

🧩 Exercícios de POO em Python

1. Cadastro de Alunos

Contexto Expandido:

Você foi contratado por uma escola de cursos técnicos para desenvolver um sistema básico de gerenciamento de alunos. O sistema deve permitir registrar os alunos e visualizar seus dados cadastrados. Isso será usado na recepção pelos atendentes para conferência de matrículas.

- Atributos: `nome`, `matricula`, `curso`.
 - Método: `exibir_dados()`.
 - ☒ `nome` não pode ser vazio.
 - 💡 **Expansão:** Criar método `mudar_curso(novo_curso)` para simular transferências internas.
-

2. Controle de Conta Bancária

Contexto Expandido:

Você está desenvolvendo o protótipo de um app bancário que simula o funcionamento de uma conta corrente. O app precisa permitir ao usuário depositar, sacar e consultar o saldo, com as devidas restrições para evitar inconsistências.

- Atributos: `titular`, `saldo`.
- Métodos:
 - `depositar(valor)` – ☒ valor deve ser positivo.
 - `sacar(valor)` – ☒ só permitir se houver saldo suficiente.
 - `ver_saldo()`.
- 💡 **Expansão:** Adicione atributo `crédito` para permitir saques com saldo negativado.

3. Sistema de Biblioteca

Contexto Expandido:

Você está criando um sistema para uma biblioteca municipal controlar o empréstimo e devolução de livros. Cada livro deve ter seu status de disponibilidade, e o sistema deve impedir que um mesmo exemplar seja emprestado para dois leitores ao mesmo tempo.

- Atributos: `titulo`, `autor`, `ano_publicacao`, `disponivel`.
- Métodos:
 - `emprestar()` – ☒ só se disponível.
 - `devolver()` – ☒ só se já emprestado.
 - `exibir_info()`.
- 💡 **Expansão:** Crie uma classe `Usuario` e relacione livros emprestados ao usuário.

4. Simulação de Carros

Contexto Expandido:

Você está desenvolvendo uma simulação para uma escola de direção, onde diferentes modelos de carro precisam ter seus dados registrados e simular ações como acelerar e frear. A ideia é testar reações do carro sob diferentes comandos.

- Atributos: `marca`, `modelo`, `ano`, `velocidade` (0).
- Métodos:
 - `acelerar()`.
 - `frear()` – ☒ não deixar velocidade negativa.
 - `exibir_velocidade()`.
- 💡 **Expansão:** Defina limite máximo de velocidade com base na marca/modelo.

5. Controle de Estoque

Contexto Expandido:

Uma pequena loja local quer automatizar o controle de seus produtos no estoque. O sistema deverá permitir adicionar e remover produtos do estoque, além de consultar informações básicas como preço e quantidade disponível.

- Atributos: `nome`, `preco`, `quantidade`.
- Métodos:
 - `adicionar_estoque(qtd)` – ☒ `qtd` deve ser positiva.
 - `remover_estoque(qtd)` – ☒ só remover se quantidade suficiente.
 - `consultar_estoque()`.
- 💡 **Expansão:** Criar método `aplicar_desconto(percentual)` para promoções.

6. Agendamento de Consultas

Contexto Expandido:

Você foi contratado por uma clínica para criar um sistema de agendamento de consultas. O atendente deve ser capaz de registrar as consultas marcadas, visualizá-las e garantir que os dados estejam no formato correto.

- Atributos: `nome_paciente`, `nome_medico`, `data`, `horario`.
- Método: `exibir_agendamento()`.
- ☒ Use `datetime.strptime()` para validar a data e o horário.
- 💡 **Expansão:** Criar método `alterar_data_nova(data, horario)` para reagendar.

7. Simulador de Personagem de Jogo

Contexto Expandido:

Você está construindo um protótipo de um jogo de RPG onde cada personagem tem uma classe (ex: mago, guerreiro), um nível e um valor de vida. Os personagens podem subir de nível, receber dano e exibir seu status completo.

- Atributos: `nome`, `classe`, `nivel`, `vida`.
 - Métodos:
 - `subir_nivel()`.
 - `levar_dano(dano)` – ☒ não reduzir a vida abaixo de 0.
 - `exibir_status()`.
 - 💡 **Expansão:** Criar método `atacar(outro_personagem)` para duelos.
-

8. Gerenciador de Tarefas

Contexto Expandido:

Sua equipe está organizando as tarefas de um projeto usando Python como ferramenta. Cada tarefa deve ter um responsável e poder ser marcada como concluída. No futuro, você poderá agrupar e filtrar essas tarefas.


- Atributos: `descricao`, `responsavel`, `concluida`.
 - Métodos:
 - `marcar_concluida()` – ☒ só marcar se ainda não estiver.
 - `exibir_tarefa()`.
 - 💡 **Expansão:** Adicione `prioridade` e crie filtros por prioridade ou status.
-

9. Sistema de Avaliação

Contexto Expandido:

Uma escola quer informatizar o cálculo das médias dos alunos. Cada aluno pode



receber várias notas e deve ter sua média e seu status de aprovação calculados de forma automática.

- Atributos: `nome`, `notas`.
 - Métodos:
 - `adicionar_nota(nota)` –  nota entre 0 e 10.
 - `calcular_media()`.
 - `status()` – retorna "Aprovado" se média ≥ 7 .
 - 💡 **Expansão:** Criar boletim que exibe nome, notas e média com formatação.
-

10. Simulador de Animal de Estimação

Contexto Expandido:

Você está criando um bichinho virtual (pet virtual) que reage a ações como brincar, comer e descansar. O animal tem energia que varia com as atividades e um status que o tutor pode visualizar.

- Atributos: `nome`, `tipo`, `energia`.
- Métodos:
 - `brincar()` –  só se energia ≥ 20 .
 - `alimentar()` –  energia não pode ultrapassar 100.
 - `descansar()` – repõe energia para 100.
 - `status()`.
- 💡 **Expansão:** Adicione atributo `felicidade` e métodos que o afetam com base no cuidado recebido.