

Capítulo 1 - O que é programar?

Objetivo: reconhecer um programa como uma sequência de instruções que trabalha com dados. **Você precisa:** apenas acompanhar os exemplos em português. **Ao terminar:** deverá identificar entrada, processamento e saída em uma situação simples.

1.1 Uma tarefa conhecida antes do computador

Imagine uma pequena biblioteca. Uma pessoa quer descobrir quantos livros estão disponíveis em duas prateleiras. Ela conta 8 na primeira e 6 na segunda, soma os valores e escreve 14 em um papel. Já existem três partes: os dados observados, uma operação e o resultado apresentado.

Podemos descrever a tarefa assim: receber a quantidade da primeira prateleira; receber a da segunda; somar as quantidades; mostrar o total. Essa descrição ainda não está em Python. Ela é um plano de solução, escrito para que possamos conversar sobre ele.

Ao programar, transformamos esse plano em instruções de uma linguagem que uma ferramenta sabe executar. O computador segue as operações descritas, mas não descobre sozinho a intenção que esquecemos de escrever. Se dissermos para subtrair quando queríamos somar, talvez ele execute a subtração corretamente e entregue um resultado inadequado para o problema.

1.2 Programa, programação e código

Um **programa** é um conjunto organizado de instruções executáveis. **Programação** é o trabalho de planejar, escrever, testar e melhorar essas instruções. **Código-fonte** é o texto em que as escrevemos. Ler código significa tentar entender esse texto; executar significa pedir à ferramenta que realize as instruções.

Um navegador, um editor de texto e um aplicativo de mensagens são programas. Os programas que faremos aqui serão menores e funcionarão principalmente em uma janela de texto, chamada terminal. Um programa não precisa começar com botões e telas desenhadas para ser útil.

Hardware é a parte física, como teclado, memória e processador. **Software** é o conjunto de programas e instruções usado pelo equipamento. Ao pressionar uma tecla, você usa o hardware; quando um programa interpreta essa ação e mostra uma letra, existe software envolvido.

1.3 Entrada, processamento e saída

Entrada é a informação que o programa recebe. Pode vir de uma pessoa, de um arquivo, de um sensor ou de outro programa. **Processamento** é o trabalho realizado sobre essa informação. **Saída** é o resultado disponibilizado, como uma mensagem ou um arquivo.

Situação	Entrada	Processamento	Saída
Total de livros	8 e 6	Somar	14 livros
Tempo de leitura	20 minutos por dia, 4 dias	Multiplicar	80 minutos
Verificar um limite	Quantidade 9, limite 10	Comparar	Está dentro do limite

Uma saída pode virar entrada de outra etapa. O total de minutos, por exemplo, pode ser usado depois para calcular horas. Neste começo, separe uma etapa de cada vez. Assim fica mais fácil descobrir onde houve um engano.

1.4 O problema precisa de uma regra clara

“Calcule uma média” parece simples, mas faltam informações: média de quantos valores? Todos têm o mesmo peso? O que fazer com um valor ausente? Antes de escrever o programa, precisamos responder o que vale para aquele problema.

Nesta primeira regra, queremos a média de duas quantidades com o mesmo peso. Se os valores são 6 e 10, somamos 16 e dividimos por 2, obtendo 8. Com a regra escrita, já sabemos como conferir o resultado. Fazer essa conferência com um caso pequeno é parte da programação.

1.5 Errar e investigar também fazem parte

Um erro de escrita pode impedir a execução. Um erro de raciocínio pode permitir a execução e produzir a resposta errada. O trabalho de investigar a causa de um problema costuma ser chamado **depuração**. Você vai praticá-lo gradualmente, primeiro comparando o esperado com o observado.

Não é necessário acertar tudo de primeira. Um bom começo é conseguir dizer: “com estes dados, eu esperava isto; o programa mostrou aquilo”. Essa frase é mais útil do que “não funciona”, porque delimita a diferença que precisamos investigar.

Atividades de base - C01

C01-A. Uma pessoa lê 12 páginas pela manhã e 9 à noite. Identifique as entradas, o processamento e a saída de um programa que mostra o total diário. Calcule o resultado antes de consultar o gabarito.

C01-B. Um programa deveria somar 8 e 6, mas foi escrito para multiplicar. Ele mostra 48. Explique se o problema está necessariamente no computador ou nas instruções escolhidas.

Conferência de aprendizagem: você consegue explicar a diferença entre escrever e executar? Consegue apontar uma entrada e uma saída? Se sim, siga para o significado de linguagem de programação.

Capítulo 2 - O que é uma linguagem de programação?

Objetivo: entender por que usamos uma linguagem para escrever instruções. **Você precisa:** distinguir programa, entrada e saída. **Ao terminar:** deverá diferenciar linguagem, código-fonte, interpretador e editor.

2.1 Uma forma de representar instruções

Uma **linguagem de programação** é um sistema com regras para representar instruções e dados de forma que possam ser processados por ferramentas computacionais. Python, JavaScript, Java e C são exemplos. Elas possuem vocabulários e formas de escrita diferentes, mas podem expressar ideias comuns, como guardar um valor, tomar uma decisão e repetir uma tarefa.

Uma linguagem natural, como o português, permite frases abertas a interpretação. Podemos dizer “mostre um resultado bonito” sem definir o que é bonito. Para um programa, precisamos traduzir a intenção em operações verificáveis. Em nosso estudo, uma mensagem será um texto determinado e uma conta terá valores e operadores explícitos.

Usar palavras em inglês não transforma qualquer frase em código. A ferramenta espera a organização prevista pela linguagem. É por isso que espaços, aspas, parênteses e sinais merecem atenção.

2.2 Sintaxe e significado

Sintaxe é o conjunto de regras de escrita. Pense na necessidade de fechar um parêntese que foi aberto. **Semântica** é o significado das construções: o que uma instrução faz quando é válida.

Veja uma primeira amostra, apenas para observar:

```
print("Estou começando.")
```

O nome `print` identifica uma operação que pode exibir informação. Os parênteses indicam o que entregamos a essa operação. As aspas delimitam o texto. No capítulo 5, você escreverá e executará esse exemplo com calma. Ainda não é necessário memorizar a linha.

Retirar a aspa final muda a escrita de um jeito que Python não consegue interpretar como o mesmo comando. Trocar o texto entre aspas por outro, mantendo a estrutura, é uma alteração válida que muda a mensagem. Esses são dois tipos diferentes de mudança.

2.3 Como o código chega à execução

O processador trabalha com instruções de máquina. O código que escrevemos em Python é uma representação de nível mais alto, mais adequada à leitura e à organização de soluções. Ele precisa de uma implementação da linguagem para ser executado no computador.

No percurso desta apostila, usaremos o **interpretador Python**. Você entrega a ele um arquivo de código e recebe o resultado ou uma mensagem de erro. A implementação mais comum, CPython, também realiza etapas internas de compilação para uma representação intermediária. Por isso, “interpretado” não deve ser entendido como uma oposição absoluta a qualquer compilação. Para começar, importa reconhecer qual ferramenta recebe o seu arquivo e o executa.

2.4 Editor, interpretador e terminal não são a mesma coisa

O **editor** permite escrever e salvar o texto do programa. O **interpretador** executa as instruções Python. O **terminal** é uma interface textual em que você pode pedir ao sistema para executar comandos, inclusive chamar o interpretador.

Uma analogia limitada ajuda: o editor é o lugar em que você escreve a receita; o arquivo é a receita salva; a ferramenta de execução realiza as etapas descritas. Salvar a receita não significa que ela já foi realizada. Da mesma forma, salvar um arquivo `.py` não implica executar seu conteúdo.

Existem programas que reúnem editor, execução e outras ferramentas na mesma janela. São ambientes de desenvolvimento, também chamados IDEs. Mesmo quando aparecem juntos na tela, os papéis continuam diferentes.

2.5 Por que vamos usar Python

Python permite escrever muitos exemplos iniciais com poucas linhas e possui ferramentas para lidar com texto, arquivos, dados e testes. Nesta apostila, ele será o meio para aprender os fundamentos e construir programas de terminal.

É possível usar Python em automação, aplicações de servidor e análise de dados, entre outras áreas. Isso não significa que estudar os primeiros capítulos ensina todas essas especializações. A base ajuda você a escolher e compreender um caminho posterior.

O material usa Python 3. A validação dos códigos de apoio usa Python 3.12.14 no Windows, conforme o relatório que acompanha esta edição. Versões de ferramentas e menus podem mudar. As referências técnicas indicam onde conferir as orientações.

Atividades de base - C02

C02-A. Relacione três ações a seus responsáveis: escrever o texto de um programa; executar um arquivo Python; digitar um comando para iniciar a execução. Use os termos editor, interpretador e terminal.

C02-B. Uma pessoa salvou um arquivo, mas nenhuma mensagem apareceu. Por que isso, isoladamente, não prova que o arquivo está errado?

Conferência de aprendizagem: explique o que é uma linguagem sem usar apenas a frase “é Python”. Se ainda confundir ferramenta e linguagem, releia a seção 2.4 antes da preparação.

Capítulo 3 - Lógica e algoritmos antes do código

Objetivo: transformar uma tarefa em etapas claras. **Você precisa:** reconhecer entrada, processamento e saída. **Ao terminar:** deverá escrever e acompanhar um algoritmo simples em português.

3.1 O que é um algoritmo

Um **algoritmo** é uma sequência organizada de passos para resolver uma tarefa. Os passos precisam ter uma ordem e um significado suficientemente claros para o contexto. Um algoritmo pode ser discutido em português, desenhado ou expresso em uma linguagem de programação.

Vamos calcular quantas páginas uma pessoa planeja ler em cinco dias. Se ela pretende ler 8 páginas por dia, uma sequência é: obter a quantidade diária; obter o número de dias; multiplicar os valores; apresentar o total. Não começamos escolhendo uma função. Primeiro entendemos a regra.

3.2 Separar o que já sabemos do que precisamos descobrir

Escreva três perguntas: quais informações recebo? O que devo fazer com elas? O que entrego ao final? No problema da leitura, as entradas são 8 páginas por dia e 5 dias. A operação é uma multiplicação. A saída é a quantidade planejada: 40 páginas.

Uma regra deve dizer também seus limites. Aqui consideramos a mesma quantidade diária em todos os dias. Se a pessoa lê quantidades diferentes a cada dia, o problema muda. Descobrir isso antes do código evita programar uma solução para outra situação.

3.3 Pseudocódigo: escrever a ideia sem uma linguagem específica

Pseudocódigo é uma forma de registrar o raciocínio com uma escrita simplificada. Não existe uma sintaxe universal de pseudocódigo. O bloco abaixo é português organizado para leitura; não deve ser executado como Python.

INÍCIO

```
receber páginas por dia
receber quantidade de dias
calcular total = páginas por dia vezes dias
mostrar total de páginas
```

FIM

Palavras como INÍCIO e receber servem para comunicar o plano. Mais tarde faremos a tradução para Python. Separar as duas escritas ajuda a entender por que colar esse bloco em um arquivo .py não criaria automaticamente um programa válido.

3.4 Teste de mesa: acompanhar sem executar

Um **teste de mesa** é uma simulação manual dos passos. Você registra os dados e verifica o que cada etapa produz. Use um caso pequeno, em que já conhece o resultado.

Etapa	Informação conhecida depois da etapa
Receber páginas por dia	8

Etapa	Informação conhecida depois da etapa
Receber dias	5
Multiplicar	8 vezes 5 = 40
Mostrar	Total planejado: 40 páginas

Agora troque a quantidade diária por 3, mantendo cinco dias. O total esperado passa a ser 15. A regra continua a mesma; mudaram os dados. Essa distinção reaparecerá quando escrevermos programas que recebem entradas do usuário.

3.5 Três ideias que você encontrará de novo

Sequência é realizar etapas em ordem. Obter um valor antes de usá-lo em uma conta é uma sequência importante. **Decisão** é escolher um caminho conforme uma condição. Por exemplo, mostrar “meta atingida” quando o total for pelo menos 40. **Repetição** é realizar uma etapa várias vezes, como registrar a leitura de cada dia.

Essas ideias serão estudadas separadamente. Neste momento, basta reconhecê-las em português. Não usaremos uma repetição só para fazer a página parecer mais avançada. Vamos chegar a ela depois de saber acompanhar uma linha de cada vez.

3.6 Uma instrução vaga pode esconder o problema

“Organize os livros” não define uma operação única. Organizar por título, por autor ou por tamanho produz resultados diferentes. Uma versão mais clara seria: “receba dois títulos e mostre-os em ordem alfabética”. Ainda precisaríamos decidir como tratar maiúsculas e acentos quando programássemos isso.

Não é preciso resolver todos os detalhes do mundo real em um exercício inicial. É preciso dizer quais detalhes entram e quais ficam fora. Nesta trilha, os exemplos terão entradas e resultados concretos para permitir uma comparação honesta.

Atividades de base - C03

C03-A. Escreva um algoritmo em português para descobrir o total de minutos de três sessões de estudo com a mesma duração. Use 25 minutos por sessão no teste de mesa.

C03-B. Classifique as situações como sequência, decisão ou repetição: salvar antes de executar; verificar se o total passou do limite; mostrar uma mensagem para cada um de cinco dias.

Conferência de aprendizagem: você consegue resolver o problema no papel antes de pensar em Python? Se sim, está preparado para montar o ambiente de execução.