Semana 11

Amostra 1:

Atende os requisitos?

Sim, extrai das fontes solicitadas uma tupla contendo nome do produto, valor e data de venda. O código utiliza de substrings para parsear os campos de data, total e nome do produto.

Pontos de melhoria:

Por usar substring, pode acontecer de quebrar em certos arquivos que possuam um número diferente de caracteres, outro método poderia ser utilizado para parsear esses dados.

Não possui testes nas classes Estrategia_Texto_1 e Estrategia_Texto_2 e aparentemente o autor não se preocupou em seguir o guia de estilos PEP 8. Possui muitas linhas comentadas.

Amostra 2:

Atende os requisitos?

Sim, extrai das fontes solicitadas uma tupla contendo nome do produto, valor e data de venda. O código faz o uso do replace() para substituir os espaços por ponto e vírgula e então utilizar o split() para separar os campos de cada linha.

Pontos de melhoria:

O código faz o replace() em duas quantidades específicas de espaçamento, em casos que o espaçamento for diferente, o código pode quebrar

O código possui testes unitários de cada classe e aparentemente o autor não se preocupou em seguir o guia de estilos PEP 8.

Amostra 3:

Atende os requisitos?

Sim, extrai das fontes solicitadas uma tupla contendo nome do produto, valor e data de venda. Vemos várias verificações no código para ignorar o cabeçalho. O código faz o uso do split() para separar os campos de cada linha, então utiliza o strip() para remover os espaços de cada valor.

Pontos de melhoria:

O código faz o split() por espaçamento, em casos que o espaçamento for diferente, o código pode quebrar

Não possui testes nas classes Estrategia_Texto_1 e Estrategia_Texto_2 e aparentemente o autor não se preocupou em seguir o guia de estilos PEP 8.

Semana 12

Amostra 1:

Atende os requisitos?

Sim, foi implementada todas as classes de características e produto, assim como solicitado no exercício.

Pontos de melhoria:

Ter realizado testes nas classes produto e características, pois o código passou somente em 2 de 10 testes unitários. O código também possui alguns erros do guia de estilos PEP8.

Amostra 2:

Atende os requisitos?

Sim, foi implementada todas as classes de características e produto, assim como solicitado no exercício.

O código apresenta vários testes e passa em todos os 35 testes.

Amostra 3:

Atende os requisitos?

Não, nem todas as classes foram instanciadas e implementadas

O código está de acordo com o guia de estilos PEP8 e todos os testes passam, porém está incompleto, não atendendo os requisitos enunciados.

Semana 13

Amostra 1:

Atende os requisitos?

Sim, o código cria 4 relatórios, sendo dois CSV e dois TXT com base nos registros dois bancos de dados SQLite conforme enunciado no desafio, somente com data e valor.

O autor replicou a classe Relatorio_TXT para Relatorio_CSV

Pontos de melhoria:

Ao invés de replicar a classe Relatorio_TXT, o formato CSV/TXT poderia ter passado por argumento, apenas modificando a classe já existente.

Não foram realizados testes nas classes de Relatórios, constam somente dois erros do guia de estilo PEP8.

Amostra 2:

Atende os requisitos?

Sim, o código cria 4 relatórios, sendo dois CSV e dois TXT com base nos registros dois bancos de dados SQLite conforme enunciado no desafio, somente com data e valor.

O autor criou uma nova classe para relatórios CSV, porém utilizando o método DictWriter() para gravar os registros no arquivo CSV

Pontos de melhoria:

Ao invés de criar a classe Relatorio_CSV, o formato CSV/TXT poderia ter passado por argumento, apenas modificando a classe já existente.

Não foram realizados testes nas classes e não passou nos testes já existentes. Aparentemente o autor não se preocupou em seguir o guia de estilos PEP8.

Amostra 3:

Atende os requisitos?

Não, o código apresenta diversos erros ao ser executado, o autor não instanciou a classe Relatorios_CSV corretamente e não importou a biblioteca CSV no arquivo Relatorios.py.

Pontos de melhoria:

Mesmo após realizar as correções que faltavam, o código não passou em todos os testes criados. Também consta vários erros do guia de estilo PEP8.

Semana 14

Amostra 1:

Atende os requisitos?

Sim, foi implementado as classes github e instagram, com a sessão UploadCode(), como no enunciado.

Pontos de melhoria:

Em sessões.py, há muitas funções com retornos duplicados, deixando o código repetitivo.

Possui teste para todas as classes, porém apresenta vários erros do padrão de estilos PEP8.

Amostra 2:

Atende os requisitos?

Não consegui executar o código

Pontos de melhoria:

Há muitas funções com retornos duplicados, deixando o código repetitivo.

Apresenta erro ao rodar o pytest e possui vários erros do guia de estilos PEP8.

Amostra 3:

Atende os requisitos?

Não consegui executar o código

Pontos de melhoria:

Há muitas funções com retornos duplicados, deixando o código repetitivo.

Apresenta erro ao rodar o pytest e possui vários erros do guia de estilos PEP8.

Semana 15

Amostra 1:

Atende os requisitos?

Sim, foi utilizado uma estrutura de repetição lendo cada linha com o next() e em seguida adicionando cada linha em uma lista. Depois, em um for, os dados são gravador de acordo com o ano de eleição.

Pontos de melhoria:

O código passa uma vez pelo arquivo e depois passa pela lista para então gravar o arquivo, o que pode levar certo tempo e não é algo imediato.

Amostra 2:

Atende os requisitos?

Sim, foi utilizado uma função para criar um gerador pelo arquivo candidaturas.

Os anos foram passados manualmente em uma lista para depois criar um for e gravar em cada arquivo os registros de cada ano

Pontos de melhoria:

Como os anos foram passados manualmente, posteriormente, caso tivesse registros mais novos ou de outros anos o código não funcionaria.

Amostra 3:

Atende os requisitos?

Sim, foi utilizado uma função para criar um gerador pelo arquivo candidaturas.

O autor criou um for in range para passar pelos anos e então gravar seus respectivos arquivos com os dados da eleição.

Pontos de melhoria:

Como os anos foram passados por uma estrutura de repetição com limite, posteriormente, caso tivesse registros mais novos ou de outros anos o código não funcionaria.

Amostra 4:

Atende os requisitos?

Sim, o autor fez o tratamento destes por pandas.

Pontos de melhoria:

Como não foi utilizado o método gerador, o código acaba alocando muita memória do computador para executar, pois armazena todo o arquivo na memória ao invés de ler linha a linha.

Amostra 5:

Atende os requisitos?

Sim, o código abre o csv uma vez para pegar quais anos existem, e depois abre novamente para criar os arquivos com as eleições de cada ano

Pontos de melhoria:

O código passa duas vezes pelo arquivo, o trabalho de pegar o ano poderia ser realizado ao mesmo tempo que é gravado o CSV.