O desafio do Professor Peichim para quem não foi candidato é a resolução de cinco questões envolvendo Persistência de Dados.
No cabeçalho do arquivo a ser enviado coloque nome, sobrenome e DRE.
Se foi candidato a Professor Peichim e só uma pessoa da dupla foi na aula faça APENAS dois exercícios destes (e deixe explícito quais fez) e indique no cabeçalho quem foi e quem não foi.
Se a dupla foi, não deve entregar esta lista, está com a pontuação máxima atribuída já.

O trabalho deve ser enviado para o e-mail luiz.paulo.carvalho@dcc.ufrj.br
O assunto do e-mail deve ser: PROFESSOR PEICHIM 2024.2 seguido de PRODUÇÃO (se você for de produção) ou MATERIAIS (se você for de materiais)

No corpo do e-mail coloque exatamente como este formato:
INTEGRANTES: <nome e sobrenome de cada integrante da dupla, se apenas um integrante da dupla foi na aula, deixe explícito aqui quem foi e quem não foi>
USO DE IA: <o que foi utilizado de IA no geral, qualquer IA. Se nada foi utilizado, nada>

Depois dessa ficha, o código deve estar no anexo da mensagem, em arquivo .py ou .txt. Todos os códigos devem estar no mesmo arquivo.

IMPORTANTE! Todos os códigos serão verificados com sistemas/ferramentas de análise de códigos produzidos por IA generativa. Se a verificação identificar o envolvimento de IA generativa na geração de código, toda a dinâmica será zerada.
Se você usou IA generativa e copiou e colou código no seu exercício, explique no campo "USO DE IA" e coloque um comentário no respectivo local do código.
É mais importante que você faça o que sabe e com o máximo esforço do que tentar alcançar a nota 3 "roubando". Se você fizer quatro já é melhor que nada, mesmo que não alcance o 3. Se você fizer quatro e depois usar IA generativa para fazer a questão inteira que falta, aí é como se fosse fazer nada, porque o sistema/ferramenta vai detectar e eu vou te zerar.
Você ainda pode usar a IA generativa para tirar dúvidas e resolver questões simples, não utilize para geração de código automático!

Depois da verificação eu vou passar para o Pedro corrigir.

Procure sobre o módulo "os" para alguns tipos de exceção específicos.

1) Você foi designado para criar um programa Python que combine informações de dois arquivos, empregados.txt e departamentos.txt, e gere um terceiro arquivo chamado empregadosComDepartamentos.txt. O arquivo empregados.txt contém informações sobre empregados, incluindo ID, nome e e-mail, separados por vírgulas. O arquivo departamentos.txt contém informações de departamentos, incluindo ID do empregado e nome do departamento, separados por ponto e vírgula. O objetivo é criar um arquivo que contenha as informações de ID, nome, e-mail e departamento, com os IDs combinados a partir dos dois arquivos.

Instruções:
Leia e Verifique a Existência dos Arquivos:

Crie um programa Python que leia o conteúdo dos arquivos empregados.txt e departamentos.txt.
Verifique se ambos os arquivos existem e dispare uma exceção caso contrário.
Obtenha e Combine os Dados:

Para cada linha do arquivo empregados.txt, obtenha o ID, nome e e-mail.
Para cada linha do arquivo departamentos.txt, obtenha o ID do empregado e o nome do departamento.
Combine os dados dos dois arquivos para criar linhas no arquivo empregadosComDepartamentos.txt.
Criação de Novo Arquivo:

Para cada ID do empregados.txt, verifique se existe um correspondente no departamentos.txt.
Se houver correspondência, crie uma nova linha no arquivo empregadosComDepartamentos.txt com o ID, nome, e-mail e departamento.
Se não houver correspondência, insira "Departamento ausente" no lugar do departamento.
Se departamentos.txt tiver um ID de empregado que não ocorre em empregados.txt, ele não deve constar em empregadosComDepartamentos.txt.
Exibição do Resultado:

Salve o arquivo empregadosComDepartamentos.txt com as informações combinadas.
Exiba na tela o resultado final da combinação.

Exemplo de Dados:
empregados.txt:
1,Maria,maria@empresa.com
2,João,joao@empresa.com
3,Carlos,carlos@empresa.com

departamentos.txt:
1;Vendas
3;TI
4;Marketing

Arquivo final gerado:
1, Maria, maria@empresa.com, Vendas
2, João, joao@empresa.com, Departamento ausente
3, Carlos, carlos@empresa.com, TI

2) Você foi encarregado de criar um programa Python que combine e analise dados de dois arquivos: funcionarios.txt e salarios.txt. O arquivo funcionarios.txt contém informações de funcionários, incluindo ID, nome e cargo, separados por vírgulas. O arquivo salarios.txt contém informações de salários, incluindo ID do funcionário, salário mensal e bônus anual, separados por ponto e vírgula. O objetivo é ler o conteúdo desses arquivos, combinar os dados, calcular o salário total anual (considerando bônus) para cada funcionário, verificar se o salário total anual está dentro de uma faixa específica e gerar um relatório detalhado em um terceiro arquivo.

Funcionalidades Avançadas:
Criação Automática dos Arquivos de Entrada:
Crie os arquivos funcionarios.txt e salarios.txt automaticamente com dados predefinidos pelo código.
Leitura e Verificação dos Arquivos:
Crie um programa Python que leia o conteúdo dos arquivos funcionarios.txt e salarios.txt.
Verifique se ambos os arquivos existem. Dispare uma exceção caso contrário.
Combinação e Análise dos Dados:

Para cada linha do arquivo funcionarios.txt, obtenha o ID, nome e cargo.
Para cada linha do arquivo salarios.txt, obtenha o ID do funcionário, salário mensal e bônus anual.
Combine os dados dos dois arquivos com base no ID do funcionário.
Calcule o salário total anual para cada funcionário (considerando bônus).
Verificação de Faixa Salarial:

Verifique se o salário total anual está dentro da faixa especificada de 40,000 a 120,000.
Marque funcionários que têm salários fora dessa faixa.
Geração do Relatório:

Crie um terceiro arquivo chamado relatorio_funcionarios.txt.
Escreva no relatório os dados combinados de cada funcionário, incluindo ID, nome, cargo, salário mensal, bônus anual, salário total anual e se está dentro da faixa especificada.
Exibição do Relatório:

Exiba na tela o relatório gerado.
Exemplo de Dados:
funcionarios.txt:
1, Maria, Gerente
2, João, Analista
3, Carlos, Desenvolvedor
4, Ana, Designer

salarios.txt:
1;5000;10000
2;4000;5000
3;3000;2000
4;2500;3000

relatorio_funcionarios.txt:
ID: 1, Nome: Maria, Cargo: Gerente, Salário Mensal: 5000.0, Bônus Anual: 10000.0, Salário Total Anual: 70000.0, Faixa: Dentro
ID: 2, Nome: João, Cargo: Analista, Salário Mensal: 4000.0, Bônus Anual: 5000.0, Salário Total Anual: 53000.0, Faixa: Dentro
ID: 3, Nome: Carlos, Cargo: Desenvolvedor, Salário Mensal: 3000.0, Bônus Anual: 2000.0, Salário Total Anual: 38000.0, Faixa: Fora
ID: 4, Nome: Ana, Cargo: Designer, Salário Mensal: 2500.0, Bônus Anual: 3000.0, Salário Total Anual: 33000.0, Faixa: Fora

3) Você foi encarregado de criar um programa em Python para gerenciar o estoque de uma livraria. O programa deve permitir adicionar, remover, listar e salvar livros em um arquivo usando serialização com o módulo pickle. O sistema deve ser orientado a objetos (POO).

Requisitos:
Classe Livro:
Atributos: id, titulo, autor, ano, preco.
Métodos: __init__, __str__.

Classe Estoque:
Atributos: livros (uma lista de objetos Livro).

Métodos:
adicionar_livro: Adiciona um livro ao estoque.
remover_livro: Remove um livro do estoque pelo id.
listar_livros: Lista todos os livros no estoque.
salvar_estoque: Salva o estoque em um arquivo usando pickle.
carregar_estoque: Carrega o estoque de um arquivo usando pickle.

Classe Principal (ou Funções Principais):
Menu interativo para o usuário escolher entre as operações (adicionar, remover, listar, salvar e carregar).
Exceções devem ser tratadas apropriadamente.

Explicação do Código:
Classe Livro:
Define os atributos do livro (id, titulo, autor, ano, preco).
Implementa o método __str__ para uma representação amigável do livro.

Classe Estoque:
Contém uma lista de livros (livros).
Métodos para adicionar, remover, listar, salvar e carregar livros.
Usa pickle para salvar e carregar os dados do estoque em arquivos binários.

Classe Principal ou Funções Principais:
Menu interativo que permite ao usuário executar várias operações no estoque.
Exceções são tratadas para garantir que o programa não quebre ao tentar carregar um arquivo inexistente ou vazio.

Validação de Dados:
Adicione verificações para garantir que os IDs sejam únicos e que os preços sejam números válidos.
Trate exceções ao converter o preço para float.

Pesquisa de Livros:
Implemente uma funcionalidade adicional para pesquisar livros pelo título ou pelo autor.

Persistência Automática:
Adicione uma opção para salvar o estado atual do estoque automaticamente ao sair do programa.

Histórico de Transações:
Mantenha um histórico de adições e remoções de livros, e salve esse histórico em um arquivo separado usando pickle.

Código da interface de usuário (este padrão é obrigatório):

```
def menu():
    print("\nMenu:")
    print("1. Adicionar Livro")
    print("2. Remover Livro")
    print("3. Listar Livros")
    print("4. Salvar Estoque")
    print("5. Carregar Estoque")
    print("6. Sair")
    return input("Escolha uma opção: ")

def main():
    estoque = Estoque()
    while True:
        opcao = menu()
        if opcao == '1':
            id = input("ID do Livro: ")
            titulo = input("Título do Livro: ")
            autor = input("Autor do Livro: ")
            ano = input("Ano de Publicação: ")
            preco = float(input("Preço do Livro: "))
            livro = Livro(id, titulo, autor, ano, preco)
            estoque.adicionar_livro(livro)
        elif opcao == '2':
            id = input("ID do Livro a ser removido: ")
            estoque.remover_livro(id)
        elif opcao == '3':
            estoque.listar_livros()
        elif opcao == '4':
            nome_arquivo = input("Nome do arquivo para salvar o estoque: ")
            estoque.salvar_estoque(nome_arquivo)
        elif opcao == '5':
            nome_arquivo = input("Nome do arquivo para carregar o estoque: ")
            estoque.carregar_estoque(nome_arquivo)
        elif opcao == '6':
            print("Saindo do programa.")
            break
        else:
            print("Opção inválida. Tente novamente.")

if __name__ == "__main__":
    main()
```

4) Você está desenvolvendo um sistema simples de gerenciamento de contas bancárias para um banco. Cada conta pode ser do tipo "ContaCorrente" ou "ContaPoupanca". Cada conta possui um número, saldo (protegido), e deve ser capaz de realizar operações específicas de acordo com seu tipo. A ContaCorrente possui um limite de crédito especial, enquanto a ContaPoupanca possui uma taxa de rendimento.

Crie uma classe abstrata chamada ContaBancaria. Essa classe deve ter um método abstrato vazio chamado operacao (pode realizar uma operação comum como depósito, saque, etc.) e atributos comuns a todas as contas, como numero (número da conta) e o getter do saldo com decorador. Quando imprimir uma conta, deve apresentar o número e o saldo.

Implemente duas classes, ContaCorrente e ContaPoupanca, que herdam da classe ContaBancaria. Cada uma dessas classes deve implementar o método operacao() de acordo com as características da conta:

ContaCorrente: possui um limite especial que pode ser utilizado em saques além do saldo disponível. O limite deve ser um valor entre 100 e 1000, definido na instanciação. Se não estiver neste intervalo, define automaticamente como 100.
No método operacao, se o valor de entrada for positivo é um depósito, se for negativo é um saque. Se o saque estiver entre 0 e o limite, deve notificar que está aplicando o limite. Se for abaixo do limite, deve notificar saldo insuficiente.
A impressão da ContaCorrente deve indicar nome e saldo (utilizando e reaproveitando o método da classe mãe) e qual o limite especial da conta.

ContaPoupanca: possui uma taxa de rendimento que, ao final do mês, acrescenta um valor ao saldo baseado na taxa. A taxa de rendimento deve ser maior que 0,1% e menor que 0,5%. Se não estiver neste intervalo, define automaticamente como 0,1%.
O método aplicar rendimento faz com que o respectivo rendimento seja adicionado ao saldo.
A impressão da ContaPoupança deve indicar nome e saldo (utilizando e reaproveitando o método da classe mãe) e qual a taxa de rendimento.

Desenvolva funções para cadastrar, listar, excluir e alterar informações das contas. Essas funções devem interagir com o usuário para coletar as informações necessárias e persistir os dados em um arquivo usando Pickle.

Crie um menu principal que permita ao usuário realizar as seguintes operações:

Cadastrar uma nova conta (informando se é uma conta corrente ou poupança).
- Listar todas as contas cadastradas.
- Excluir uma conta pelo número.
- Alterar informações de uma conta pelo número.
- Sair do programa.

Certifique-se de tratar os erros de tentativa de excluir ou alterar uma conta que não existe.

Utilize a modularização do código, separando a lógica de persistência em funções específicas.

As informações das contas devem ser persistidas em um arquivo utilizando o módulo Pickle.

Ao cadastrar uma nova conta, permita ao usuário escolher se é uma conta corrente ou poupança, e colete as informações específicas para cada tipo (limite especial para ContaCorrente e taxa de rendimento para ContaPoupanca).

Interface:
** Menu Principal **
1. Cadastrar Conta
2. Listar Contas
3. Excluir Conta
4. Alterar Conta
5. Sair
Escolha uma opção:

<ao escolher a opção 2>
ContaCorrente: Número: 12345 - Saldo: R$ 1500.00 - Limite Especial: R$ 500.00
ContaPoupanca: Número: 67890 - Saldo: R$ 3000.00 - Taxa de Rendimento: 0.3%

<ao escolher a opção 1>
Digite o número da conta: <usuário digita>
Digite o saldo inicial da conta: <usuário digita>
A conta é do tipo ContaCorrente ou ContaPoupanca? <permite apenas essas duas opções, cuidado com letras maiúsculas e minúsculas>
Digite o limite especial (para ContaCorrente): <usuário digita>
Objetos salvos com sucesso em contas.pkl

5) Faça um código em python para contar as três palavras que mais ocorrem em um arquivo.
Deve ignorar pontuações.
Deve ignorar termos com uma letra só.
Deve ignorar números.
Deve dizer qual a palavra e quantas ocorrências.
Deve tratar as exceções de persistência de dados/arquivo devidamente.
Deve utilizar funções corretamente.
Não deve utilizar nenhum módulo, biblioteca ou função que não seja nativo do Python básico (por exemplo, math).
Deve salvar o resultado em outro arquivo.