Resolução Semana 17 - Code Review

Semana 11

Amostra 1: O conjunto de arquivos da amostra1 atende a resolução dos desafios propostos. A classe Estrategia_Texto1 (Estrategias.py) faz herança da classe abstrata Estrategia e extrai os dados do arquivo txt informado (arquivo_texto1_modelo1.txt) e retorna uma lista de tuplas. Há pontos que podem ser melhorados, por exemplo, seguir um guia de estilos como PEP-8. Há comentários de códigos não utilizados que podem ser removidos para maior legibilidade. A classe Estrategia_Texto2 (Estrategias.py) foi implementada do mesmo modo da classe anterior com ajustes das posições do índice das strings para atender outro modelo de txt.

Amostra 2: O conjunto de arquivos da amostra2 atende a resolução dos desafios propostos. A classe Estrategia_Texto1 (Estrategias.py) faz herança da classe abstrata Estrategia e extrai os dados do arquivo txt informado (arquivo_texto1_modelo1.txt) e retorna uma lista de tuplas. Há pontos de melhorias, por exemplo, ao percorrer o arquivo txt é feito um replace na linha substituindo uma certa quantidade de espaços em branco por ‘;’ a fim de delimitá-los e trabalhar como um novo arquivo, poderia ter sido isolado esse trecho de código em uma função para separar as responsabilidades e tornar mais claro a manipulação. A classe Estrategia_Texto2 (Estrategias.py) foi implementada do mesmo modo da classe anterior com os devidos ajustes para atender outro modelo de txt.

Amostra 3: O conjunto de arquivos da amostra3 atende a resolução dos desafios propostos. A classe Estrategia_Texto1 (Estrategias.py) faz herança da classe abstrata Estrategia e extrai os dados do arquivo txt informado (arquivo_texto1_modelo1.txt) e retorna uma lista de tuplas. Um ponto de melhoria observado seria ao ignorar o cabeçalho e campos não desejáveis do arquivo txt, foi criados várias condicionais com o uso de startswith para identificar os textos a serem ignorados, uma boa prática seria criar uma lista com tais textos e fazer a conferência a partir delas, deixando apenas um if. A classe Estrategia_Texto2 (Estrategias.py) foi implementada do mesmo modo da classe anterior com os devidos ajustes para atender outro modelo de txt.

Semana 12

Amostra 1: O conjunto de arquivos da amostra1 atende a resolução dos desafios propostos. Foram feitas as implementações propostas utilizando o padrão Bridge para criar novas características de produtos (Caracteristicas.py) e novas classes abstratas (Produtos.py). Um ponto de melhoria, poderia ser retirado os comentários do script produtos.py pois as instâncias estão com nomes claros que permite identificar do que se trata.

Amostra 2: O conjunto de arquivos da amostra2 atende a resolução dos desafios propostos. Foram feitas as implementações propostas utilizando o padrão Bridge para criar novas características de produtos (Caracteristicas.py) e novas classes abstratas (Produtos.py). Como sugestão de melhoria para garantir a qualidade dos códigos seria fazer mais testes unitários.

Amostra 3: O conjunto de arquivos da amostra3 atende a resolução dos desafios propostos. Foram feitas as implementações propostas utilizando o padrão Bridge para criar novas características de produtos (Caracteristicas.py) e novas classes abstratas (Produtos.py). Os

códigos implementados estão em conformidade com o guia de estilos python PEP-8. A cobertura de testes é superior a 90%.

Semana 13
Amostra 1: O conjunto de arquivos da amostra1 atende em partes a resolução dos desafios propostos. Nos Desafios 1 e 2 foram feitas alterações nas classes ERP1 e ERP2 do arquivo Extrair.py para retornar somente os campos total e vendido_em. Foi implementado a classe Relatorio_CSV (Relatorios.py) que tem a classe base Relatorios e faz a geração de um arquivo csv. Foi utilizado o mesmo código da classe Relatorio_TXT, alterando apenas a extensão do arquivo de txt para csv, porém sem alterar a estrutura, não houve a reestruturação do arquivo para ficar delimitado, precisaria ser revisado o código.

Amostra 2: O conjunto de arquivos da amostra2 atende a resolução dos desafios propostos. Nos Desafios 1 e 2 foram feitas alterações nas classes ERP1 e ERP2 do arquivo Extrair.py para retornar somente os campos total e vendido_em. Foi implementado a classe Relatorio_CSV (Relatorios.py) que tem a classe base Relatorios e faz a geração de um arquivo csv. Foi utilizado a biblioteca csv para auxiliar na construção do arquivo, criando assim um arquivo delimitado. Um ponto de melhoria seria remover os códigos comentados pois prejudica a legibilidade.

Amostra 3 : O conjunto de arquivos da amostra3 atende em partes a resolução dos desafios propostos. Nos Desafios 1 e 2 foram feitas alterações nas classes ERP1 e ERP2 do arquivo Extrair.py para retornar somente os campos total e vendido_em. Foi implementado a classe Relatorio_csv (Relatorios.py) que tem a classe base Relatorios e deveria fazer a geração de um arquivo csv, porém há erros. Não está sendo importado a biblioteca csv, no script Abstracao.py está sendo invocado Relatorio_CSV que não existe, que deveria ser a classe criado anteriormente.

Semana 14
Amostra 1: O conjunto de arquivos da amostra1 atende em partes a resolução dos desafios propostos. Foi implementado a classe da rede social Github redes_sociais.py) e sua sessão UploadCode (sessoes.py), também implementado a classe da rede social Instagram. Não foi feita a criação de testes do script sessões.py apenas do redes_sociais.py. Há um erro no script principal (main.py), foi feito o import da classe instagram mas quando o usuário executa o script pede para fazer o input de ‘intagram’, há um erro de digitação e de um eventual teste de execução.

Amostra 2: O conjunto de arquivos da amostra2 atende em partes a resolução dos desafios propostos. Foi implementado a classe da rede social Github redes_sociais.py) e sua sessão UploadCode (sessoes.py), também implementado a classe da rede social Instagram. Não foi feita a criação de testes do script sessões.py apenas do redes_sociais.py. Há um erro de pacote no script redes_sociais.py ao tentar importar as sessões do script sessoes.py, como os arquivos estão no mesmo diretório deveria ter sido feito uma importação relativa, isso impediu a execução do código principal.

Amostra 3: O conjunto de arquivos da amostra3 atende em partes a resolução dos desafios propostos. Foi implementado a classe da rede social Github redes_sociais.py) e sua sessão UploadCode (sessoes.py), também implementado a classe da rede social Instagram. Não

foi feita a criação de testes do script sessões.py apenas do redes_sociais.py porém com erro (test_redes_sociais.py). Está ausente o arquivo __init__.py no mesmo diretório do teste, o pytest precisa desse init para reconhecer como módulos de forma padrão. Há um erro de pacote no script redes_sociais.py ao tentar importar as sessões do script sessoes.py, como os arquivos estão no mesmo diretório deveria ter sido feito uma importação relativa, assim como na amostra2.

Semana 15

Amostra 1: O arquivo amostra1.py atende a resolução proposta: ler o arquivo csv (candidatura.csv) e gerar arquivos CSVs menores segmentado pelo ano de eleição. O arquivo é aberto uma vez para extrair o cabeçalho e depois aberto novamente para percorrer o conteúdo e separar pelo conteúdo do ano adicionando em um dicionário, no final sendo percorrido o dicionário e criado os respectivos arquivos csvs. Não foram implementados testes.

Amostra 2: O arquivo amostra2.py atende a resolução proposta: ler o arquivo csv (candidatura.csv) e gerar arquivos CSVs menores segmentado pelo ano de eleição. Foi criado uma lista manualmente com os anos das eleições, isso é um problema caso venha a surgir outro ano no arquivo pois exigiria uma inclusão manual na lista. O arquivo é aberto várias vezes para filtrar o ano informado, demora mais para fazer a geração dos arquivos. O código foi estruturado em funções, mas faltou chamar a função gera_csv para dar início a transformação.

Amostra 3: O arquivo amostra3.py atende a resolução proposta: ler o arquivo csv (candidatura.csv) e gerar arquivos CSVs menores segmentado pelo ano de eleição. Foi implementado um range para incrementar os anos e verificar se contém no arquivo para fazer a escrita. O problema está no resultado final da escrita, os arquivos estão sem cabeçalhos e as linhas foram escritas como uma única string.

Amostra 4: O arquivo amostra4.py atende a resolução proposta: ler o arquivo csv (candidatura.csv) e gerar arquivos CSVs menores segmentado pelo ano de eleição. A solução adotada faz uso da biblioteca pandas, o arquivo csv é carregado em um dataframe e depois filtrado pelo ano e gerado o respectivo csv ainda utilizando a função to_csv do pandas. O pandas carrega todo o arquivo na memória, seria interessante fazer uso da propriedade chunksize para ler o arquivo em pedaços.

Amostra 5: O arquivo amostra5.py atende a resolução proposta: ler o arquivo csv (candidatura.csv) e gerar arquivos CSVs menores segmentado pelo ano de eleição. O arquivo é aberto uma primeira vez para extrair o cabeçalho com os anos, depois é percorrido a lista de anos para fazer a escrita. O código é funcional mas poderia ser refatorado para abrir o arquivo uma única vez. Não foram implementados testes.