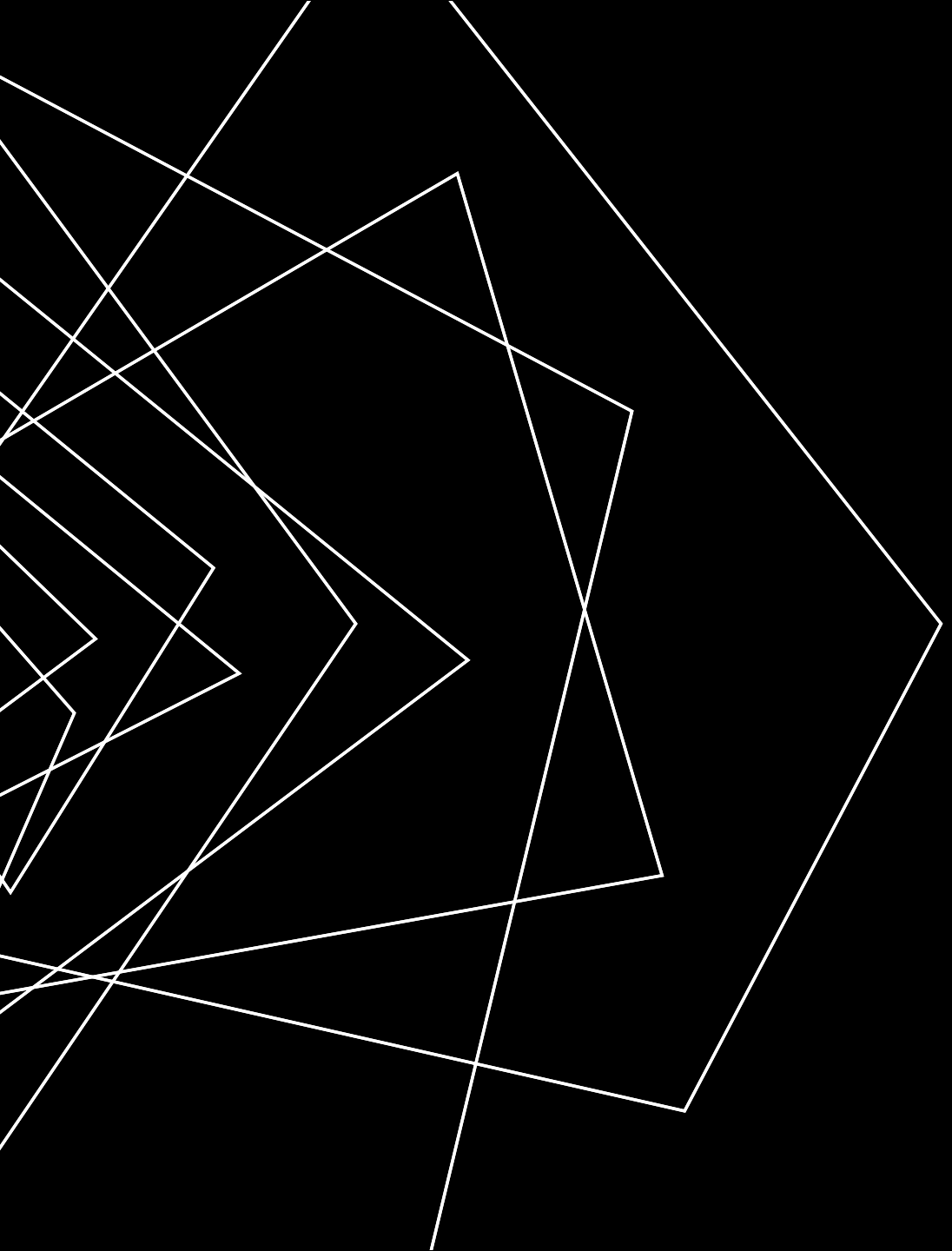


- O método **.pop()** é outra ferramenta super útil para remover itens.
- Qual é a diferença dele para o **del** que vimos antes?
- O comando **del** apaga o item sem devolver nada. É como usar uma borracha.
- O método **.pop()** remove o item, mas também o devolve. É como tirar um livro de uma prateleira para ler, e a prateleira fica vazia naquele lugar. Essa característica permite que você guarde o item removido em uma variável para usá-lo depois.

TAMANHO DE UMA LISTA

- O tamanho de uma lista é igual à quantidade de elementos que ela contém.
- Para descobrir o tamanho de uma lista existe a função **len**.
- A função **len** retorna a quantidade de elementos na lista.

```
# Nossa lista inicial de números
lista_numeros = [1, 2, 3, 4, 5, 6, 7]

print("Lista original:", lista_numeros)
print("Tamanho da lista:", len(lista_numeros))
```



FIM DE AULA