Lista de Exercícios — Lógica usando Python

Prof. João Victor Tinôco

Instruções gerais. Resolva todos os exercícios em Python. Comente seu código, valide entradas quando necessário e evite uso de funções prontas exceto quando explicitamente permitido. **Como estudar?** Faça no caderno no começo. Memorize as palavras reservadas. Deixa o processo ficar muscular. Quando você começar a programar mentalmente a solução, está na hora de ir para o teclado. Indico que você estude todos os dias, no mínimo 2h. Quanto mais dúvidas você tiver, mais horas você vai adicionar aos estudos.

Usando IA

Vá no ChatGPT e entre no **Modo Estudos**. Cole o seguinte prompt:

```
Você é meu professor particular de Programação em Python.
Quero estudar a lista de exercícios que o professor passou
(condicionais, laços, funções,
recursividade, listas, arquivos e equações matemáticas).

Sua missão é:
1. Explicar a teoria antes de cada exercício (conceito, sintaxe, exemplos práticos).
2. Resolver passo a passo o exercício, mostrando o raciocínio antes do código.
3. Escrever o código Python final bem comentado.
4. Sugerir pequenas variações para eu treinar mais.
5. Destacar erros comuns e boas práticas.
6. No final, resumir o que aprendi naquele exercício em tópicos.

Importante:
- Sempre explique como abstrair a fórmula matemática ou o problema para código.
- Use exemplos numéricos diferentes dos da lista para treinar.
- Se eu pedir, faça comparações rápidas entre Python e C++ no mesmo exercício.

Comece pelo primeiro exercício da lista e só avance para o próximo quando eu pedir.
```

Na dúvida, não esqueça: pegue seu biscoito recheado favorito, sua playlist do spotify, a cadeira mais fofinha que você tiver na sua casa, uma garrafa de Coca (de preferência Zero), papel, lápis e borracha. Com isso tu ganha o Mundo. Bons estudos!

Introdução — Fácil

1) **Número positivo, negativo ou zero.** Leia um número inteiro digitado pelo usuário. Mostre a mensagem: - "Positivo" se o número for maior que 0. - "Negativo" se o número for menor que 0. - "Zero" se o número for exatamente 0.

2) **Par ou ímpar.** Peça um número inteiro. Verifique se ele é divisível por 2. - Se for, mostre "Par". - Caso contrário, mostre "Ímpar".

3) **Maioridade.** Solicite a idade de uma pessoa (em anos). Informe se a pessoa é: - "Maior de idade" se a idade for ≥ 18. - "Menor de idade" se a idade for < 18.

4) **Maior entre dois números.** Leia dois números inteiros e informe qual deles é o maior. Se forem iguais, informe "Números iguais".

5) **Menor entre dois números.** Leia dois números inteiros e informe qual deles é o menor. Se forem iguais, informe "Números iguais".

6) **Múltiplo de 5.** Peça um número inteiro ao usuário. Informe se ele é múltiplo de 5 (ou seja, se o resto da divisão por 5 é igual a 0).

7) **Vogal ou consoante.** Receba um único caractere (letra). Informe se é vogal (a, e, i, o, u) ou consoante. Dica: considere apenas letras minúsculas ou transforme a entrada para minúscula.

8) **Validação de senha simples.** Leia uma senha digitada pelo usuário. - Se a senha for exatamente "1234", mostre "Acesso permitido". - Caso contrário, mostre "Acesso negado".

9) **Aprovado ou reprovado.** Receba uma nota (float entre 0 e 10). - Se a nota for ≥ 7, mostre "Aprovado". - Caso contrário, mostre "Reprovado".

10) **Intervalo numérico.** Peça um número inteiro. Verifique se ele está dentro do intervalo $[10, 50]$ (inclusive). - Se estiver, mostre "Dentro do intervalo". - Senão, mostre "Fora do intervalo".

Parte I — Condicionais: Intermediário

11) Classificador de notas. Leia uma nota (número real entre 0 e 10). - Se a nota for ≥ 9, classifique como "A". - Se ≥ 7, classifique como "B". - Se ≥ 5, classifique como "C". - Se ≥ 3, classifique como "D". - Caso contrário, "F". Exemplo: nota = 8.5 → "B".

12) Ano bissexto. Peça ao usuário um ano (número inteiro). Informe se ele é bissexto ou não. - Um ano é bissexto se for divisível por 400, ou se for divisível por 4 mas não por 100. Exemplo: 2000 → bissexto; 1900 → não bissexto; 2024 → bissexto.

13) Par, ímpar e múltiplo de 3. Leia um número inteiro. - Informe "PAR" se for par, ou "ÍMPAR" se for ímpar. - Se também for múltiplo de 3, adicione "M3" na saída. Exemplo: número = 9 → "ÍMPAR M3".

14) Preço com desconto. Receba o valor de uma compra. - Se o valor for > 200, aplique 10% de desconto. - Senão, aplique 5% de desconto. Mostre o valor final com duas casas decimais.

15) Triângulo válido e tipo. Peça três medidas de lados. - Verifique se podem formar um triângulo (a soma de dois lados precisa ser maior que o terceiro). - Se não formar, mostre "Inválido". - Se formar, classifique em: * "Equilátero" (3 lados iguais), * "Isósceles" (2 lados iguais), * "Escaleno" (3 lados diferentes).

16) Login simplificado. Simule um login. O sistema tem usuário fixo "admin" e senha "1234". - Se o usuário e a senha digitados estiverem corretos, mostre "Acesso permitido". - Caso contrário, mostre "Usuário ou senha incorretos". Atenção: nunca exiba a senha digitada na tela.

17) Ordem crescente de três números. Leia três números inteiros e exiba-os em ordem crescente. Restrição: não use funções prontas como `sort()`, `min()` ou `max()`. Exemplo: entrada = 8, 3, 5 → saída = 3, 5, 8.

18) Classificador de idade. Leia a idade de uma pessoa e classifique em categorias: - "Criança" se < 12 anos. - "Adolescente" se < 18 anos. - "Adulto" se < 60 anos. - "Idoso" se ≥ 60 anos.

19) Conversor de temperatura. Leia um valor de temperatura e uma flag indicando o modo de conversão: - "C2F" para converter Celsius → Fahrenheit. - "F2C" para converter Fahrenheit → Celsius. Fórmulas: - $F = C \times 1.8 + 32$ - $C = (F - 32) \div 1.8$

20) Menor e maior de três números. Leia três números inteiros e mostre: - O menor valor. - O maior valor. Restrição: não usar funções prontas como `min()` ou `max()`.

Parte II — Condicionais: Intermediário-Avançado

21) Validador de senha. Leia uma senha digitada pelo usuário e valide as seguintes regras: - Tamanho mínimo de 8 caracteres. - Deve conter pelo menos uma letra maiúscula. - Deve conter pelo menos uma letra minúscula. - Deve conter pelo menos um dígito. Caso todas as condições sejam atendidas, mostre "OK". Caso contrário, mostre mensagens específicas como "Faltou maiúscula" ou "Senha muito curta".

22) Sanitização de texto para HTML. Receba uma string de entrada. - Se contiver os caracteres "¡" ou "¿", substitua-os por "<" e ">". - Caso não haja esses caracteres, retorne o texto inalterado. Exemplo: entrada = "¡script¿" → saída = "<script>".

23) Cálculo de IMC. Solicite peso (em kg) e altura (em metros). Calcule o IMC:

$$IMC = \frac{peso}{altura^2}$$

Classifique de acordo com a OMS: - Magreza: $IMC < 18.5$ - Normal: $18.5 \leq IMC < 25$ - Sobrepeso: $25 \leq IMC < 30$ - Obesidade I: $30 \leq IMC < 35$ - Obesidade II: $35 \leq IMC < 40$ - Obesidade III: $IMC \geq 40$

24) Conversor de moeda. Leia: - Valor em dinheiro. - Taxa de conversão (float > 0). - Modo: "BRL2USD" (converter reais → dólares) ou "USD2BRL" (converter dólares → reais). Valide se a taxa é positiva e se o modo é válido. Mostre o valor convertido com 2 casas decimais.

25) Classificação de risco. Leia um número inteiro representando um score entre 0 e 100. - Se < 33: "baixo". - Se ≤ 66: "médio". - Se > 66: "alto". Exemplo: entrada = 50 → saída = "médio".

26) Validador de data. Leia três números inteiros representando dia, mês e ano. Verifique se formam uma data válida no calendário. - Considere meses com 30/31 dias. - Fevereiro tem 28 ou 29 dias dependendo de ano bissexto. Retorne "Data válida" ou "Data inválida".

27) Classificação de IPv4. Leia uma string no formato "a.b.c.d". - Verifique se existem 4 partes separadas por ponto. - Cada parte deve ser um número inteiro entre 0 e 255. - Caso seja válido, classifique o endereço como: * Classe A: 0–127 * Classe B: 128–191 * Classe C: 192–223 * Classe D: 224–239 * Classe E: 240–255

28) Calculadora de tarifa elétrica. Receba: - Horário de consumo (0 a 23 horas). - Consumo em kWh (número real). Defina uma regra de tarifas (exemplo fictício): - Horário de ponta (18–22h): R$ 0.80 por kWh. - Fora de ponta: R$ 0.50 por kWh. Calcule o valor total a ser pago.

29) Normalizador de nome. Receba um nome digitado pelo usuário, que pode ter letras em maiúsculas/minúsculas misturadas e espaços extras. Exemplo: entrada = " joÃO vIcTor " → saída = "João Victor". Regras: - Remover espaços desnecessários. - Colocar a primeira letra de cada palavra em maiúsculo e o restante em minúsculo.

30) Seleção de frete. Receba o peso de um pacote (em kg). Classifique o tipo de frete: - Se > 30: "indisponível". - Se 10–30: "transportadora". - Se ≤ 10: "expresso". - Se ≤ 2: "econômico". Exemplo: peso = 1.5 → "econômico".

Parte III — Condicionais: Avançado

31) Avaliação de expressão lógica segura. Leia três valores booleanos (True/False) e um operador lógico escolhido pelo usuário: "AND", "OR" ou "XOR". - Calcule o resultado da expressão aplicando o operador entre os três valores. - Não utilize funções perigosas como `eval()`. Exemplo: valores = True, False, True; operador = "AND" → saída = False.

32) Classificador de senha forte. Leia uma senha e calcule sua força baseada em critérios: - Pontos por comprimento (+2 pontos por caractere acima de 8). - Pontos por conter maiúsculas, minúsculas, dígitos e símbolos. - Penalize se a senha estiver em uma lista de senhas fracas comuns (ex.: "123456", "senha", "admin"). Mostre a classificação final: "Fraca", "Média" ou "Forte".

33) Regras de bloqueio de login. Dada a quantidade de tentativas falhas e o tempo decorrido em minutos: - Se houver mais de 5 tentativas falhas em até 10 minutos, mostre "BLOQUEAR". - Se houver entre 3 e 5 falhas, mostre "CAPTCHA". - Caso contrário, mostre "LIBERAR". Exemplo: 4 falhas em 7 minutos → "CAPTCHA".

34) Roteamento de requisições HTTP. Receba dois valores: método ("GET", "POST", "PUT", "DELETE") e caminho da URL. - GET → "listar recurso". - POST → "criar recurso". - PUT → "atualizar recurso". - DELETE → "remover recurso". Caso o método seja inválido, retorne "404".

35) Cálculo de imposto progressivo. Leia a renda mensal de uma pessoa e se ela é aposentada (S/N). - Calcule o imposto progressivo por faixas (ex.: até 2000 isento, até 5000 taxa de 10%, acima de 5000 taxa de 20%). - Se for aposentado e a renda for menor que 2 salários mínimos (considere salário mínimo = 1412), aplicar isenção total. Mostre o valor do imposto devido.

36) Analisador de cabeçalho HTTP. Receba o valor do cabeçalho "Content-Type". - Se não existir, retorne "400". - Se for diferente de "application/json", retorne "415". - Se for igual a "application/json", retorne "200".

37) Validador de CPF (formato). Leia uma string e verifique se ela está no formato "xxx.xxx.xxx-xx" (onde x é dígito). - Não é necessário validar os dígitos verificadores. - Apenas confirme se segue o padrão de pontos e hífen corretamente. Exemplo: entrada = "123.456.789-09" → válido.

38) Política de senha expirada. Receba a data atual e a data de expiração da senha. - Se a senha já expirou (data atual > expiração), mostre "reset imediato". - Se faltar 7 dias ou menos para expirar, mostre "alerta". - Caso contrário, mostre "ok".

39) Rate limit (limite de requisições). Leia dois valores: - Número de requisições N realizadas em 1 minuto. - Limite máximo permitido L. - Se $N \geq L$, retorne "429" (Too Many Requests). - Caso contrário, retorne "200".

40) Controle de acesso baseado em papéis (RBAC). Receba o papel do usuário ("ALUNO", "PROF" ou "ADMIN") e a ação desejada (ex.: "criar", "editar", "remover", "visualizar"). - Defina uma matriz de permissões, por exemplo: * ALUNO: apenas "visualizar". * PROF: "visualizar" e "editar". * ADMIN: todas as ações. -

Verifique se a ação é permitida para o papel informado. Exemplo: papel = "ALUNO", ação = "remover" → "Acesso negado".

Estruturas de Repetição — Fácil

1) **Contagem de 1 a 10.** Use um laço `for` para imprimir os números de 1 até 10, um por linha.

2) **Contagem regressiva.** Imprima os números de 10 até 1 em ordem decrescente usando `while`.

3) **Tabuada simples.** Leia um número inteiro n e mostre sua tabuada de 1 a 10.

4) **Somatório.** Leia um número inteiro N e calcule a soma de todos os números de 1 até N.

5) **Desenho de linha.** Peça ao usuário um número n e desenhe uma linha horizontal de n caracteres "*".

6) **Quadrado de estrelas.** Leia n e desenhe um quadrado $n \times n$ no terminal com o caractere "*".

7) **Contar vogais.** Leia uma string e conte quantas vogais (a, e, i, o, u) ela possui.

8) **Média de notas.** Leia n notas (valor definido pelo usuário) e calcule a média.

9) **Impressão de pares.** Imprima apenas os números pares de 1 até 50.

10) **Pirâmide crescente.** Leia n e imprima uma pirâmide de altura n, como no exemplo ($n = 4$):

```
*
**
***
****
```

Estruturas de Repetição — Intermediário

11) **Fatorial.** Leia um número inteiro n e calcule $n!$ usando laço.

12) **Contagem de dígitos.** Leia um número inteiro e conte quantos dígitos ele possui sem converter para string.

13) **Mínimo e máximo em sequência.** Leia vários números até que o usuário digite 0. Mostre o maior e o menor valor digitado.

14) **Desenho de triângulo invertido.** Leia n e desenhe um triângulo de estrelas invertido. Exemplo ($n = 4$):

```
****
***
**
*
```

15) Tabuleiro de xadrez. Leia n e imprima um tabuleiro $n \times n$ alternando “” e “.”. Exemplo ($n = 4$):

```
#.#.
.#.#
#.#.
.#.#
```

16) Sequência de Fibonacci. Leia n e mostre os n primeiros termos da sequência de Fibonacci.

17) Número primo. Leia um inteiro e determine se é primo (divisível apenas por 1 e ele mesmo).

18) Somar pares e ímpares. Leia vários números até o usuário digitar 0. - Some separadamente os pares e os ímpares.

19) Matriz diagonal. Leia n e imprima uma matriz $n \times n$ com “1” na diagonal principal e “0” no resto.

20) Pirâmide centralizada. Leia n e desenhe uma pirâmide centralizada. Exemplo ($n = 4$):

```
   *
  ***
 *****
*******
```

Estruturas de Repetição — Intermediário-Avançado

21) N primeiros primos. Leia n e imprima os n primeiros números primos.

22) Remover duplicatas mantendo ordem. Leia uma lista de números e produza uma nova lista sem duplicatas, preservando a ordem.

23) Contar palavras por tamanho. Leia uma frase e conte quantas palavras possuem 1, 2, 3... letras.

24) Palíndromo com laço. Leia uma palavra e verifique se é palíndromo (mesmo de trás pra frente).

25) Multiplicação de matrizes pequenas. Leia duas matrizes 2×2 e mostre o resultado da multiplicação.

26) Desenho de losango. Leia n e imprima um losango de altura $2n - 1$. Exemplo ($n = 3$):

```
  *
 ***
*****
 ***
  *
```

27) Compactação RLE. Leia uma string e aplique Run-Length Encoding (ex.: "aaabbc" → "a3b2c1").

28) Varredura de logs. Leia várias linhas contendo códigos HTTP (200, 404, 500). Conte quantos de cada tipo e mostre as porcentagens.

29) Busca BFS em grade. Leia uma grade $n \times n$ (0 = livre, 1 = obstáculo) e use um laço para percorrer em largura a partir da posição (0,0).

30) Relógio digital. Simule um relógio imprimindo todas as horas possíveis de 00:00 até 23:59.

Estruturas de Repetição — Avançado

31) Crivo de Eratóstenes. Leia n e gere todos os números primos até n usando o crivo.

32) Multiplicação de matrizes grandes. Leia duas matrizes $n \times n$ e calcule o produto usando laços aninhados.

33) Caminhos em matriz. Conte quantos caminhos existem de (0,0) até (n-1,n-1) andando apenas para direita e para baixo.

34) Fila de atendimento. Simule uma fila com operações ENQUEUE e DEQUEUE lidas do usuário. Mostre o estado final da fila.

35) Desenho de espiral. Leia n e desenhe uma matriz $n \times n$ preenchida em espiral com números de 1 até n^2.

36) Parser de CSV simples. Leia uma string com valores separados por vírgula, considerando que textos podem estar entre aspas. Separe corretamente em colunas.

37) Varredura de acessos. Leia um arquivo de log (simulado) com endereços IP e conte quantas vezes cada IP aparece. Mostre os 3 mais frequentes.

38) BFS em labirinto. Dada uma matriz representando um labirinto, encontre a distância mínima entre entrada e saída usando laço com fila.

39) Desenho de fractal simples. Imprima um padrão em forma de triângulo de Sierpinski em escala pequena (apenas lógica com loops).

40) Simulador de baralho. Gere um baralho de 52 cartas, embaralhe (Fisher-Yates) e distribua 5 cartas para cada jogador.

Funções — Fácil

1) **Função par ou ímpar.** Implemente uma função `is_even(n)` que recebe um inteiro e retorna True se for par e False caso contrário. Teste a função com diferentes valores.

2) **Área do círculo.** Crie uma função `area_circle(r)` que retorna a área de um círculo dado o raio r. Use a constante `math.pi`. Mostre resultado com 2 casas decimais.

3) **Contador de vogais.** Implemente uma função `count_vowels(s)` que recebe uma string e retorna a quantidade de vogais. Exemplo: "Python" $\rightarrow$ 1.

4) **Clamp de valores.** Escreva uma função `clamp(x, lo, hi)` que retorna lo se $x < lo$, hi se $x > hi$, ou o próprio x se estiver dentro do intervalo.

5) **Primeira letra maiúscula.** Crie uma função que recebe uma string e retorna a primeira letra em maiúscula, ou None se não houver letra.

6) **Conversor de segundos.** Faça uma função que recebe um número de segundos e retorna no formato "HH:MM:SS".

7) **Nota para conceito.** Implemente `grade(score)` que recebe uma nota (0–10) e retorna A/B/C/D/F de acordo com cortes pré-definidos.

8) **Divisão segura.** Escreva uma função que recebe dois números e retorna a divisão. Se o divisor for zero, retorne None.

9) **Soma de lista.** Faça uma função que recebe uma lista de números e retorna a soma (sem usar `sum()`).

10) **Normalização de nome.** Crie uma função que recebe um nome e retorna o nome capitalizado ("joÃO viCTor" $\rightarrow$ "João Victor").

Funções — Intermediário

11) **Validador de senha.** Implemente uma função `password_strength(pwd)` que verifica tamanho ≥ 8, maiúscula, minúscula e dígito. Retorne True ou False.

12) **Moda de uma lista.** Escreva uma função `mode(lst)` que retorna o valor mais frequente de uma lista. Se houver empate, retorne o menor valor.

13) **Slugify.** Crie uma função que recebe uma string e retorna uma versão "slug" para URL ("Título Exemplo!" $\rightarrow$ "titulo-exemplo").

14) **Classe de IP.** Implemente uma função que recebe um IPv4 e retorna sua classe (A, B, C, D, E).

15) **Média móvel.** Escreva uma função que recebe uma lista e um número k, e retorna a lista das médias móveis de tamanho k.

16) **Top K frequentes.** Crie uma função que recebe uma lista e retorna os k elementos mais frequentes.

17) Conversor de algarismos romanos. Implemente uma função que converte um número romano em inteiro.

18) Validador de data. Escreva uma função que recebe (dia, mês, ano) e retorna True se a data for válida.

19) Aplicar múltiplas regras. Faça uma função que recebe um número e uma lista de funções booleanas, retornando quantas regras foram satisfeitas.

20) Conversor BRL -¿ USD. Crie uma função que recebe um valor, uma taxa e um modo ("BRL2USD" ou "USD2BRL"). Retorne o valor convertido.

Funções — Intermediário-Avançado

21) Funções compostas. Implemente `compose(f, g)` que retorna uma nova função $h(x) = f(g(x))$.

22) Memoização simples. Escreva um decorador que armazena resultados já calculados em um dicionário.

23) Retry. Implemente uma função `retry(fn, n)` que tenta executar `fn` até n vezes.

24) Função parcial. Implemente `partial(fn, *args)` que fixa alguns argumentos de uma função.

25) Map Reduce. Escreva uma função que aplica um `map_fn` a todos os elementos e depois um `reduce_fn`.

26) Validação de schema. Implemente uma função que verifica se um dicionário segue o tipo esperado para cada chave.

27) Diff de dicionários. Escreva uma função que recebe dois dicionários e retorna chaves adicionadas, removidas e alteradas.

28) Função de logging. Crie uma função que registra chamadas de outras funções em um arquivo de log.

29) Rate limit com função. Escreva uma função que aceita um `key_fn` e limita chamadas a L vezes por minuto.

30) Sanitizador SQL. Faça uma função que recebe uma string e substitui aspas simples por duas aspas (para evitar injeção).

Funções — Avançado

31) Agendador de tarefas. Faça uma função que recebe uma lista de tarefas (hora, função) e executa apenas as que estão vencidas.

32) Executor com fallback. Implemente uma função que recebe várias funções e executa até que uma não lance exceção.

33) **Função recursiva aninhada.** Crie uma função que retorna outra função recursiva para calcular fatorial.

34) **Pipeline de funções.** Escreva uma função que recebe uma lista de funções e retorna um pipeline onde a saída de uma é a entrada da próxima.

35) **Mini-framework de validação.** Crie funções que validam email, CPF, CNPJ e combine todas em uma função principal.

36) **Executor com timeout.** Implemente uma função que executa outra função mas cancela se passar de certo tempo (simulação com contador).

37) **Versão com cache em disco.** Faça uma função que salva resultados de chamadas em arquivo JSON e reutiliza nas próximas execuções.

38) **Dispatcher de comandos.** Escreva uma função que recebe um comando string ("start", "stop", "status") e chama a função correspondente.

39) **Validador de JSON contra schema.** Crie uma função que verifica se um objeto JSON segue um schema definido (tipos e campos obrigatórios).

40) **Função polimórfica.** Implemente uma função que se comporta diferente dependendo do tipo do argumento (int, str, list).

Recursividade — Fácil

1) **Fatorial recursivo.** Implemente uma função que calcula $n!$ recursivamente.

2) **Fibonacci recursivo.** Escreva uma função que retorna o n-ésimo termo da sequência de Fibonacci.

3) **Soma recursiva de lista.** Dada uma lista, some seus elementos usando recursão.

4) **Inverter string.** Faça uma função recursiva que recebe uma string e a retorna invertida.

5) **Contagem de dígitos.** Conte recursivamente quantos dígitos tem um número inteiro positivo.

6) **Palíndromo recursivo.** Verifique recursivamente se uma string é palíndromo.

7) **Máximo recursivo.** Retorne o maior elemento de uma lista usando recursão.

8) **Converter decimal para binário.** Escreva uma função recursiva que converte um inteiro em sua representação binária (string).

9) **Contar ocorrências de caractere.** Conte quantas vezes um caractere aparece em uma string de forma recursiva.

10) **Potência recursiva.** Implemente a^b usando recursão.

Recursividade — Intermediário

11) **MDC recursivo.** Implemente o algoritmo de Euclides recursivo para calcular o máximo divisor comum.

12) **Busca binária recursiva.** Escreva uma função que busca um elemento em lista ordenada usando recursão.

13) **Merge Sort.** Implemente o algoritmo de ordenação Merge Sort de forma recursiva.

14) **Subconjuntos de lista.** Gere todos os subconjuntos possíveis de uma lista usando recursão.

15) **Permutações de string.** Gere todas as permutações possíveis de uma string usando recursão.

16) **Profundidade de árvore binária.** Calcule recursivamente a profundidade máxima de uma árvore binária.

17) **Soma dos nós de árvore.** Some todos os valores de uma árvore binária com função recursiva.

18) **Flatten de lista.** Escreva uma função que transforma lista aninhada em lista simples usando recursão.

19) **Validação de parênteses.** Use recursão para verificar se uma string de parênteses está balanceada.

20) **Caminhos em grade.** Conte quantos caminhos existem de (0,0) a (n-1,n-1) movendo apenas para baixo ou direita.

Recursividade — Intermediário-Avançado

21) **N-rainhas.** Resolva o problema das N-rainhas (conte soluções válidas).

22) **Labirinto (DFS).** Dado um labirinto representado por matriz, encontre um caminho de entrada até saída usando DFS recursivo.

23) **Subset Sum.** Verifique se existe subconjunto de uma lista que some a um valor alvo.

24) **Gerador de expressões.** Gere todas as expressões possíveis com dígitos e operadores +,-,* que resultem em um alvo.

25) **Sudoku solver.** Resolva um tabuleiro 9x9 de Sudoku usando backtracking recursivo.

26) **Busca de palavras (word search).** Verifique se uma palavra pode ser formada em uma grade de letras usando DFS recursivo.

27) **Distância de edição.** Calcule a distância de Levenshtein entre duas strings recursivamente com memoização.

28) Parsing aritmético. Construa um parser recursivo simples para expressões do tipo "3+5*2".

29) Árvore de diretórios. Liste todos os arquivos de uma árvore de diretórios representada como listas aninhadas.

30) Todos os caminhos em grafo. Liste todos os caminhos simples entre dois vértices em um grafo sem ciclos, usando recursão.

Recursividade — Avançado

31) Fractal de Sierpinski. Imprima recursivamente o triângulo de Sierpinski em escala pequena.

32) Fractal árvore. Gere recursivamente os ramos de uma árvore binária em texto.

33) Hanoi. Resolva as Torres de Hanói com n discos e mostre cada movimento.

34) Gerador de labirinto. Crie um labirinto aleatório usando divisão recursiva.

35) Compressão recursiva. Implemente uma compressão simples recursiva (tipo RLE).

36) Intérprete de linguagem mínima. Implemente um avaliador recursivo para expressões com soma, subtração e parênteses.

37) Árvore de Huffman. Construa árvore de Huffman recursivamente a partir de frequências de caracteres.

38) Resolver quebra-cabeça 8-puzzle. Use backtracking recursivo para resolver o 8-puzzle.

39) Geração de labirinto perfeito. Implemente algoritmo recursivo para gerar labirinto sem ciclos redundantes.

40) Exploração de grafo grande. Percorra recursivamente todas as conexões possíveis em um grafo não direcionado, evitando ciclos.

Listas e Arquivos — Fácil

1) Soma de lista. Leia uma lista de inteiros digitados pelo usuário e calcule a soma (sem usar `sum()`).

2) Média de lista. Calcule a média aritmética de uma lista de números.

3) Contagem de pares e ímpares. Leia uma lista de inteiros e conte quantos são pares e quantos são ímpares.

4) Maior e menor valor. Receba uma lista e mostre o maior e o menor número.

5) Remoção de duplicatas. Receba uma lista de inteiros e retorne outra lista sem elementos repetidos.

6) **Inverter lista.** Leia uma lista e mostre seus elementos na ordem inversa.

7) **Concatenação de listas.** Leia duas listas de inteiros e retorne uma nova lista com todos os elementos.

8) **Busca em lista.** Leia uma lista e um valor. Informe se o valor está presente ou não.

9) **Gravação simples em arquivo.** Peça ao usuário seu nome e idade e grave no arquivo "dados.txt".

10) **Leitura simples de arquivo.** Leia o arquivo "dados.txt" e mostre seu conteúdo na tela.

Listas e Arquivos — Intermediário

11) **Listas de nomes.** Leia vários nomes até que o usuário digite "fim". Mostre todos em ordem alfabética.

12) **Filtrar valores.** Leia uma lista de inteiros e crie uma nova lista apenas com números maiores que 10.

13) **Contagem de palavras em arquivo.** Leia um arquivo texto e conte quantas palavras ele possui.

14) **Contagem de linhas em arquivo.** Leia um arquivo texto e mostre quantas linhas existem nele.

15) **Frequência de letras.** Leia uma string e conte quantas vezes cada letra aparece, armazenando em um dicionário.

16) **Arquivo de notas.** Leia várias notas digitadas pelo usuário e grave todas no arquivo "notas.txt". Em seguida, leia o arquivo e calcule a média.

17) **Separar pares e ímpares em arquivos.** Leia uma lista de inteiros. Grave os pares em "pares.txt" e os ímpares em "impares.txt".

18) **Ordenação manual.** Leia uma lista de números e ordene manualmente (sem usar `sort()`).

19) **Concatenação de arquivos.** Leia dois arquivos texto e crie um terceiro juntando o conteúdo.

20) **Histórico de execuções.** Toda vez que o programa rodar, grave a data/hora no arquivo "log.txt".

Listas e Arquivos — Intermediário-Avançado

21) **Matriz NxN.** Crie uma lista de listas representando uma matriz $n \times n$, preenchida com valores lidos do usuário.

22) **Transposta de matriz.** Dada uma matriz (lista de listas), calcule sua transposta.

23) Arquivo CSV simples. Leia um arquivo "dados.csv" (separado por vírgulas) e carregue as linhas em uma lista de listas.

24) Busca em arquivo. Leia um arquivo de texto grande e permita que o usuário busque uma palavra, mostrando em quais linhas ela aparece.

25) Remover stopwords. Leia um arquivo texto e crie uma nova lista sem palavras comuns ("de", "a", "o", "e", ...).

26) Ordenar log por data. Leia um arquivo "log.txt" onde cada linha tem uma data. Ordene as linhas cronologicamente.

27) Contagem de acessos por IP. Leia um arquivo de log de acessos (um IP por linha). Mostre quantas vezes cada IP aparece.

28) Exportar para CSV. Leia uma lista de dicionários (usuários com nome, idade) e salve em um arquivo CSV.

29) Mesclar notas. Leia arquivos "notas1.txt" e "notas2.txt". Gere "resultado.txt" com a média de cada aluno.

30) Compactação RLE em arquivo. Leia um arquivo texto e grave outro aplicando Run-Length Encoding.

Listas e Arquivos — Avançado

31) Indexador de palavras. Leia um arquivo grande e crie um índice: para cada palavra, liste as linhas em que aparece.

32) Análise de log HTTP. Leia um arquivo de log HTTP e conte quantas requisições 200, 404 e 500 ocorreram.

33) Ranking de palavras. Leia um arquivo texto e mostre as 10 palavras mais frequentes, em ordem decrescente.

34) Banco de dados em arquivo. Implemente um mini-banco de dados: cadastre usuários (nome, email, idade) e grave em "usuarios.txt". Permita listar e buscar.

35) Cópia de arquivos binários. Faça um programa que leia um arquivo qualquer (binário) e faça uma cópia idêntica.

36) Comparador de arquivos. Leia dois arquivos e diga se são idênticos ou não.

37) Parser de JSON. Leia um arquivo "dados.json" e carregue-o em listas/dicionários (usando biblioteca `json`).

38) Backup incremental. Leia uma pasta e copie apenas os arquivos modificados depois da última execução, registrando num arquivo de controle.

39) Relatório de logs por usuário. Leia um log com formato "usuário,ação,data". Gere um relatório de quantas ações cada usuário realizou.

40) **Mini-sistema de arquivos.** Implemente um programa que permita criar, ler, editar e excluir arquivos de texto via menu interativo.

Equações Matemáticas em Python — Exercícios de Abstração

1) **Equação do segundo grau (Bhaskara).** Leia os coeficientes a, b, c e calcule as raízes da equação $ax^2 + bx + c = 0$ usando a fórmula de Bhaskara. Exiba as duas raízes reais (se existirem).

2) **Área de um triângulo (Heron).** Leia os lados a, b, c de um triângulo e calcule a área pela fórmula de Heron:

$$s = \frac{a+b+c}{2}, \quad A = \sqrt{s(s-a)(s-b)(s-c)}$$

3) **Velocidade média.** Leia distância (km) e tempo (h) e calcule a velocidade média:

$$v = \frac{d}{t}$$

4) **Energia cinética.** Leia massa (kg) e velocidade (m/s) de um objeto e calcule sua energia cinética:

$$E_c = \frac{1}{2}mv^2$$

5) **Lei de Ohm.** Leia tensão V (volts) e resistência R (ohms) e calcule a corrente I:

$$I = \frac{V}{R}$$

6) **Função exponencial.** Calcule $f(x) = e^x$ para um valor de x digitado pelo usuário, usando a biblioteca `math`.

7) **Função logarítmica.** Leia um número $n > 0$ e calcule $\log_{10}(n)$ e $\ln(n)$ (logaritmo natural).

8) **Probabilidade simples.** Leia número de casos favoráveis k e número total de casos n, calcule a probabilidade:

$$P = \frac{k}{n}$$

9) **Equação da reta.** Leia dois pontos (x_1, y_1) e (x_2, y_2) e calcule a equação da reta que passa por eles:

$$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1)$$

10) **Desvio padrão.** Leia uma lista de números e calcule o desvio padrão amostral:

$$s = \sqrt{\frac{\sum (x_i - \bar{x})^2}{n-1}}$$

onde $\bar{x}$ é a média.

Equações Matemáticas em Python — Avançado (Abstração)

1) **Entropia de Shannon (base 2).** Leia probabilidades $p_1, \ldots, p_n$ (todas > 0 e com soma ≈ 1). Calcule

$$H = -\sum_{i=1}^{n} p_i \log_2 p_i.$$

Valide a soma (tol. 10^{-6}). Mostre H com 4 casas. (*Dica:* `math.log2`.)

2) **Coeficiente binomial de duas formas.** Implemente *binomnk* de dois jeitos: (a) via fatoriais; (b) via produto multiplicativo estável

$$\binom{n}{k} = \prod_{i=1}^{k} \frac{n-k+i}{i}.$$

Compare resultados e tempo para $n = 1000, k = 5, 10, 20$. (Sem `math.comb`.)

3) **Aproximação de Stirling e erro relativo.** Para $n \in \{5, 10, 50, 100\}$, compare $n!$ com

$$S(n) = \sqrt{2\pi n}\,\left(\frac{n}{e}\right)^n.$$

Calcule o erro relativo $|n! - S(n)|/n!$ e imprima em notação científica. (*Dica:* use `math.lgamma` para
$log(n!)$ e evitar `Overflow`.)

4) **Teorema de Bayes (diagnóstico).** Dadas $P(A)$ (prevalência), $P(B|A)$ (sensibilidade), $P(B|\neg A)$ (1 - especificidade), calcule

$$P(A|B) = \frac{P(B|A)P(A)}{P(B|A)P(A) + P(B|\neg A)(1 - P(A))}.$$

Mostre $P(A|B)$ com 4 casas e interprete (texto curto).

5) **Normal: PDF e CDF aproximada.** Para média μ e desvio $\sigma > 0$, implemente a densidade

$$\varphi(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

e a CDF usando $\Phi(x) \approx \frac{1}{2}\left[1 + \operatorname{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right]$. Calcule φ e Φ para $x = \mu - 2\sigma, \mu, \mu + 2\sigma$. (*Dica:* `math.erf`.)

6) **Derivada numérica: avanço, recuo e central.** Para $f(x) = \sin x$, implemente

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}, \quad \frac{f(x) - f(x-h)}{h}, \quad \frac{f(x+h) - f(x-h)}{2h}.$$

Compare o erro com $f'(x) = \cos x$ em $x = \pi/4$ para $h = 10^{-1}, 10^{-2}, \ldots, 10^{-6}$ e identifique qual esquema é mais preciso.

7) **Integração numérica: Trapézios e Simpson.** Aproxime $\int_0^{\pi} \sin x\, dx = 2$ com N subintervalos (N par para Simpson). Imprima o erro absoluto para $N \in \{10, 100, 1000\}$.

8) Newton-Raphson com salvaguarda. Encontre raiz de $f(x) = x^3 - 2x - 5$. Implemente Newton:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \quad f'(x) = 3x^2 - 2.$$

Pare quando $|f(x_k)| < 10^{-8}$ ou em 50 iterações. Se $|f'(x_k)|$ ficar muito pequeno, troque para método da secante por 1 passo e retome. Mostre iterações.

9) Crescimento logístico (simulação discreta). Modele $P_{t+1} = P_t + rP_t\left(1 - \frac{P_t}{K}\right)\Delta t$. Dado $P_0, r, K, \Delta t$, simule até atingir $0.9K$ e reporte o tempo. Compare dois cenários (r, K) e discuta impacto.

10) Exponenciação modular rápida. Implemente *fast power* para $a^b \bmod n$ usando exponenciação binária (sem `pow(..., mod)`). Teste com $(a, b, n) = (7, 560, 561)$ e $(5, 117, 19)$. Mostre o número de multiplicações realizadas.

Extras (opcional, para quem quer ir além)

E1. Regressão linear por mínimos quadrados (sem `numpy`). Dado (x_i, y_i), calcule

$$\hat{\beta}_1 = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}, \quad \hat{\beta}_0 = \bar{y} - \hat{\beta}_1\bar{x}.$$

Imprima $\hat{\beta}_0, \hat{\beta}_1$ e R^2.

E2. Série de Fourier (onda quadrada). Aproxime $s(x) = \frac{4}{\pi}\sum_{k=1,3,5}^{N}\frac{\sin(kx)}{k}$ para $x \in [0, 2\pi]$ e $N \in \{5, 25, 125\}$. Mostre valores em uma malha de pontos (100 amostras) e comente o *overshoot* (Gibbs).

Regras de implementação

- Use apenas a biblioteca padrão (`math`, `time`, `decimal` se quiser comparar precisão).
- Formate saídas com clareza (casas decimais, notação científica quando fizer sentido).
- Explique em 1–2 linhas por exercício como validou/interpretou o resultado.