# Introdução à Programação em Python
## II Lista de Exercícios

Prof. Evandro C. R. Rosa
UNIVALI

Nome Completo: _____ Código de Aluno: _____

1. Crie uma classe `Produto` com os seguintes atributos:

   - `codigo`: um número inteiro único que identifica cada produto.
   - `nome`: o nome do produto.
   - `preco`: o preço do produto, que deve ser do tipo `float`.

   A classe deve incluir os seguintes métodos:

   - `__init__`: para inicializar os valores dos atributos `codigo`, `nome` e `preco`.
   - `__str__`: que retorna uma string formatada contendo o `codigo`, o `nome` e o `preco` do produto.

   Em seguida, implemente um programa que permita ao usuário inserir os dados de vários produtos (código, nome e preço). O programa deve armazenar esses dados em uma lista de objetos da classe `Produto` e, ao final, salvar essa lista em um arquivo chamado `produtos.json`. Utilize a biblioteca `json` para salvar os dados no formato JSON. Além disso, implemente tratamento de exceções para garantir que o preço inserido seja um valor numérico válido do tipo `float`.

2. Desenvolva um programa que leia os dados do arquivo `produtos.json` (gerado no exercício anterior). O programa deve converter os dados de volta para uma lista de objetos da classe `Produto` e exibir o código, nome e preço de cada produto. O programa também deve incluir tratamento de exceções para garantir que o arquivo exista antes de ser aberto. Caso o arquivo não seja encontrado ou se o conteúdo do arquivo não for um JSON válido, o programa deve capturar a exceção e exibir uma mensagem informando o erro.

3. Crie uma classe `ContaBancaria` com os seguintes atributos:

   - `titular`: o nome do titular da conta.
   - `numero`: um número único que identifica a conta.
   - `saldo`: o saldo da conta, que deve ser inicialmente 0.

   A classe deve incluir os seguintes métodos:

   - `depositar`: para realizar um depósito, adicionando um valor ao saldo.
   - `sacar`: para realizar um saque, subtraindo um valor do saldo (verificando se há saldo suficiente).
   - `exibir_saldo`: para consultar o saldo atual da conta.

Implemente um programa que permita ao usuário criar uma nova conta bancária, realizar depósitos, saques e consultar o saldo. O programa deve armazenar todas as contas em um arquivo chamado `contas.json`.

Ao iniciar o programa, ele deve tentar carregar as contas bancárias previamente salvas. Se o arquivo `contas.json` não existir, o programa deve permitir ao usuário inserir os dados da conta, como o `titular` e o `numero` da conta. Caso o arquivo já exista, o programa deve garantir que ele seja válido (ou seja, que contenha dados no formato JSON corretamente estruturados) e carregar as contas armazenadas, para que o usuário possa consultar ou atualizar as informações existentes.

Além disso, implemente tratamento de exceções para garantir que o arquivo seja manipulado corretamente e que erros, como tentativas de saque com saldo insuficiente ou valores inválidos de depósito/saque, sejam devidamente tratados.

4. Implemente um gerenciador de tarefas onde o usuário pode adicionar novas tarefas, marcar tarefas como concluídas e visualizar a lista de tarefas. Para isso, crie uma classe `Tarefa` com os seguintes atributos:

   - `titulo`: o nome da tarefa.
   - `descricao`: detalhes adicionais sobre a tarefa.
   - `status`: um valor booleano (`True` ou `False`) que indica se a tarefa foi concluída ou não.

   Em seguida, crie uma classe `GerenciadorDeTarefas`, responsável por armazenar e manipular as tarefas. A classe deve permitir adicionar tarefas, marcar tarefas como concluídas e exibir a lista de todas as tarefas.

   O programa deve salvar as tarefas em um arquivo `tarefas.json`. Ao iniciar, ele deve tentar carregar as tarefas salvas previamente. Sempre que uma tarefa for adicionada, alterada ou removida, o programa deve atualizar o arquivo para garantir que as tarefas sejam corretamente persistidas.

   Além disso, o programa deve tratar exceções adequadas ao tentar ler ou escrever no arquivo `tarefas.json`, garantindo que erros, como problemas de leitura/escrita ou dados inválidos, sejam tratados de forma apropriada.

5. Implemente um sistema de gerenciamento de biblioteca com duas classes principais: `Livro` e `Biblioteca`. A classe `Livro` deve conter os seguintes atributos:

   - `titulo`: o título do livro.
   - `autor`: o nome do autor do livro.
   - `ano_publicacao`: o ano em que o livro foi publicado.

   A classe `Biblioteca` será responsável por armazenar uma coleção de livros. Ela deve incluir os seguintes métodos:

   - `adicionar_livro`: para adicionar um livro à biblioteca.
   - `remover_livro`: para remover um livro da biblioteca, dado o seu título.
   - `buscar_livro`: para buscar um livro pelo título.
   - `listar_livros`: para listar todos os livros cadastrados na biblioteca.

O programa deve permitir que o usuário adicione e remova livros da biblioteca, e os livros devem ser armazenados em um arquivo `biblioteca.json`. Ao iniciar o programa, ele deve carregar os livros previamente salvos no arquivo e exibir todos os livros cadastrados. Caso o arquivo `biblioteca.json` não exista, o programa deve criar um novo arquivo.

O programa também deve garantir que não haja títulos duplicados na biblioteca. Caso o usuário tente adicionar um livro com um título já existente, o programa deve lançar uma exceção apropriada. Além disso, o programa deve tratar exceções para garantir que o arquivo `biblioteca.json` seja manipulado corretamente, lidando com possíveis erros de leitura ou escrita.