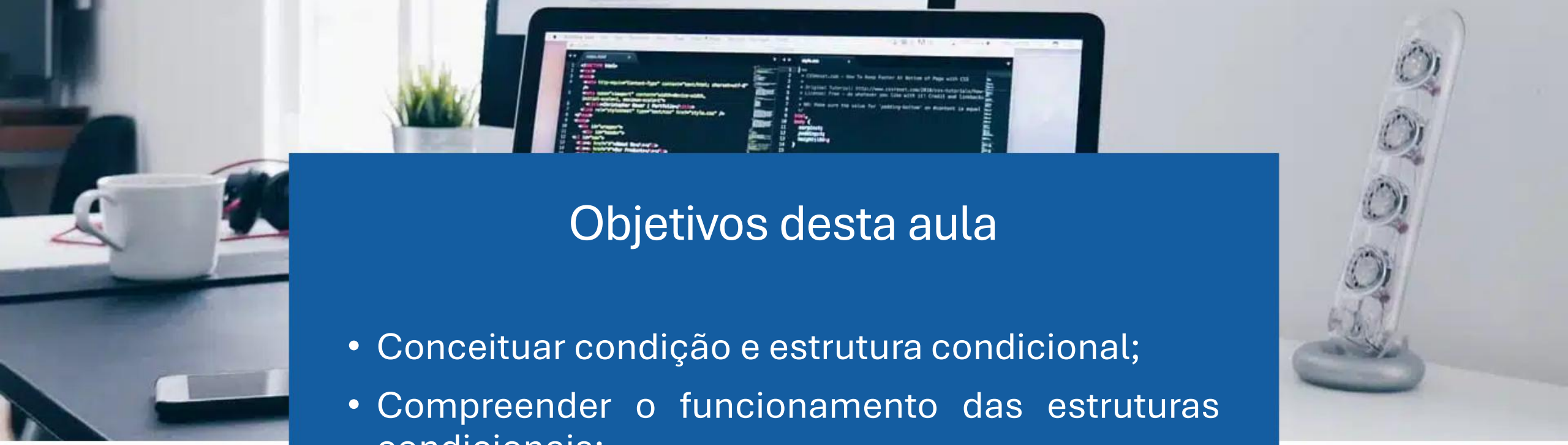
Aula #5:

Estruturas condicionais

Prof. Dr. Luiz Álvaro de Oliveira Júnior



Objetivos desta aula

- Conceituar condição e estrutura condicional;
- Compreender o funcionamento das estruturas condicionais;
- Classificar as estruturas condicionais;
- Aplicar as estruturas condicionais na resolução de problemas;

Sumário

- Estruturas condicionais 04
- Classificação 11
- Condições compostas 15
- Exercícios 22

Estruturas condicionais

As **estruturas condicionais** são aquelas que permitem que um programa tome decisões com base em uma ou mais **condições lógicas**, alterando o fluxo de execução do código conforme o atendimento dessas condições.

Em Python, as estruturas condicionais são classificadas em simples, composta, múltipla ou aninhada.

Vamos analisar a forma básica delas para entender como funcionam.

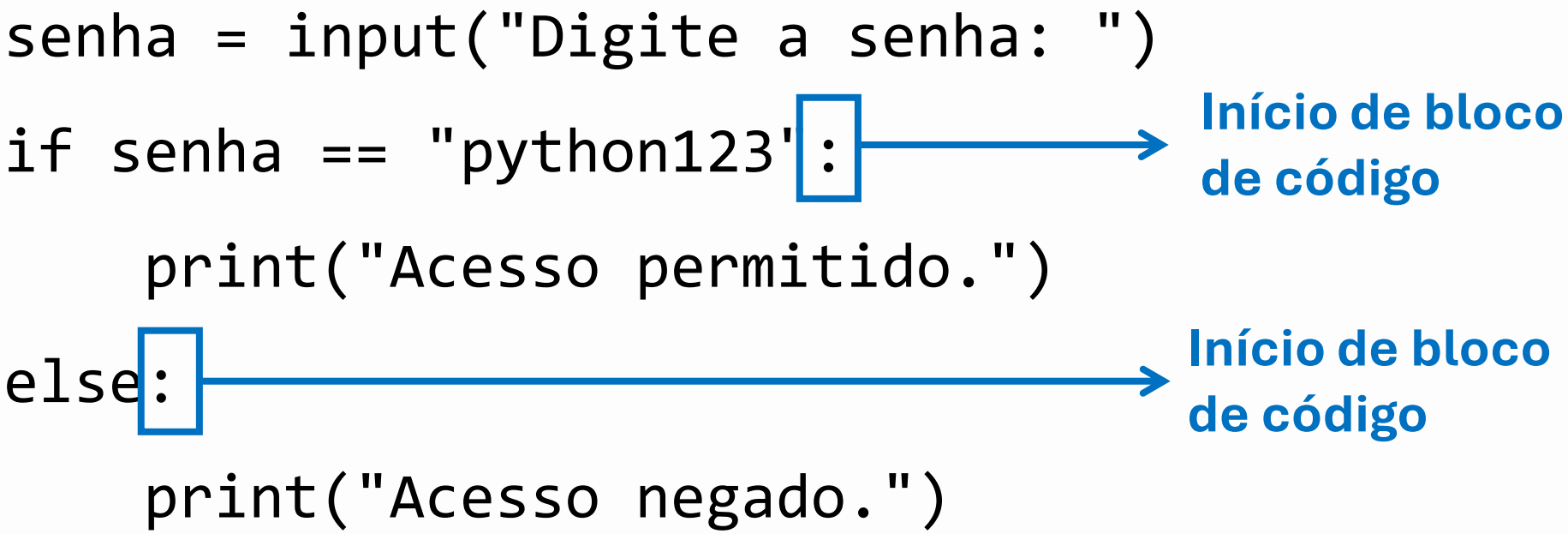
Estruturas condicionais

Vamos analisar a forma de uma estrutura condicional

```
senha = input("Digite a senha: ")
if senha == "python123":
    print("Acesso permitido.")
else:
    print("Acesso negado.")
```

Estruturas condicionais

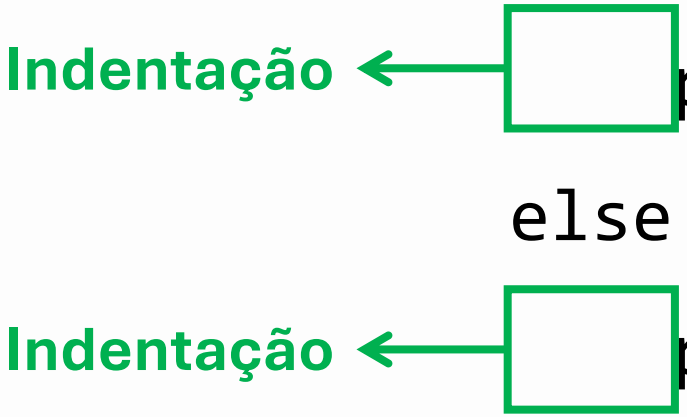
Vamos analisar a forma de uma estrutura condicional



Estruturas condicionais

Vamos analisar a forma de uma estrutura condicional

```
senha = input("Digite a senha: ")
if senha == "python123":
    print("Acesso permitido.")
else:
    print("Acesso negado.")
```



Estruturas condicionais

No exemplo vimos que:

- **No mínimo**, uma **condição** é formada por **uma variável, um operador relacional e um valor**. Em alguns casos, **ela pode ser ainda um valor booleano**;
- o sinal de "**:**" é usado para indicar que na linha abaixo se **inicia o bloco de código** que será executado se a condição que precede esse sinal for verdadeira;
- **avanços no alinhamento** são chamados **indentações** e são usados para **definir a hierarquia** das instruções. Instruções de um mesmo bloco de código compartilham a mesma indentação;

Estruturas condicionais

A partir dessas considerações podemos ainda dizer que:

- Quando um bloco de código contém outro, esse segundo bloco aparece numa nova camada de indentação. É assim que o Python entende onde começa e termina cada bloco de código.
- Dito isto, **toda linha terminada em if**, **elif**, **else**, **for**, **while**, **def**, **try**, **except**, **finally**, **with** e **class** terminarão com **:** para indicar que ali começa um bloco de código.
- **else jamais tem condição**, pois ele representa a situação em que as demais condições anteriores já foram testadas e retornaram falso!!

Classificação

Em Python, as estruturas condicionais são classificadas conforme sua **complexidade e organização**, como o uso de **ramificações simples**, **alternativas**, **múltiplas opções** ou **decisões aninhadas** dentro de outras.

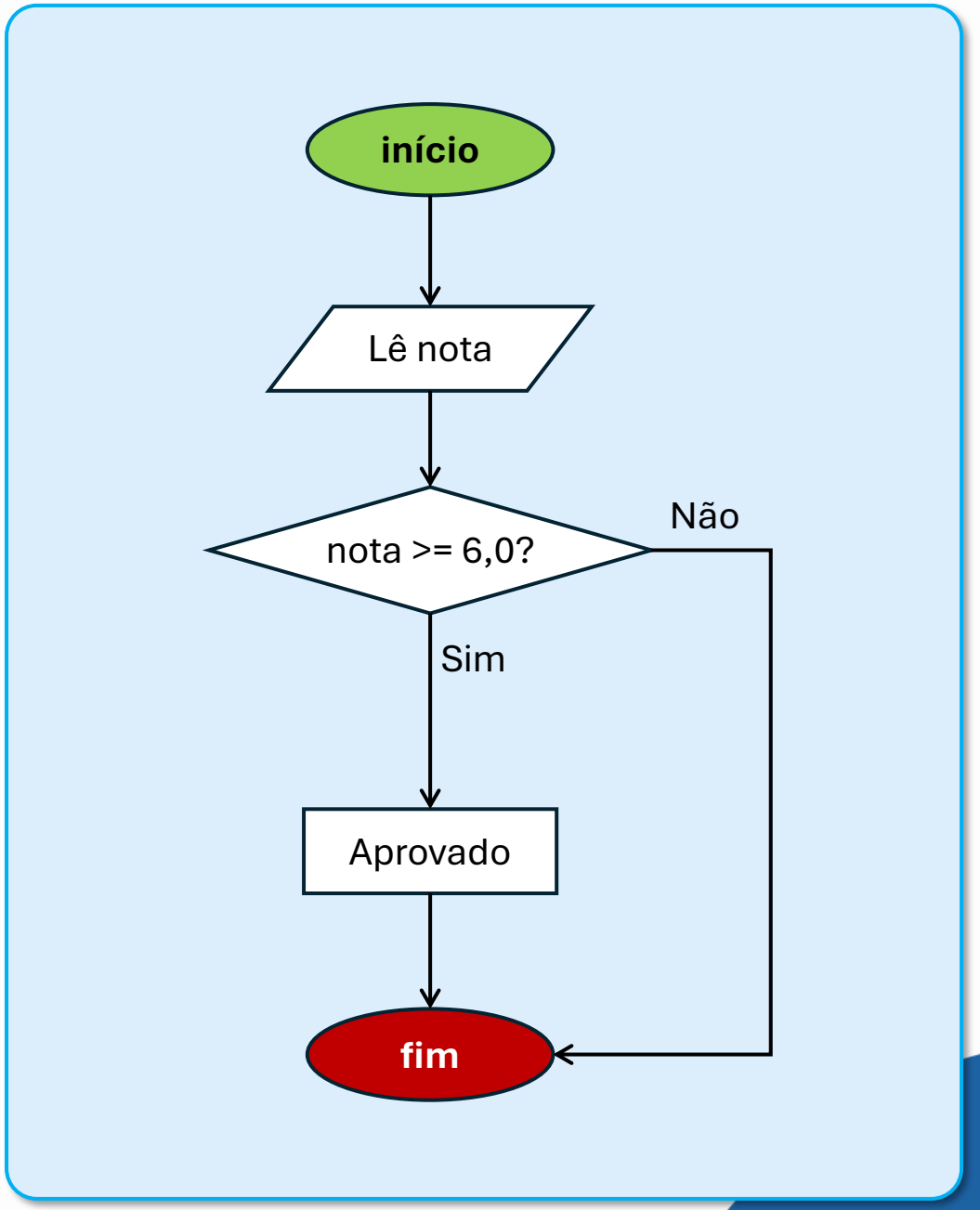
De acordo com essa classificação, uma estrutura condicional pode ser classificada em **simples**, **composta**, **múltipla** ou **aninhada**.

Vejamos um exemplo de cada uma:

Classificação

Simples: é a estrutura mais básica e usa apenas um **if** para tratar a condição quando for **verdadeira**.

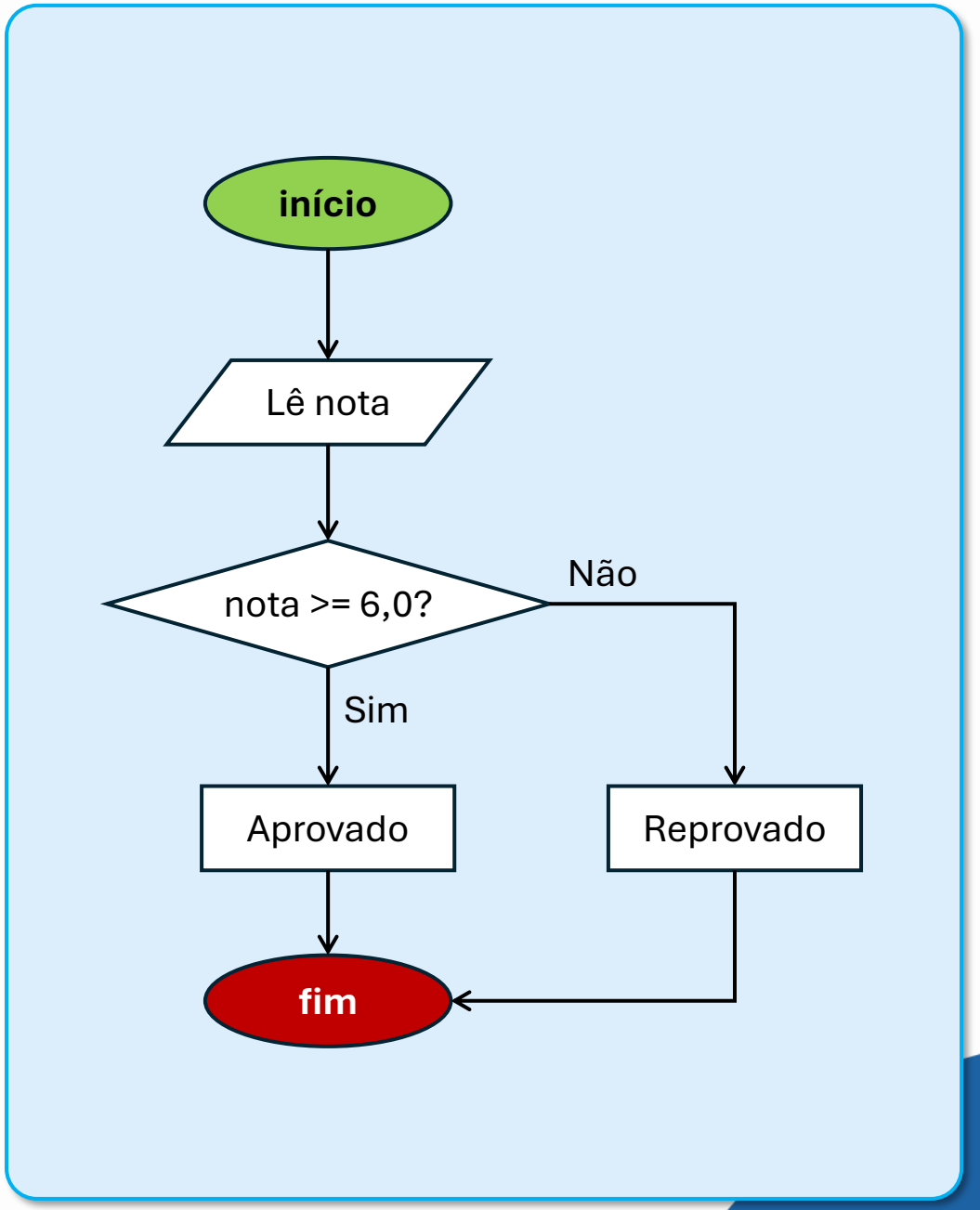
```
nota = float(input("Digite a nota: "))
if nota >= 6.0:
    print("Aprovado")
```



Classificação

Composta: usa **if** e **else** para tratar duas possibilidades: uma quando a condição é **verdadeira** e outra quando a condição é **falsa**.

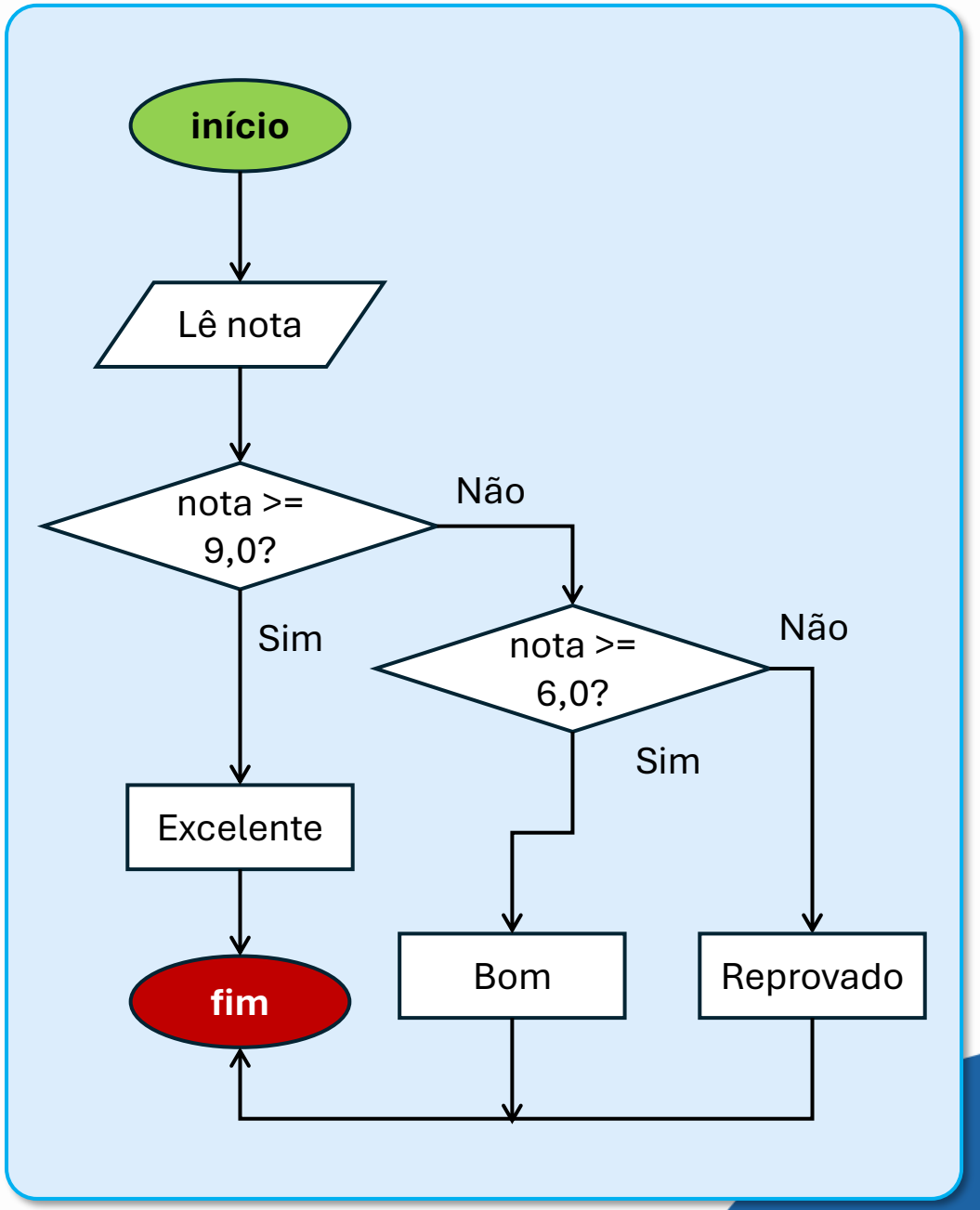
```
nota = float(input("Digite a nota: "))
if nota >= 6.0:
    print("Aprovado")
else:
    print("Reprovado")
```



Classificação

Múltipla: emprega **if**, **elif** e **else** para testar **mais de duas possibilidades**.

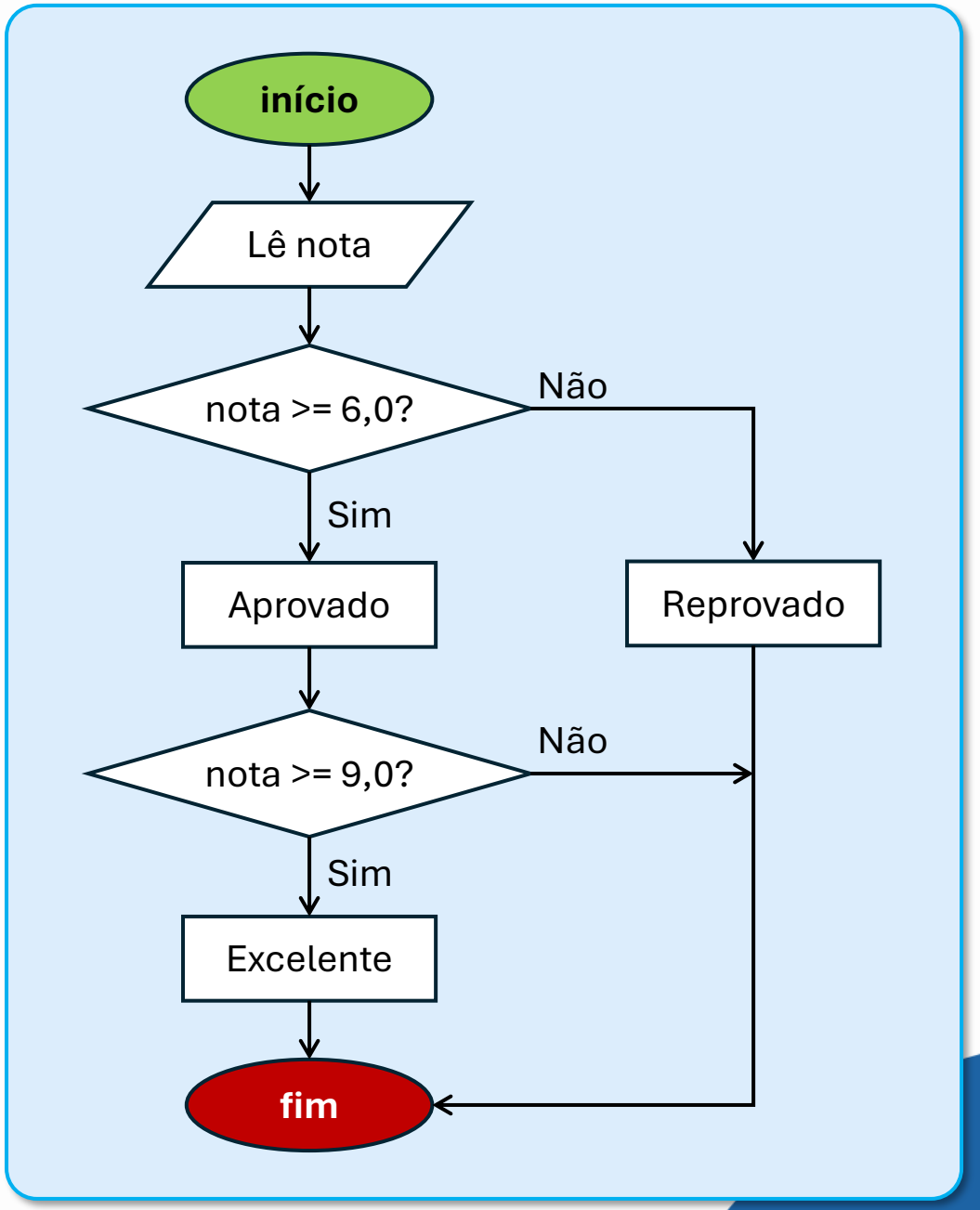
```
nota = float(input("Digite a nota: "))
if nota >= 9.0:
    print("Excelente")
elif nota >= 6.0:
    print("Bom")
else:
    print("Reprovado")
```



Classificação

Aninhada: quando **uma estrutura condicional está dentro de outra** estrutura condicional.

```
nota = float(input("Digite a nota: "))
if nota >= 6.0:
    print("Aprovado")
    if nota >= 9.0:
        print("Excelente")
else:
    print("Reprovado")
```



Condições compostas

Condições compostas podem ser construídas combinando condições simples com os operadores lógicos **and**, **or** e **not**. **Quando usamos `and`, para que o bloco de código seja executado, todas as condições simples devem ser verdadeiras**.

Por outro lado, **quando usamos `or`, para que o bloco de código seja executado, basta que pelo menos uma das condições simples seja verdadeira**.

O operador **`not`** é o operador de negação, que **inverte o valor booleano da condição ou expressão que acompanha**.

Condições compostas

Em condições compostas que utilizem operadores lógicos distintos, o interpretador avalia as condições na seguinte ordem: **not**, **and** e **or**.

O uso de parênteses força o interpretador a avalia-lo primeiro.

Vejamos alguns exemplos:

Condições compostas

Condição composta usando **and**:

```
x = 10
y = 5
if x > 5 and y < 10:
    print("x é maior que 5 e y é menor que 10")
```

As duas condições simples são verdadeiras, pois 10 é maior que 5 e 5 é menor que 10, tornando a condição composta verdadeira. O bloco de código será executado.

Condições compostas

Condição composta usando **or**:

```
x = 3
y = 15
if x < 5 or y > 20:
    print("x é menor que 5 ou y é maior que 20")
```

A primeira condição simples é verdadeira e a segunda é falsa, pois 3 é menor que 5 e 15 não é maior que 20. O bloco de código será executado porque uma delas foi verdadeira.

Condições compostas

Condição composta usando **not** e **and**:

```
x = 8
y = 12
if not x < 5 and y == 12:
    print("x não é menor que 5 e y é igual a 12")
```

- x < 5 é falso, então **not** x < 5 é verdadeiro
- y == 12 é verdadeiro

O bloco é executado!

Condições compostas

Condição composta usando **and** e **or**:

```
x = 7
y = 20
z = 3
if x > 5 and (y < 15 or z == 3):
    print("x é maior que 5 e (y é menor que 15 ou z é igual a 3)")
```

- x > 5 é verdadeiro
- y < 15 é falso e z == 3 é verdadeiro, mas por causa do or, retorna verdadeiro

O bloco é executado!

Condições compostas

Condição composta usando **not**, **and** e **or**:

```
x = 4
y = 10
z = 8
if not (x > 5 and y < 15) or z == 8:
    print("A condição complexa é verdadeira")
```

- x > 5 é falso e y < 15 é verdadeiro, então dá falso. Por causa do not, fica verdadeiro.
- z == 8 é verdadeiro.

O bloco é executado!

Exercícios

Exercício #1

Escreva um programa em Python que peça a idade de uma pessoa e, em seguida, avalie e informe se a pessoa pode dirigir. Use estrutura condicional simples.

```
idade = int(input("Digite a idade: "))
if idade >= 18:
    print("Tire sua CNH. Você já pode dirigir")
```

Exercício #2

Modifique o programa anterior para decidir se a pessoa pode ou não dirigir e informe a decisão na tela. Use estrutura condicional composta.

```
idade = int(input("Digite a idade: "))
if idade >= 18:
    print("Tire sua CNH. Você já pode dirigir")
else:
    print("Você é muito novo(a) para dirigir")
```

Exercício #3

Escreva um programa em Python que peça a idade de uma pessoa e decida se ela é criança (até 12 anos), adolescente (12 a 17 anos), adulto (18 a 59 anos) ou idoso (mais de 60 anos).

```
idade = int(input("Digite a idade: "))
if idade < 12:
    print("Você é criança.")
elif idade <= 17:
    print("Você é adolescente.")
elif idade <= 59:
    print("Você é adulto.")
else:
    print("Você é idoso.")
```

Exercício #4

Escreva um programa em Python que simule uma calculadora simples (com as operações de soma, subtração, multiplicação e divisão). O programa deverá devolver na tela o resultado da operação escolhida pelo usuário. Para a divisão, considere que o denominador será sempre o segundo número e verifique a divisão por zero. **Use estrutura condicional aninhada**.

```
a, b = float(input("Digite a: ")), float(input("Digite b: "))
operacao = input("Digite a operação desejada: +, -, * ou /: ")
ok = True  # assume que a operação será válida
if operacao == "+":
    c = a + b
elif operacao == "-":
    c = a - b
elif operacao == "*":
    c = a * b
elif operacao == "/":
    if b != 0:
        c = a / b
    else:
        print("Erro: divisão por zero.")
        ok = False
else:
    print("Operação inválida.")
    ok = False
if ok:
    print(f"O resultado de {a} {operacao} {b} é {c}")
```