Especificação do Trabalho 3

Objetivos

1. Usar os conhecimentos abordados em estruturas compostas e arquivos.
2. Organização do código em funções.

Introdução

Você foi encarregado de escrever um back-end para algum software de academia que permitirá a uma academia manter estatísticas de seus usuários. Você começará sem nenhuma informação e receberá um nome de arquivo que representa alguns exercícios de um usuário. Isso pode acontecer várias vezes. Além disso, você será solicitado a gerar e recuperar informações sobre os exercícios que já carregou.

Formato do treino

O formato de arquivo de treino é bastante genérico. Primeiro, a capitalização será irrelevante em todo este formato. Todas as strings conterão apenas letras, espaços e nenhum caractere especial. Todos os nomes serão considerados idênticos, independentemente da capitalização utilizada. Um nome terá duas strings - um nome e sobrenome. A primeira linha de qualquer arquivo de treino será um nome e nada mais. O restante do arquivo conterá dados sobre os treinos. Cada exercício começará com uma data, cujo formato será [dia] - [mês] - [ano]. Esta será a única coisa nessa linha. Em seguida, haverá uma linha opcional que contém um campo: Peso. O Peso terá o seguinte formato: Peso: [peso]. O peso é em kg.

Cada linha subsequente terá o formato:
[Nome do exercício]: [peso da primeira série], [repetições na primeira série]; [peso da segunda série], [repetições na segunda série]; [. . . .]; [peso da última série], [repetições na última série]

O nome do exercício pode ser qualquer sequência válida de strings. Em particular, ele não será necessariamente escolhido a partir de algum conjunto de nomes de exercícios canônicos. O peso é um inteiro seguido por kg. O número de repetições é um inteiro. Se um exercício for um treino, ele contém pelo menos uma série. O uso de kg não precisa ser consistente em todo o conjunto. Diferentes exercícios podem ter diferentes números de repetições. Assim que houver uma linha de espaço em branco, ou uma data, isso representa o fim das informações sobre o treino que acabou de ser escrito. Os treinos não precisam estar em ordem cronológica.

O primeiro treino (primeiro em termos de data do calendário) para qualquer pessoa terá seu peso. Se um treino não contém uma linha de peso, você deve assumir que durante aquele treino a pessoa pesou tanto quanto o treino anterior (de acordo

com o calendário) em que a pessoa especificou uma data. Um exemplo de arquivo de treino está aqui.

Fulano Ciclano
10-10-2010
Peso: 75kg
Agachamento: 0kg,6;20kg,6;40kg,6;60kg,6;75kg,6;75kg,6;75kg,6
Flexões: 0kg,10;0kg,10;0kg,10

21-12-2011
Agachamento: 0kg,6;20kg,6;40kg,6;60kg,6;75kg,6;75kg,6;75kg,6
Supino: 55kg,6;44kg,10

21-11-2010
Peso: 76kg
Agachamento: 0kg,6;20kg,6;40kg,6;60kg,6;75kg,6;75kg,6;75kg,6
Supino: 55kg,6;44kg,10

Recuperação da informação

Você será responsável por carregar muitos desses arquivos e, em seguida, retornar as informações neles contidas. No entanto, você não será responsável apenas por recuperar as informações contidas nesses arquivos. Você também precisará gerar meta-informações contidas neles. Isso significa que você precisará pensar cuidadosamente sobre como projetar seus objetos para que possa gerar facilmente essas meta-informações. Você deve ler as funções que precisa gerar com atenção e, em seguida, pensar em como deseja carregar os dados para poder responder a essas perguntas facilmente.

Sua tarefa

Você precisa implementar as seguintes funções:

- carrega_arquivo(nome_arquivo: str) -> NoneType: Esta função adiciona as informações do arquivo denominado nome_arquivo às suas estruturas de dados para que as chamadas subsequentes às outras funções possam retornar os valores corretos.
- pega_exercicios(nome: str, data: str) -> lista de str: Retorne uma lista de todos os nomes de exercícios realizados por nome na data. Pode retornar uma lista vazia.
- pega_conjunto_exercicios(nome: str, data: str, nome_exercicio: str) -> lista de tuplas (float, int): Retorne uma lista representando os conjuntos de nome_exercicio realizados por nome na data. Cada elemento da lista é uma tupla de dois elementos. O primeiro elemento é um float que representa o peso do exercício em kg. O segundo é um int que representa o número de repetições naquele peso. Pode retornar uma lista vazia.
- pega_treino(nome: str, data: str) -> lista de (str, lista de (float, int) tuplas)): Retorna uma representação do treino realizado por nome na data. Pode

retornar uma lista vazia. Cada exercício do treino é representado por uma string que é o nome do exercício e uma lista. Cada elemento da lista é uma tupla de dois elementos. O primeiro elemento é um float que representa o peso do exercício em kg. O segundo é um int que representa o número de repetições naquele peso.

- max_peso_exercicio(nome_exercicio: str) -> (str, str, float): Retorne o nome e a data da pessoa que teve o maior peso em nome_exercicio, junto com o peso.
- max_peso_total(nome_exercicio: str) -> (str, str, float): Retorna o nome, data e peso levatando por uma pessoa no nome_exercicio (contando todas as repetições em uma única série) em todas as séries e repetições de um determinado exercício em um dia. Portanto, se para o supino, a pessoa A levantasse 100 kg 3 vezes e a pessoa B levantasse 75 kg 5 vezes, essa função retornaria ('B', '11-11-2010 ', 375,0) (assumindo que B realizou este treino em 11 de novembro de 2010). Esta função pesquisa em todas as datas possíveis.
- max_peso_data(data: str, peso: float) -> lista de str: Retorna uma lista de nomes cujo peso corporal era menor que o peso na data.

Requisitos adicionais

- Não precisa construir entrada e saída para o usuário. Adicionarei chamadas de teste na main.
- Feche os arquivos abertos ao final da leitura.
- Não use break ou continue no código.

Código

Para este trabalho, não é fornecido um código inicial. Peço que você crie um código chamado trabalho3.py contendo todas as funções acima. Também deve conter um "__main__" para que sejam colocados códigos de teste pelo professor.

Não serão aceitos trabalho com outros nomes e extensões.

Nota

Estes são os aspectos do seu trabalho que podem ser avaliados para nota:

- Estilo do código (20%):
  - Certifique-se de seguir as diretrizes de estilo do Python que apresentamos e as convenções de codificação do Python que temos usado ao longo do semestre.
  - Todas as funções, incluindo funções auxiliares, devem ter docstrings completas, incluindo pré-condições, quando você achar que elas são necessárias.
- Corretude (80%):
  - Suas funções devem ser executadas conforme especificado.

Como submeter

A submissão é feita no sigaa. Está liberada a ressubmissão. Submeta o seu código trabalho3.py. Não é zip e nem é doc.