Condições em Python

Nem sempre todas as linhas dos programas serão executadas. Muitas vezes, será mais interessante decidir que partes do programa devem ser executadas com base no resultado de uma condição.

A base dessas decisões consistirá em expressões lógicas que permitam representar escolhas em programas. Em Python, utilizamos estruturas condicionais para controlar o fluxo de execução.

A Estrutura if

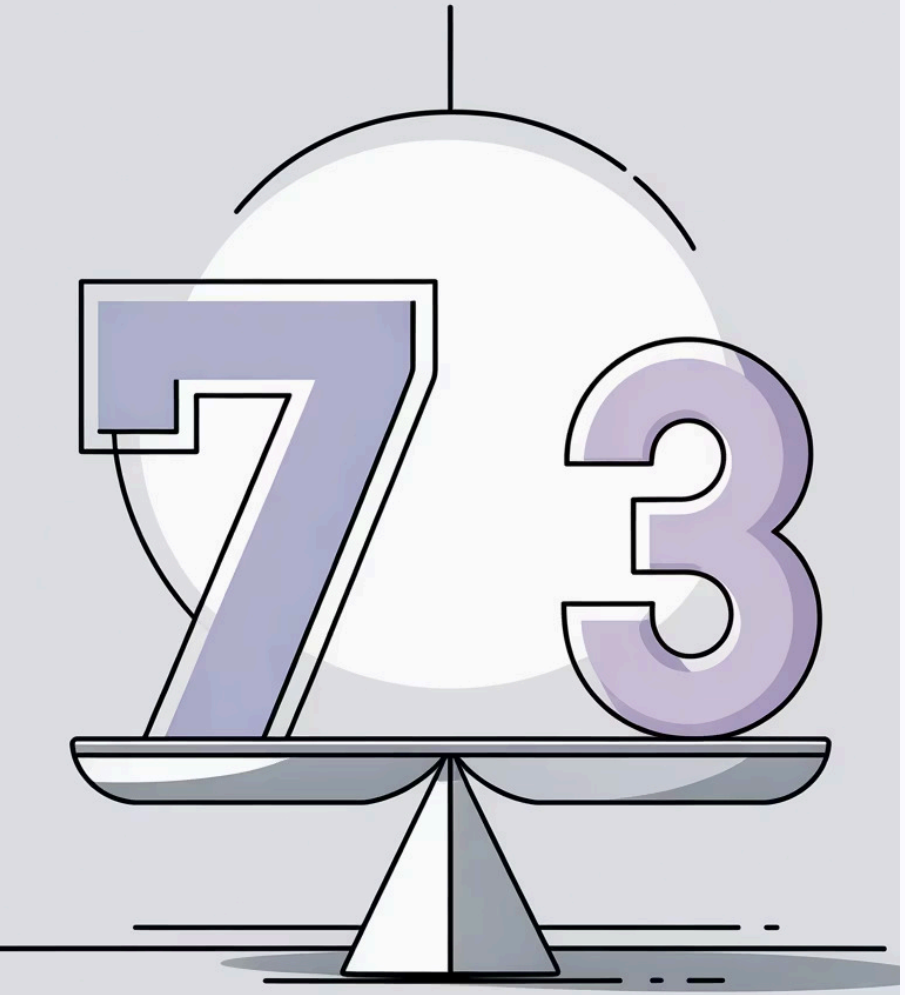
As condições servem para selecionar quando uma parte do programa deve ser ativada e quando deve ser simplesmente ignorada. Em Python, a estrutura de decisão é o **if**.

Formato Básico

```
if <condição>:
    bloco verdadeiro
```

Interpretação

O **if** é nosso "se": se a condição for verdadeira, faça alguma coisa.



Exemplo Prático: Comparando Valores

Vejamos um exemplo: ler dois valores e imprimir o maior deles.

```
a = int(input("Primeiro valor: "))
b = int(input("Segundo valor: "))

if a > b:
   print ("O primeiro número é o maior!")
if b > a:
   print ("O segundo número é o maior!")
```

A sequência de execução do programa é alterada de acordo com os valores digitados. Quando o primeiro valor é maior, executamos uma sequência; quando menor, executamos outra.

Blocos e Indentação

01

Dois Pontos (:)

As linhas com condições terminam com dois pontos, anunciando um bloco.

02

Deslocamento

Em Python, um bloco é representado deslocando o início da linha para a direita.

03

Fim do Bloco

O bloco continua até a primeira linha com deslocamento diferente.

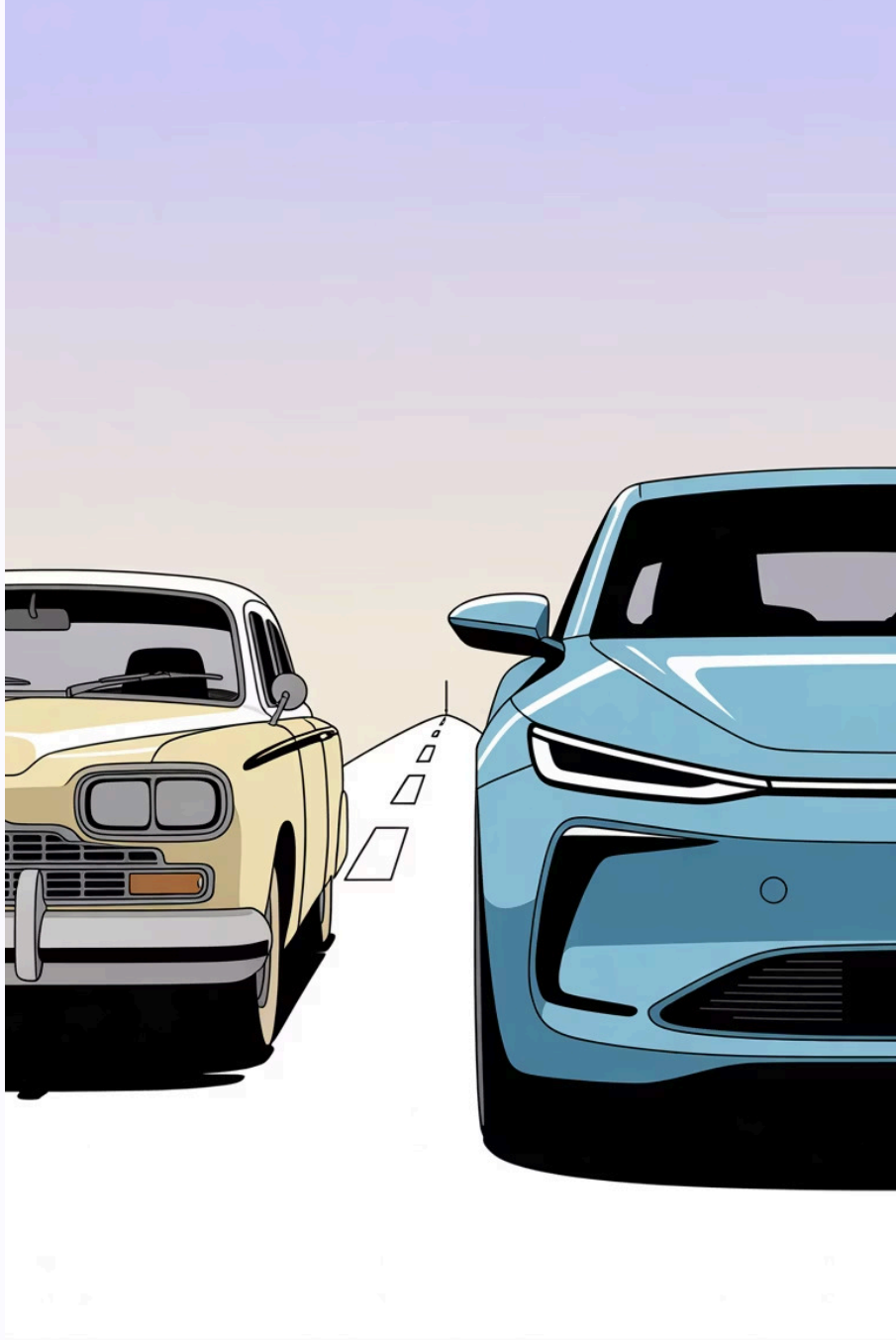
Python é uma das poucas linguagens que utiliza indentação para marcar blocos, diferente de outras que usam chaves ({ }) ou palavras como BEGIN/END.

Exemplo: Carro Novo ou Velho

Vamos criar um programa que determine se um carro é novo ou velho baseado na idade:

```
idade = int(input("Digite a idade do seu carro: "))
if idade <= 3:
  print("Seu carro é novo")
if idade > 3:
  print("Seu carro é velho")
```

A primeira condição decide se exibiremos a mensagem do carro novo para idades 0, 1, 2 e 3. A segunda condição é o inverso da primeira - não há número que torne ambas verdadeiras simultaneamente.



Cálculo de Imposto de Renda

Exemplo mais complexo: calcular imposto por faixas de salário. Para salários menores que R$ 1.000,00 não há imposto. Entre R$ 1.000,00 e R$ 3.000,00 paga-se 20%. Acima disso, 35%.

```
salário = float(input("Digite o salário: "))
base = salário
imposto = 0
if base > 3000:
    imposto = imposto + ((base - 3000) * 0.35)
    base = 3000
if base > 1000:
    imposto = imposto + ((base - 1000) * 0.20)
```

Utilizamos uma variável auxiliar **base** para não perder o valor original do salário durante os cálculos.

Rastreamento do Cálculo

500

Salário R$ 500

Imposto: R$ 0,00

1500

Salário R$ 1.500

Imposto: R$ 100,00

5000

Salário R$ 5.000

Imposto: R$ 1.100,00

O programa calcula o imposto progressivamente: primeiro a faixa de 35% para valores acima de R$ 3.000, depois a faixa de 20% para valores entre R$ 1.000 e R$ 3.000.

A Cláusula else

Quando a segunda condição é simplesmente o inverso da primeira, podemos usar **else** para simplificar os programas. O **else** especifica o que fazer quando a condição do **if** é falsa.

Antes (com dois if)

```
if idade <= 3:
   print("Novo")
if idade > 3:
   print("Velho")
```

Depois (com else)

```
if idade <= 3:
   print("Novo")
else:
   print("Velho")
```

O **else** deve estar alinhado na mesma coluna do **if** correspondente.

Estruturas Aninhadas

Muitas vezes precisamos aninhar vários **if** para obter o comportamento desejado. Aninhar significa utilizar um **if** dentro de outro.

Exemplo: calcular conta de telefone celular com três faixas de preço:

- Abaixo de 200 minutos: R$ 0,20 por minuto
- Entre 200 e 400 minutos: R$ 0,18 por minuto
- Acima de 400 minutos: R$ 0,15 por minuto

Implementação de Estruturas Aninhadas

```
minutos = int(input("Quantos minutos você utilizou:"))
if minutos < 200:
    preço = 0.20
else:
    if minutos < 400:
        preço = 0.18
    else:
        preço = 0.15
print("Você vai pagar: R$%6.2f" % (minutos * preço))
```

O **if** interno está aninhado dentro do **else**. O alinhamento do texto é muito importante em Python para definir corretamente os blocos.

Exemplo Complexo: Categorias de Produto

Programa que determina preço por categoria de produto:

Categoria 1

R$ 10,00

Categoria 2

R$ 18,00

Categoria 3

R$ 23,00

Categoria 4

R$ 26,00

Categoria 5

R$ 31,00

Com múltiplas categorias, o alinhamento se torna um problema, pois temos que deslocar à direita a cada **else**.

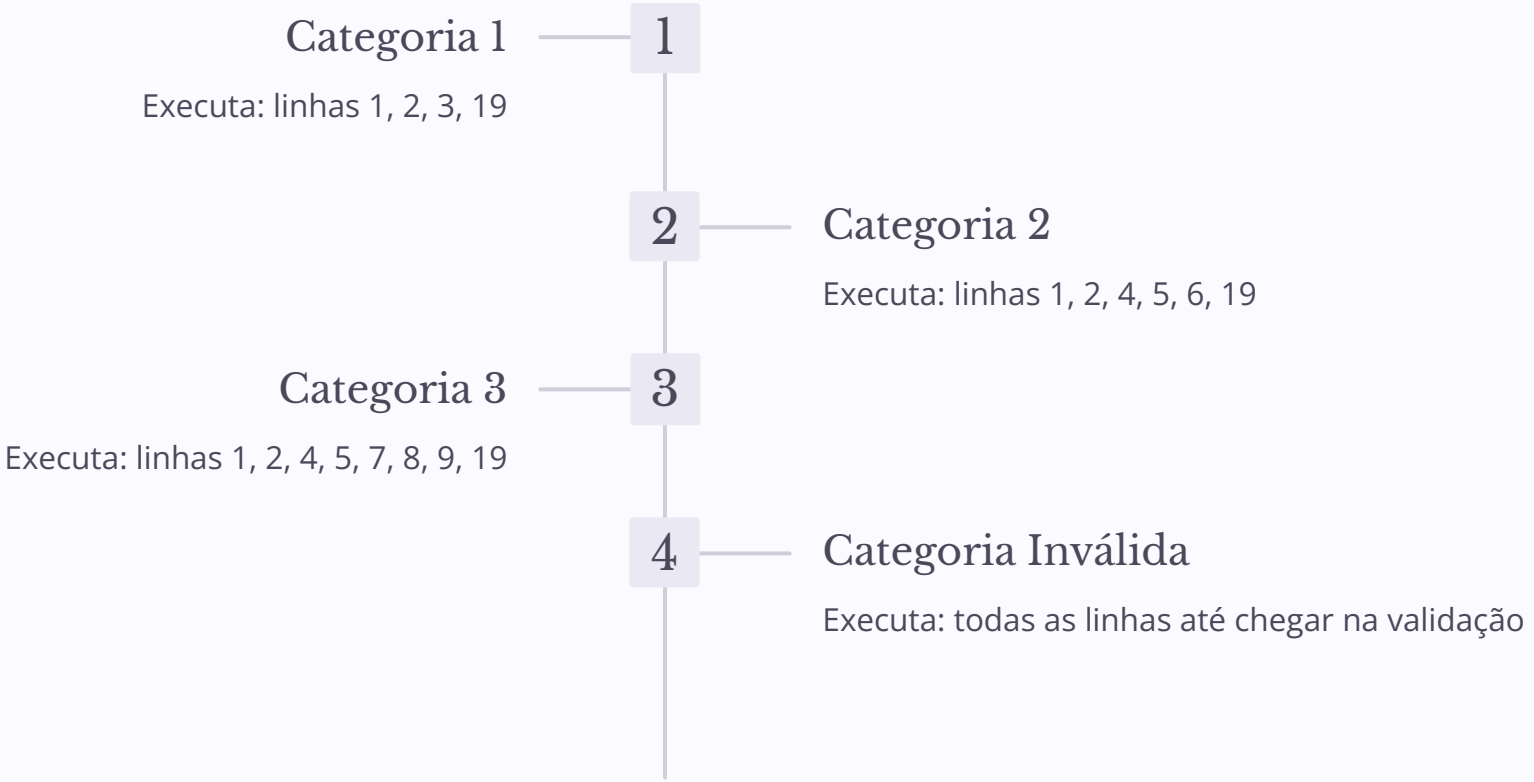
Validação de Entrada

No programa de categorias, introduzimos o conceito de validação da entrada. Se o usuário digitar um valor inválido, receberá uma mensagem de erro:

```
if categoria == 1:
    preço = 10
else:
    if categoria == 2:
        preço = 18
    else:
        # ... mais condições ...
        else:
            print("Categoria inválida!")
            preço = 0
```

A validação garante que o programa trate entradas incorretas de forma adequada.

Fluxo de Execução por Categoria



Quando lemos programas com estruturas aninhadas, devemos prestar atenção ao alinhamento para visualizar corretamente os blocos.

A Solução elif

Python apresenta uma solução elegante ao problema de múltiplos **if**s aninhados. A cláusula **elif** substitui um par **else if**, mas sem criar outro nível de estrutura.

Isso evita problemas de deslocamentos desnecessários à direita, mantendo o código mais limpo e legível.

1

Múltiplos if aninhados

Deslocamento excessivo à direita

2

elif

Estrutura linear e clara

Implementação com elif

Vamos reescrever o programa de categorias usando **elif**:

```
categoria = int(input("Digite a categoria:"))
if categoria == 1:
    preço = 10
elif categoria == 2:
    preço = 18
elif categoria == 3:
    preço = 23
elif categoria == 4:
    preço = 26
elif categoria == 5:
    preço = 31
else:
    print("Categoria inválida!")
    preço = 0
```

Muito mais limpo e fácil de ler! O **elif** mantém o mesmo nível de indentação.

Exercícios Práticos

1 Multa por Velocidade

Programa que verifica se o usuário foi multado por ultrapassar 80 km/h, cobrando R$ 5 por km acima do limite.

2 Calculadora Básica

Programa que realiza operações matemáticas básicas (+, -, *, /) entre dois números.

3 Aprovação de Empréstimo

Programa que aprova empréstimo bancário verificando se a prestação não ultrapassa 30% do salário.

Exercício: Preço da Passagem

Escreva um programa que calcule o preço da passagem baseado na distância:

Até 200 km

R$ 0,50 por km

Acima de 200 km

R$ 0,45 por km

```
distancia = float(input("Distância em km: "))
if distancia <= 200:
 preco = distancia * 0.50
else:
 preco = distancia * 0.45
print("Preço da passagem: R$%.2f" % preco)
```

Exercício: Conta de Energia

Programa que calcula o preço da energia elétrica por tipo de instalação e consumo:

Residencial

Até 500 kWh: R$ 0,40
Acima: R$ 0,65

Comercial

Até 1000 kWh: R$ 0,55
Acima: R$ 0,60

Industrial

Até 5000 kWh: R$ 0,55
Acima: R$ 0,60

Este exercício combina validação de tipo de instalação com cálculo por faixas de consumo.

Boas Práticas com Condições

Indentação Consistente

Mantenha sempre o mesmo padrão de indentação para facilitar a leitura do código.

Validação de Entrada

Sempre valide as entradas do usuário para evitar erros e comportamentos inesperados.

Use elif

Prefira elif ao invés de múltiplos if aninhados para manter o código limpo.

Lógica Clara

Estruture as condições de forma lógica e intuitiva para facilitar a manutenção.

Resumo do Capítulo

Neste capítulo aprendemos sobre estruturas condicionais em Python, fundamentais para controlar o fluxo de execução dos programas.

01

Estrutura if

Executa código apenas quando uma condição é verdadeira.

02

Cláusula else

Define o que fazer quando a condição do if é falsa.

03

Estruturas Aninhadas

Permite decisões mais complexas com múltiplos níveis.

04

Comando elif

Simplifica múltiplas condições mantendo código limpo.

Dominar essas estruturas é essencial para criar programas que tomem decisões inteligentes baseadas em diferentes situações e entradas do usuário.