Variáveis e Entrada de Dados em Python

Explore os conceitos fundamentais de variáveis, tipos de dados e entrada de informações em Python. Este capítulo amplia seu conhecimento sobre programação, apresentando operações avançadas e novos tipos de dados essenciais para o desenvolvimento de aplicações robustas.

Começar Aprendizado

Exercícios Práticos

Nomes de Variáveis em Python

Em Python, nomes de variáveis devem iniciar obrigatoriamente com uma letra, mas podem conter números e o símbolo sublinha (_). A versão 3 da linguagem permite acentos em nomes de variáveis, utilizando o conjunto de caracteres UTF-8.

Nomes Válidos

- a1 - inicia com letra
- velocidade - apenas letras
- salario_medio - com sublinha
- _b - sublinha no início

Nomes Inválidos

- 1a - inicia com número
- salário médio - contém espaços
- var-nome - hífen não permitido

Variáveis Numéricas

Variáveis numéricas armazenam números inteiros ou de ponto flutuante. Números inteiros são aqueles sem parte decimal (1, 0, -5, 550), enquanto números de ponto flutuante possuem parte decimal (1.0, 5.478, 10.478).

Números Inteiros

Valores sem parte decimal:

- 5
- -2
- 100
- 30000

Ponto Flutuante

Valores com parte decimal:

- 5.0
- 4.3
- 1.333
- -2.5

Em Python, utilizamos o ponto (.) como separador decimal, não a vírgula. Exemplo: 1.000.000 é escrito como 1000000.

Representação Binária de Números

Internamente, todos os números são representados no sistema binário (base 2). Python utiliza precisão ilimitada para inteiros, permitindo cálculos com números extremamente grandes. Para ponto flutuante, existem limites de representação.

01

Sistema Decimal

$531 = 5 \times 10^2 + 3 \times 10^1 + 1 \times 10^0$

02

Sistema Binário

$1010_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 10_{10}$

03

Precisão Python

Inteiros: ilimitada. Ponto flutuante: limitada pela mantissa e expoente

Variáveis do Tipo Lógico

Variáveis lógicas ou booleanas armazenam apenas dois valores: True (verdadeiro) ou False (falso). São fundamentais para tomada de decisões em programas e controle de fluxo.

True

Representa verdadeiro. Sempre com T maiúsculo.

```
resultado = True
```

False

Representa falso. Sempre com F maiúsculo.

```
aprovado = False
```

Operadores Relacionais

Operadores relacionais permitem comparações entre valores, retornando sempre True ou False. São essenciais para criar condições e tomar decisões em programas.

Operador	Operação	Exemplo
==	igualdade	a == b
>	maior que	b > a
<	menor que	a < b
!=	diferente	d != a
>=	maior ou igual	b >= a
<=	menor ou igual	c <= b

Operadores Lógicos

Os operadores lógicos permitem combinar expressões booleanas. Python suporta três operadores básicos: not (não), and (e), or (ou). Cada um segue regras específicas definidas por tabelas verdade.

1

not (não)

Operador unitário que inverte o valor lógico.

- not True = False
- not False = True

2

and (e)

Verdadeiro apenas quando ambos operandos são verdadeiros.

- True and True = True
- True and False = False

3

or (ou)

Falso apenas quando ambos operandos são falsos.

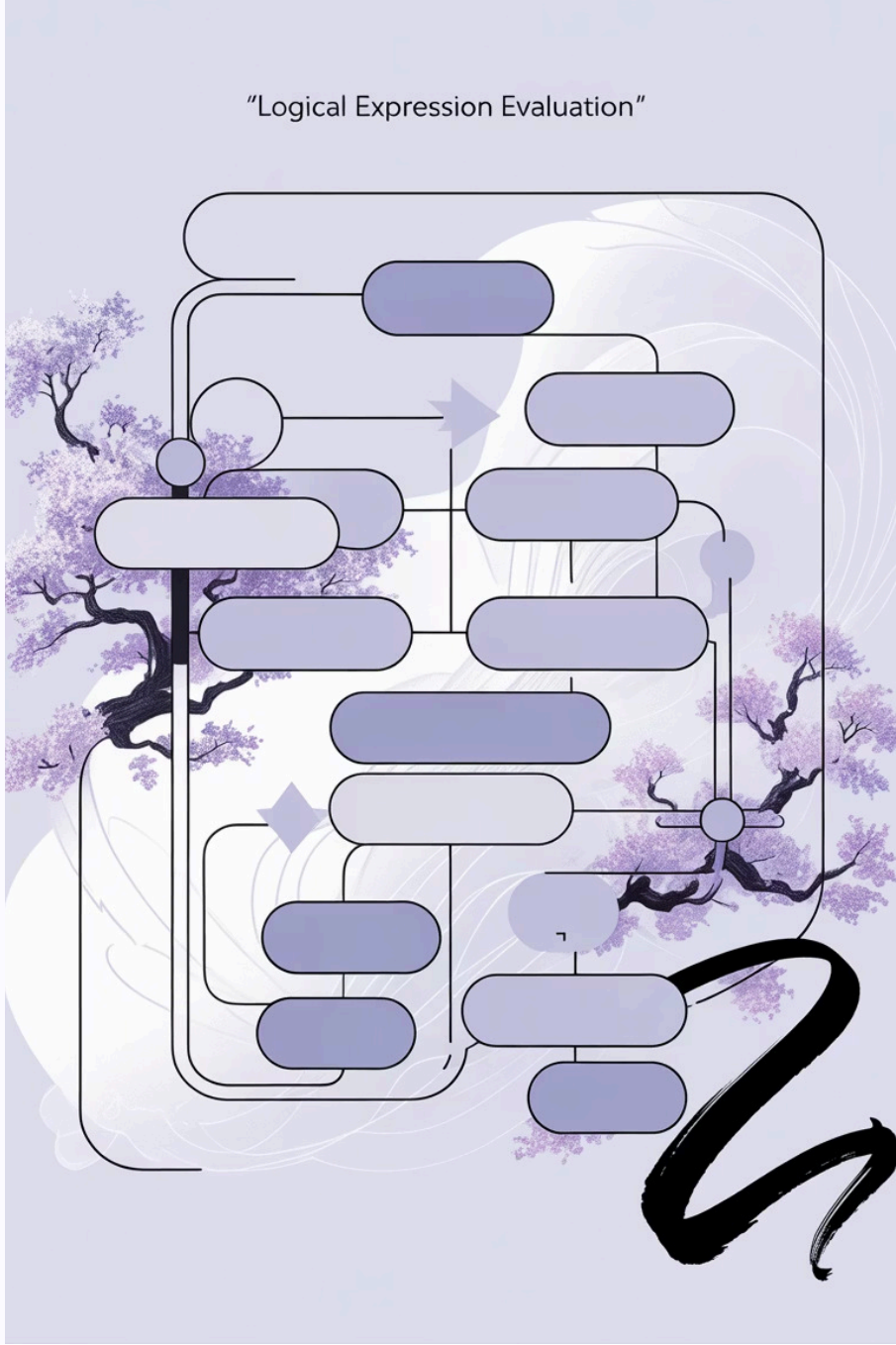
- False or False = False
- True or False = True

Expressões Lógicas Complexas

Operadores lógicos podem ser combinados em expressões complexas. A ordem de avaliação é: primeiro not, depois and, e finalmente or. Operadores relacionais são avaliados antes dos lógicos.

> Exemplo prático: salário > 1000 and idade > 18
>
> Esta expressão verifica se uma pessoa atende aos critérios para um empréstimo.

1. ### Avaliação
   Operadores relacionais primeiro
2. ### Precedência
   not → and → or
3. ### Resultado
   True ou False



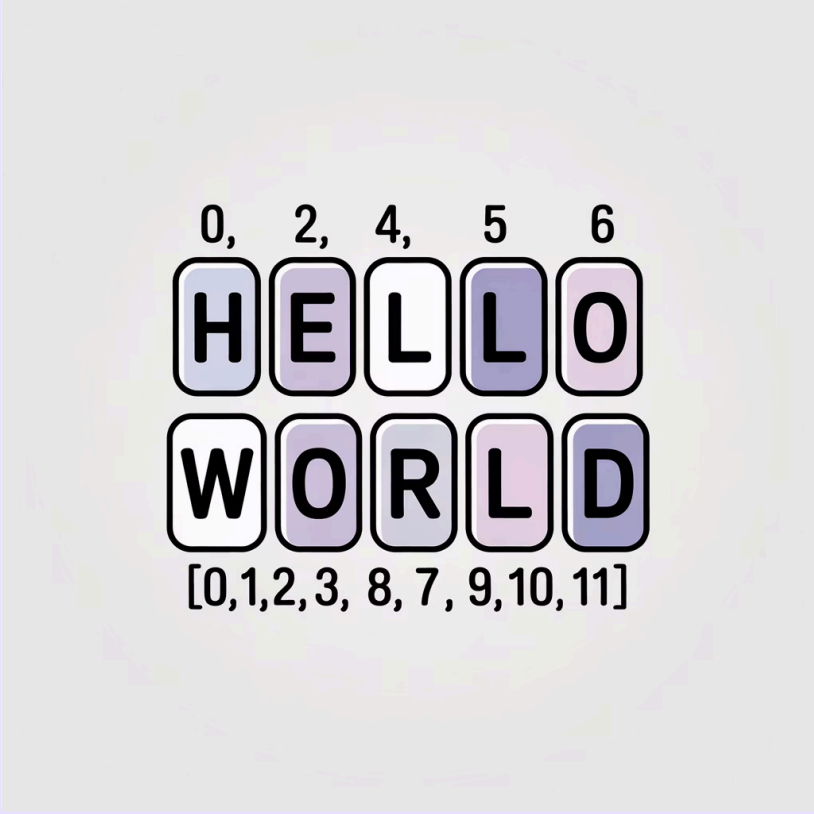
Variáveis String

Strings armazenam cadeias de caracteres como nomes e textos. São delimitadas por aspas (") e podem conter letras, números, espaços e símbolos. Cada caractere ocupa uma posição específica, começando do índice 0.

Características das Strings

- Delimitadas por aspas duplas
- Índices começam em 0
- Função len() retorna o tamanho
- Acesso por índice: string[0]

```
nome = "Python"
print(len(nome))  # 6
print(nome[0])    # P
```



⚠ Tentar acessar um índice maior que o tamanho da string resulta em erro IndexError.

Operações com Strings

Strings suportam três operações principais: concatenação (juntar strings), repetição (multiplicar strings) e fatiamento (extrair partes). Essas operações são fundamentais para manipulação de texto.

Concatenação

Une duas ou mais strings usando o operador +

```
"ABC" + "DEF" = "ABCDEF"
```

Repetição

Repete uma string usando o operador *

```
"A" * 3 = "AAA"
```

Fatiamento

Extrai partes da string usando [início:fim]

```
"PYTHON"[0:2] = "PY"
```

Composição de Strings

A composição permite inserir valores em strings usando marcadores. É mais prática que concatenação para criar mensagens complexas. Utiliza o símbolo % e marcadores específicos para cada tipo de dado.

Marcador	Tipo	Exemplo
%d	Inteiros	"Idade: %d" % 25
%s	Strings	"Nome: %s" % "João"
%f	Decimais	"Preço: %.2f" % 19.99

> Exemplo completo:
>
> ```
> "%s tem %d anos e R$%.2f" % ("João", 22, 51.34)
> ```
>
> Resultado: "João tem 22 anos e R$51.34"

Fatiamento Avançado de Strings

O fatiamento permite extrair partes específicas de strings usando a sintaxe [início:fim]. O índice de início é incluído, mas o de fim é excluído. Índices negativos contam a partir do final.

1 — Básico

s[0:2] - do início até posição 2 (exclusive)

2 — Omissão

s[:2] - do início até 2

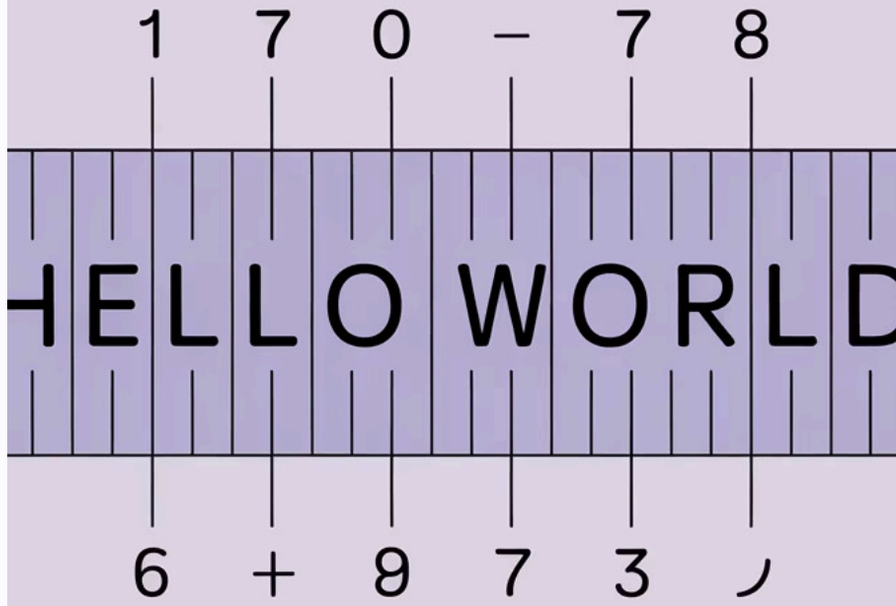
s[1:] - de 1 até o fim

3 — Negativos

s[-1] - último caractere

s[-2:-1] - penúltimo

```
s = "ABCDEFGHI"
print(s[0:2]) # AB
print(s[:2]) # AB
print(s[1:]) # BCDEFGHI
print(s[-1]) # I
```



F-Strings

As F-Strings (Formatted String Literals) são uma maneira poderosa e legível de incorporar expressões Python dentro de strings. Introduzidas no Python 3.6, elas permitem formatar strings de forma concisa, eliminando a necessidade de métodos como `.format()` ou o operador `%`, tornando o código mais claro e eficiente.

Inserindo Valores

Para incluir variáveis, expressões ou chamadas de função, basta colocá-las dentro de chaves `{}` precedidas por um `f` (de "formatted") antes das aspas da string. Isso torna o código mais legível e direto.

```
nome = "Alice"
idade = 30
mensagem = f"Olá, {nome}! Você tem {idade} anos."
print(mensagem)
# Saída: Olá, Alice! Você tem 30 anos.
```

Modificadores de Formatação

Dentro das chaves, você pode adicionar um especificador de formato após dois pontos `:`. Isso permite controlar a apresentação dos valores, como o número de casas decimais para números ou o alinhamento de texto.

```
preco = 49.998
txt = f"O preço é {preco:.2f} dólares."
print(txt)
# Saída: O preço é 49.99 dólares.
```

Você também pode formatar valores literais ou expressões diretamente nas f-strings, sem a necessidade de uma variável intermediária. Exemplo: `f"Metade de 100 é {100/2:.0f}"`.

Sequência e Tempo em Programas

Programas são executados linha por linha, e o conteúdo das variáveis pode mudar com o tempo. Cada atribuição substitui o valor anterior. Compreender essa sequência temporal é fundamental para programação.

Inicialização

dívida = 0 (valor inicial)

Primeira Compra

compra = 100
dívida = dívida + compra (0 + 100 = 100)

Segunda Compra

compra = 200
dívida = dívida + compra (100 + 200 = 300)

Resultado Final

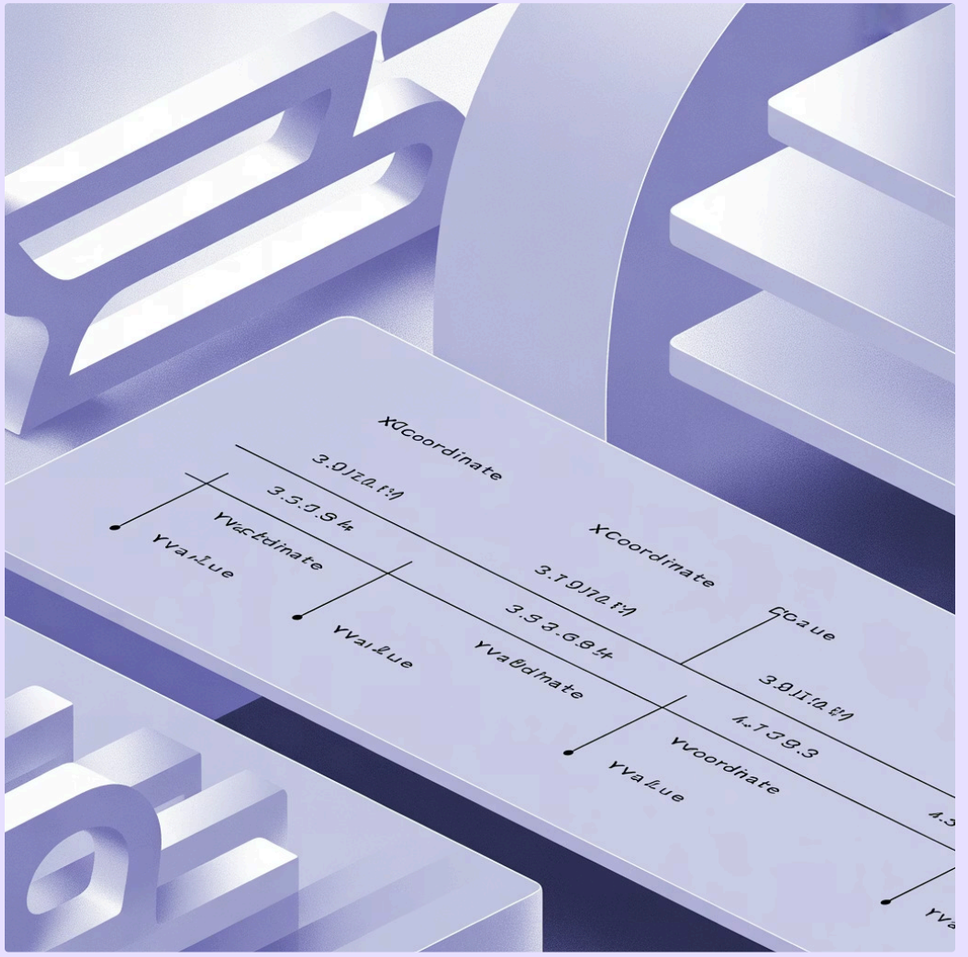
dívida = 600 (soma de todas as compras)

Rastreamento de Programas

Rastreamento é a técnica de acompanhar mudanças nas variáveis linha por linha. É essencial para entender programas e encontrar erros. Use papel e lápis para criar uma tabela com os nomes das variáveis e seus valores em cada etapa.

Como Rastrear

1. Escreva nomes das variáveis como colunas
2. Leia uma linha do programa por vez
3. Anote o valor atribuído a cada variável
4. Risque valores antigos quando mudarem
5. Continue até o final do programa



Dominar o rastreamento é essencial para programar. Se um programa não funciona, o rastreamento é a melhor ferramenta para encontrar o problema.

Entrada de Dados com input()

A função input() permite que programas recebam dados do usuário durante a execução. Ela exibe uma mensagem e aguarda o usuário digitar algo, retornando sempre uma string. Para usar números, é necessário converter o resultado.

Entrada Básica

```
nome = input("Digite seu nome: ")
```

Sempre retorna string

1

Conversão Inteira

```
idade = int(input("Idade:"))
```

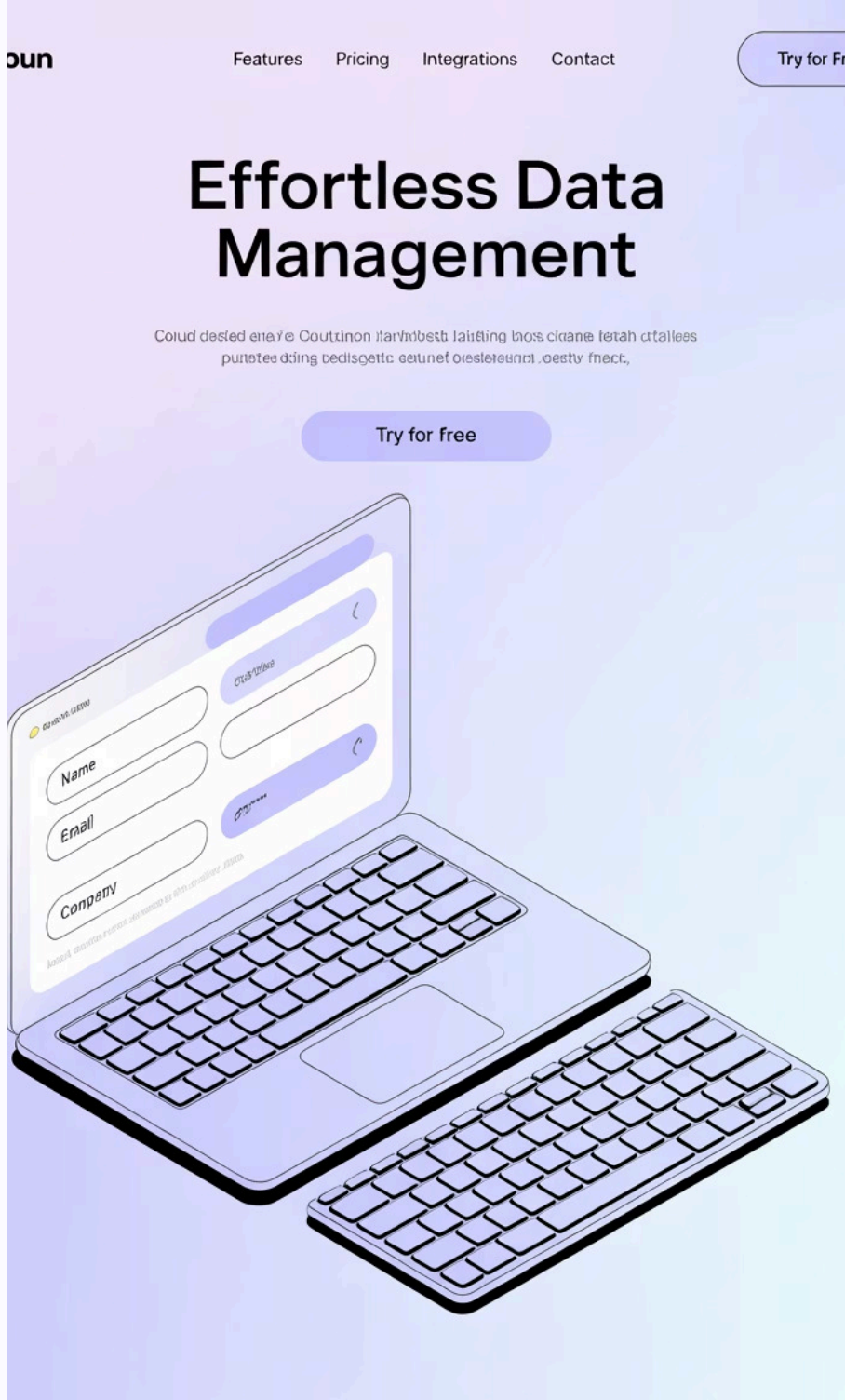
Converte para número inteiro

10

Conversão Decimal

```
valor = float(input("Valor: "))
```

Converte para ponto flutuante



Exemplo Prático: Cálculo de Bônus

Vamos criar um programa que calcula bônus por tempo de serviço, demonstrando entrada de dados, conversões e cálculos. Este exemplo mostra como combinar diferentes tipos de dados em uma aplicação prática.

```
anos = int(input("Anos de serviço: "))
valor_por_ano = float(input("Valor por ano: "))
bônus = anos * valor_por_ano
print("Bônus de R$ %5.2f" % bônus)
```

10	25	250
Anos	Reais/Ano	Bônus Total
Tempo de serviço	Valor por ano	Resultado final

Erros Comuns na Entrada de Dados

A entrada de dados é um ponto frágil nos programas. Erros ocorrem quando o usuário digita valores incompatíveis com o tipo esperado. Python gera exceções específicas para cada tipo de erro de conversão.

ValueError - Letras em Números

Digitar "abc" quando esperado um inteiro

```
ValueError: invalid literal for int() with base 10: 'abc'
```

ValueError - Vírgula no Decimal

Usar vírgula (17,4) em vez de ponto (17.4)

```
ValueError: invalid literal for float(): 17,4
```

IndexError - Índice Inválido

Acessar posição inexistente em string

```
IndexError: string index out of range
```

Exercícios Práticos - Parte 1

Pratique os conceitos aprendidos com exercícios que envolvem entrada de dados, conversões e cálculos básicos. Estes exercícios consolidam o conhecimento sobre variáveis e operações.

1 Soma de Dois Números

Peça dois números inteiros e imprima a soma

2 Conversão de Metros

Leia um valor em metros e converta para milímetros

3 Cálculo de Tempo

Leia dias, horas, minutos e segundos. Calcule total em segundos

4 Aumento Salarial

Calcule aumento de salário com valor e percentual

Exercícios Práticos - Parte 2

Continue praticando com exercícios mais elaborados que envolvem fórmulas matemáticas e cálculos do mundo real. Estes exercícios preparam você para aplicações mais complexas.

01

Desconto em Mercadoria

Calcule valor do desconto e preço final

02

Tempo de Viagem

Calcule tempo baseado em distância e velocidade

03

Conversão de Temperatura

Converta Celsius para Fahrenheit usando F = (9×C)/5 + 32

04

Aluguel de Carro

Calcule preço: R$ 60/dia + R$ 0,15/km

05

Impacto do Fumo

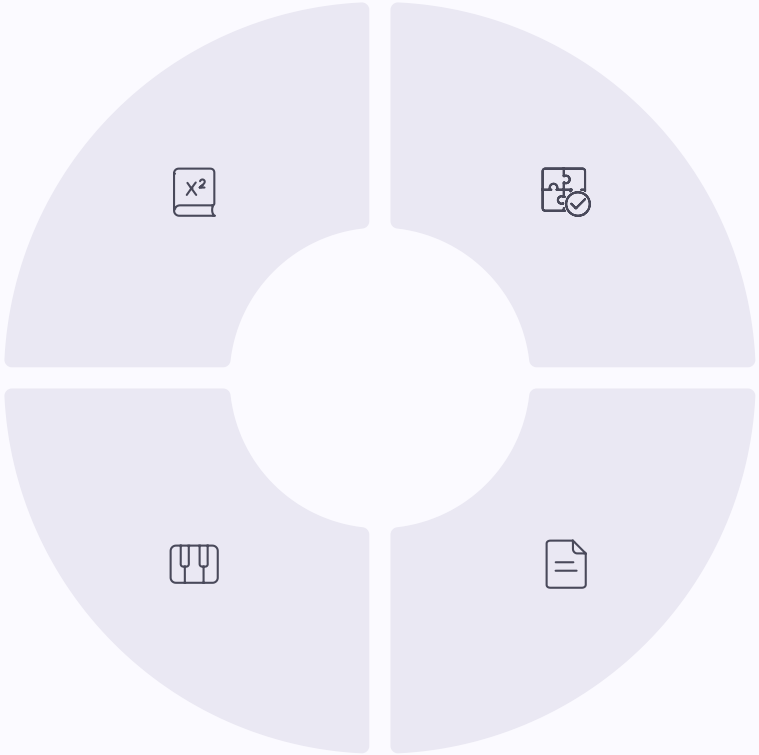
Calcule dias de vida perdidos (10 min/cigarro)

Conclusão e Próximos Passos

Você dominou os conceitos fundamentais de variáveis, tipos de dados e entrada de informações em Python. Estes conhecimentos são a base para programas mais complexos e estruturas de controle avançadas.

Variáveis

Nomes, tipos numéricos, lógicos e strings



Operadores

Relacionais, lógicos e de comparação

Entrada de Dados

Função input() e conversões de tipo

Strings

Manipulação, fatiamento e composição

Continue praticando com os exercícios propostos e prepare-se para os próximos capítulos, onde exploraremos estruturas de controle, loops e funções que tornarão seus programas ainda mais poderosos.