



P Y T H O N & S U A S A P L I C A Ç Õ E S

---

# PRIMEIROS PASSOS COM A LINGUAGEM

POR JOÃO VÍTOR PAMPONET ESTEVES

# O QUE IREMOS APRENDER

---

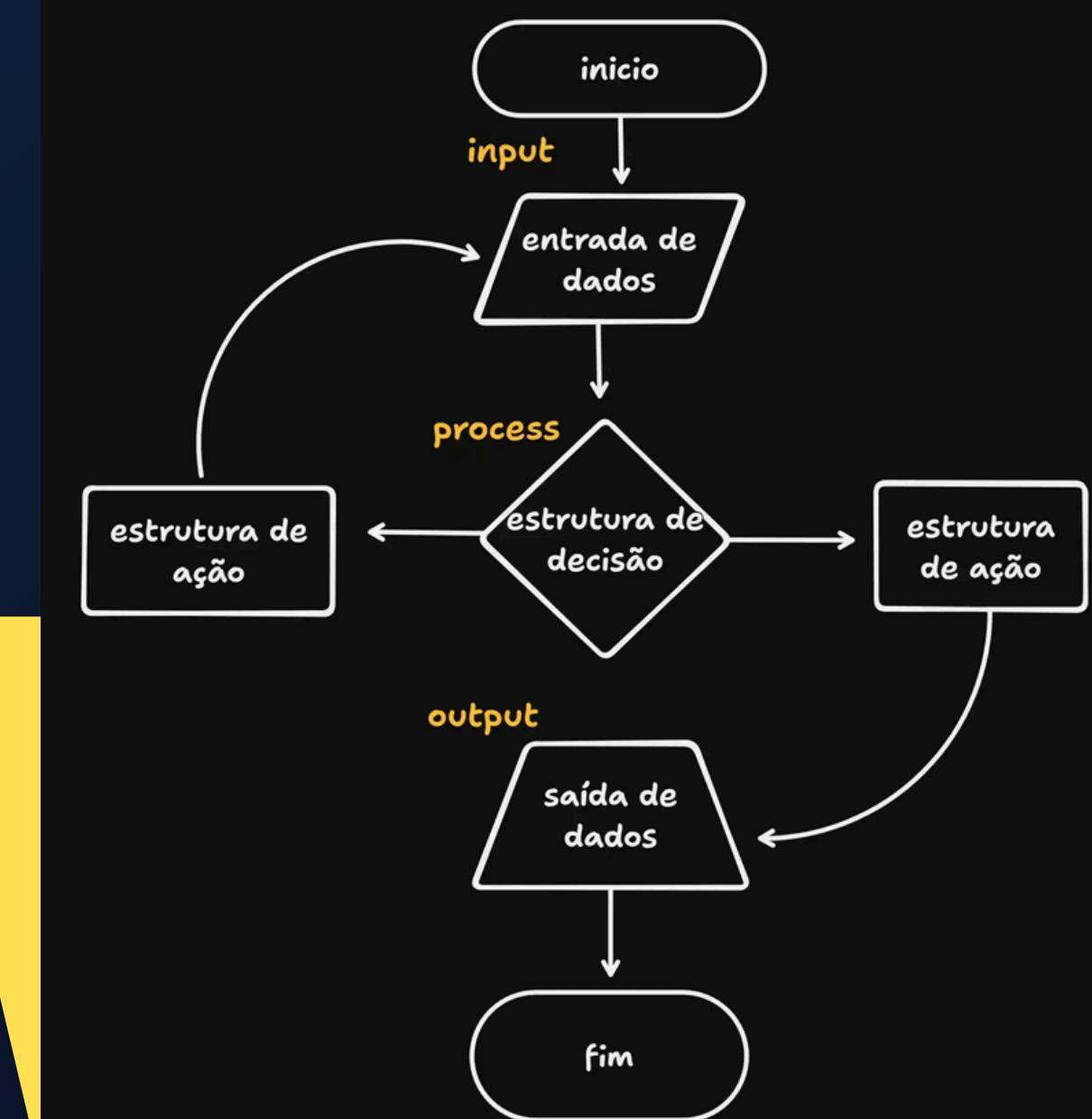
- 1      Estruturas condicionais
- 2      Operadores lógicos
- 3      Laços de repetição
- 4      Exercícios práticos

# UTILIDADE DAS ESTRUTURAS CONDICIONAIS

As estruturas condicionais são um dos pilares da programação, pois permitem que os programas tomem decisões de forma automática, baseadas em situações específicas.

Em vez de executar sempre a mesma sequência de comandos, o programa pode se adaptar, escolhendo caminhos diferentes conforme as condições apresentadas.

Isso traz flexibilidade e torna o código mais próximo da lógica que aplicamos no nosso dia a dia.



# UTILIDADE DAS ESTRUTURAS CONDICIONAIS

---



No cotidiano, utilizamos esse tipo de raciocínio o tempo todo: se estiver chovendo, levo um guarda-chuva; se o sinal estiver verde, atravesso; caso contrário, espero.

Na programação, o mesmo ocorre.

Por meio das estruturas condicionais, conseguimos criar programas que simulam esse processo de decisão, garantindo que os resultados variem conforme as entradas recebidas.

## EXEMPLOS DO DIA A DIA

### DECISÃO SOBRE O GUARDA-CHUVA



Está Chovendo?

Sim



Levar guarda-chuva



Não



Não levar  
guarda-chuva

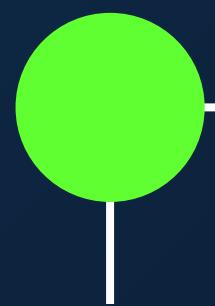


# EXEMPLOS DO DIA A DIA

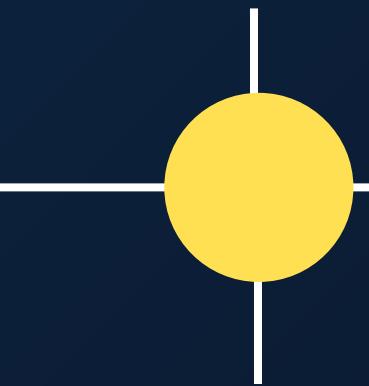
## COR DO SINAL



Verde



Atravesse



Amarelo



Vermelho



Espere

Não Atravesse



# ESTRUTURAS CONDICIONAIS

---

As estruturas condicionais permitem que o programa tome decisões com base em uma condição lógica.

**if**

executa um bloco de código **se a condição for verdadeira.**

**elif**

usado quando há **mais de uma condição a ser testada.**

**else**

executa um bloco de código caso **nenhuma condição seja satisfeita.**

# EXEMPLO EM CÓDIGO

---

Programa que verifica se a eficiência de uma máquina está dentro do esperado.

```
eficiencia = float(input("Digite a eficiência da máquina (%): "))

if eficiencia >= 90:
    print("Excelente desempenho.")

elif eficiencia >= 75:
    print("Bom desempenho.")

else:
    print("Necessário manutenção.")
```

# OPERADORES LÓGICOS

---

Os operadores lógicos permitem combinar múltiplas condições dentro das estruturas condicionais, tornando as análises mais precisas.

**and**

todas as condições precisam ser verdadeiras.

**or**

pelo menos uma condição precisa ser verdadeira.

**not**

inverte o valor booleano de uma expressão



# OPERADOR and (E)

*Tabela Verdade*

Condição 1	Condição 2	Resultado
True	True	True
True	False	False
False	True	False
False	False	False

# EXEMPLO EM CÓDIGO

---

Programa que verifica se um material atende a DOIS critérios técnicos simultaneamente para aprovação no controle de qualidade.

```
resistencia = 85
dureza = 70

if resistencia >= 80 and dureza >= 65:
    print("✅ Material APROVADO nos testes de qualidade")

else:
    print("❌ Material REPROVADO - Não atende aos critérios")
```

# OPERADOR or (OU)

*Tabela Verdade*

Condição A

Condição B

A or B

True

True

True

True

False

True

False

True

True

False

False

False

# EXEMPLO EM CÓDIGO

---

Sistema de monitoramento que dispara um alarme se pelo menos UMA condição crítica for detectada nos parâmetros operacionais.

```
temperatura = 150
pressao = 8

if temperatura > 120 or pressao < 5:
    print("⚠ ALARME: Condições críticas detectadas!")

else:
    print("✓ Parâmetros dentro da normalidade")
```

# OPERADOR **not**

---

*Tabela Verdade*

Condição A	not A
True	False
False	True

# EXEMPLO EM CÓDIGO

---

Sistema que verifica se o modo de manutenção NÃO está ativo para liberar o funcionamento operacional da planta industrial.

```
modo_manutencao = False

if not modo_manutencao:
    print("⚙️ Sistema operacional liberado para produção")

else:
    print("🔧 Modo manutenção ativo - Sistema suspenso")
```

# LAÇOS DE REPETIÇÃO

---



Os laços de repetição permitem que um bloco de código seja executado várias vezes, seja para percorrer uma lista de dados ou para repetir um cálculo até que uma condição seja atendida.

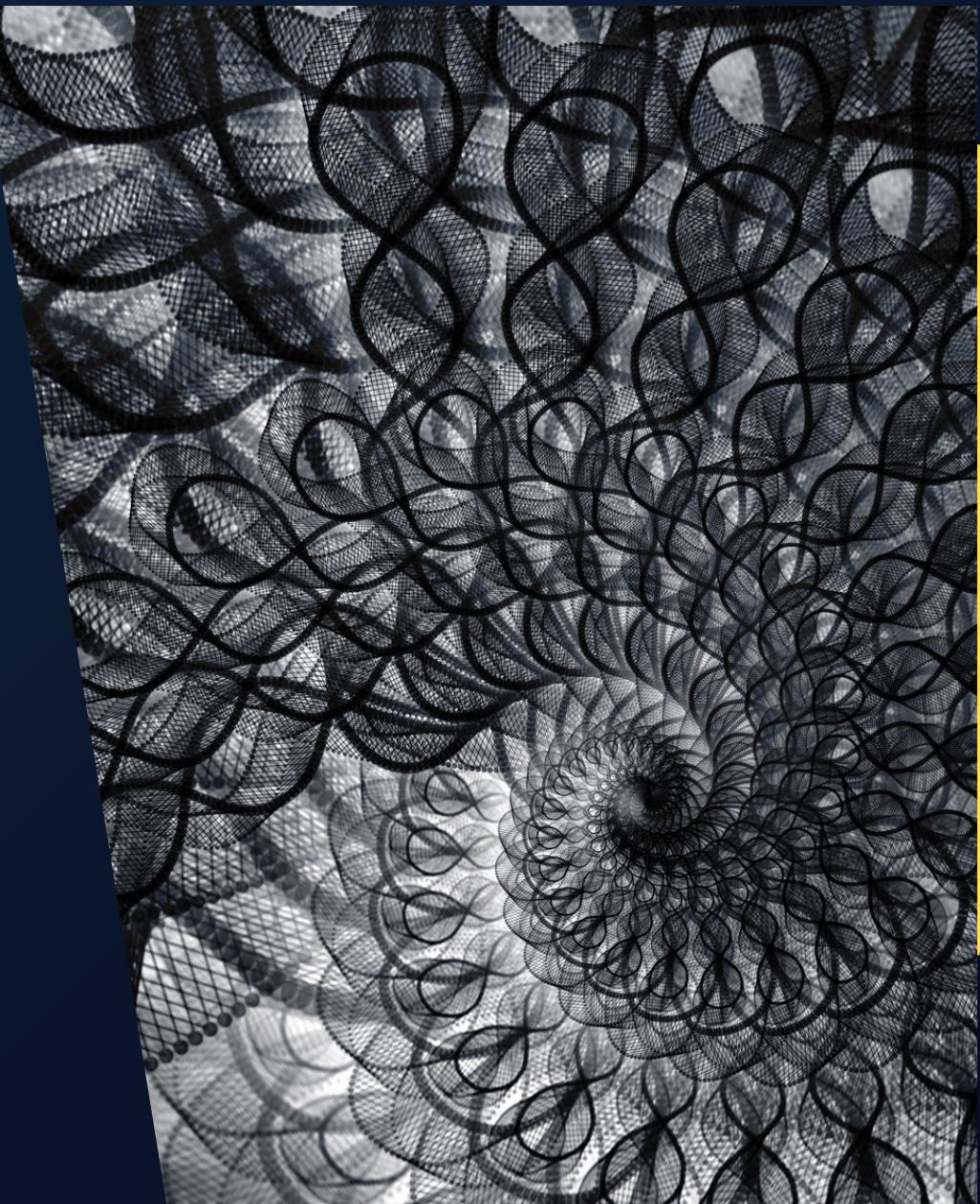
Em Python, esses laços são representados pelas palavras reservadas `for` (ideal para iterar sobre sequências conhecidas) `while` (usado quando a repetição depende de uma condição dinâmica).

# UTILIDADE DOS LAÇOS DE REPETIÇÃO

---

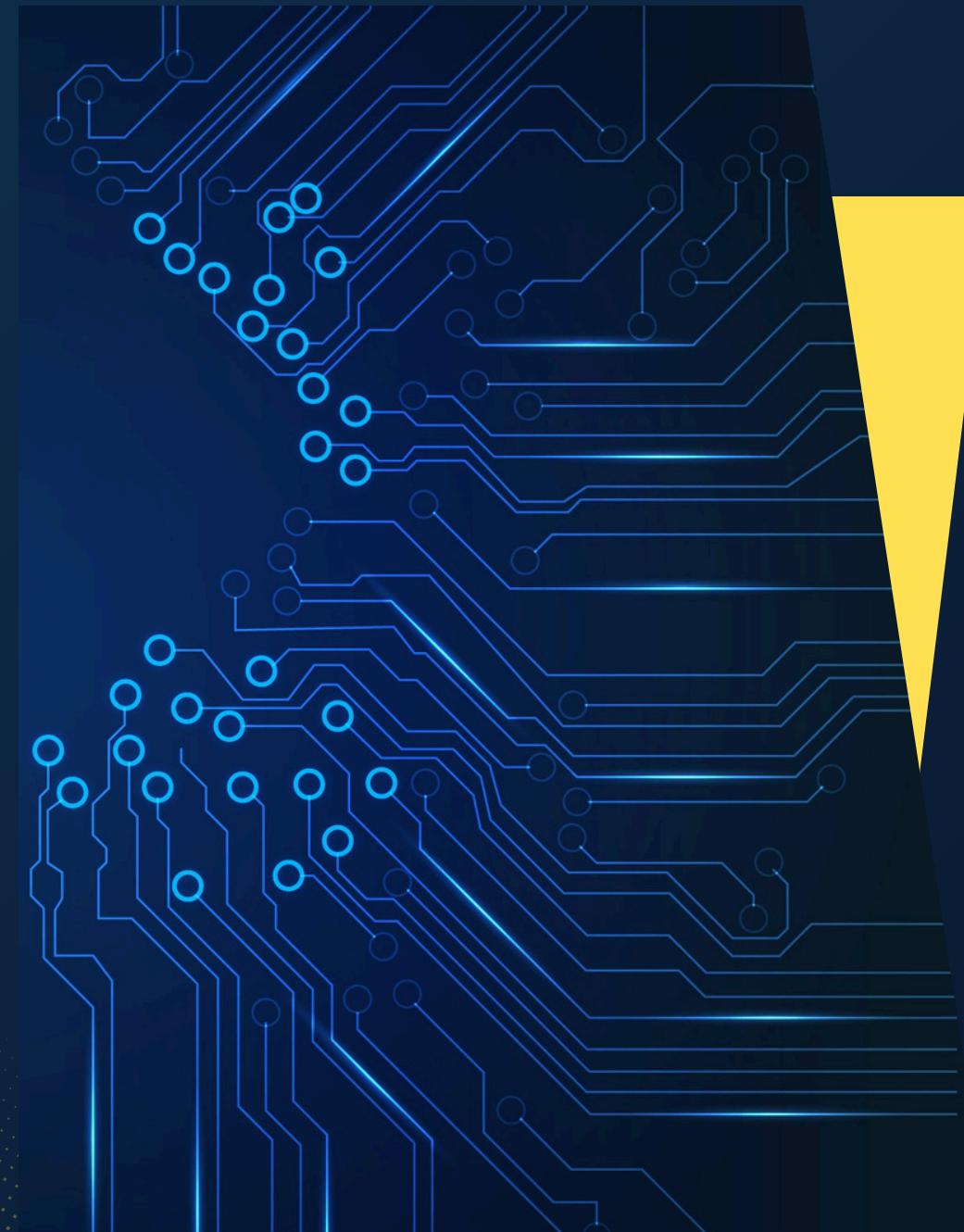
Os laços de repetição são ferramentas fundamentais na programação porque permitem que uma tarefa seja executada automaticamente várias vezes, sem que seja necessário repetir manualmente os comandos.

Isso torna o código mais limpo, eficiente e fácil de manter, principalmente em situações onde lidamos com grandes quantidades de dados ou cálculos repetitivos.



# UTILIDADE DOS LAÇOS DE REPETIÇÃO

---



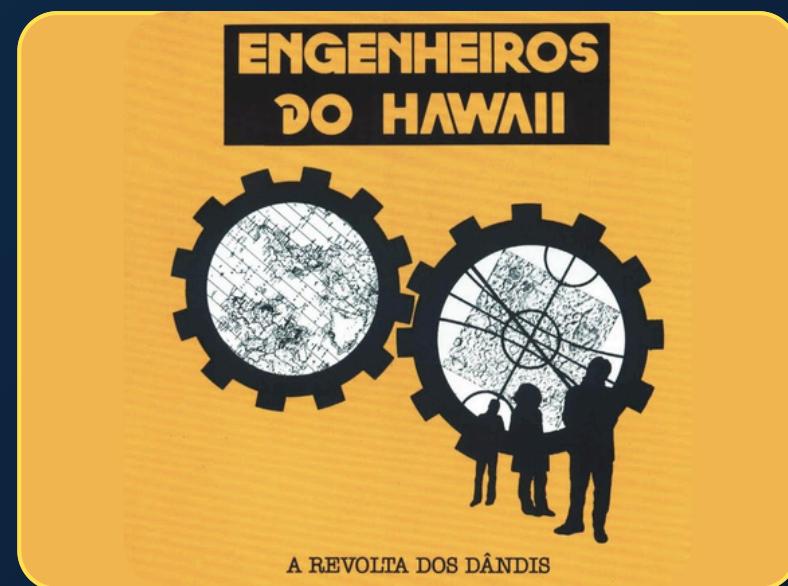
Na prática, eles são aplicados sempre que precisamos percorrer coleções de informações, repetir medições ou executar um mesmo cálculo diversas vezes.

Em vez de escrever várias linhas iguais, o laço cria um fluxo inteligente que se repete até que a condição seja satisfeita.

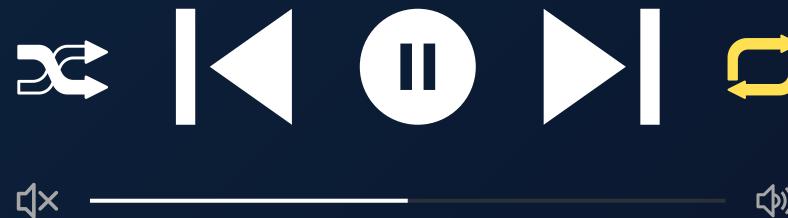
Isso não apenas economiza tempo, mas também reduz a chance de erros humanos.

# EXEMPLOS DO DIA A DIA

## PLAYLIST



Infinita Highway  
Engenheiros do Hawaii



Musica 2  
Engenheiros do Hawaii



Musica 3  
Engenheiros do Hawaii



Musica 4  
Engenheiros do Hawaii



Musica 5  
Engenheiros do Hawaii



Musica 6  
Engenheiros do Hawaii

...

...

...

...

...

Podemos manter a mesma música tocando ou repetir a playlist infinitamente

# EXEMPLOS DO DIA A DIA

---

## VÍDEO GRAVANDO



00:15:03

Podemos programar o fim de uma gravação

# EXEMPLOS

## FOR

Usado quando sabemos previamente quantas vezes repetir.

```
for i in range(5):  
    print("Teste número", i+1)
```

## WHILE

Usado quando a repetição depende de uma condição lógica.

```
contador = 0  
while contador < 3:  
    print("Medição", contador+1)  
    contador += 1
```

# EXEMPLO INTEGRADO

(CONDICIONAIS + OPERADORES LÓGICOS + LAÇO)

Um engenheiro deseja monitorar o funcionamento de uma caldeira em 3 leituras consecutivas.

01

Condições normais: temperatura  $\leq 100$  °C e pressão  $\leq 10$  bar.

02

Se em qualquer leitura essas condições não forem atendidas, exibir alerta e interromper o monitoramento.

03

Caso todas estejam normais, exibir “Sistema em operação segura”.

```
seguro = True
```

```
for i in range(3):
```

```
    temperatura = float(input("Leitura {i+1} - Temperatura (°C): "))  
    pressao = float(input("Leitura {i+1} - Pressão (bar): "))
```

```
if temperatura > 100 or pressao > 10:
```

```
    print("Alerta: Condições inseguras detectadas!")
```

```
    seguro = False
```

```
    break # encerra o laço imediatamente
```

```
if seguro:
```

```
    print("Sistema em operação segura.")
```

# EXERCÍCIOS PRÁTICOS

---

## CÁLCULO DE EFICIÊNCIA

01

Peça a potência de entrada e de saída de uma máquina.

02

Calcule a eficiência ( $\text{eficiência} = (\text{Psaída} / \text{Pentrada}) * 100$ ).

03

Exiba mensagens diferentes dependendo do valor

( $>=90\%$  excelente,  $>=75\%$  bom,  $<75\%$  precisa de manutenção).

# EXERCÍCIOS PRÁTICOS

---

## MÉDIAS DE PRODUÇÃO

01

Receba a produção diária de uma máquina durante 7 dias.

02

Use um laço (`for` ou `while`) para somar os valores.

03

Calcule e exiba a média semanal.

# EXERCÍCIOS PRÁTICOS

---

## CONTROLE DE QUALIDADE

01

Receba **N** medições de resistência de um material.

02

O programa deve verificar se todas as medições estão acima de um limite mínimo estabelecido.

03

Exiba se o lote foi **aprovado** ou **reprovado**

ATÉ A PROXIMA

