Armazém

Estaremos construindo um conjunto de funções para modelar um simples sistema de contabilidade de armazém, que mantém o estoque para um conjunto de mercadorias, que representaremos por strings, por exemplo, `'a'`, `'b'`, `'c'`. Portanto, o armazém pode ter 10 unidades de `'a'`, 20 de `'b'` e 0 de `'c'`.

Haverá transações no armazém que aumentam a quantidade de uma mercadoria, por exemplo, `('receive', 'a', 10)` que aumenta o total de `'a'` em 10, ou `('ship', 'a', 10)` que diminui o total de `'a'` em 10.

Representaremos os totais para as várias mercadorias usando um dicionário onde as chaves são os nomes das mercadorias e os valores são os totais atuais das mercadorias.

Parte 1: Processo

Escreva uma função `warehouse_process` que receba dois argumentos:

- um dicionário representando os totais do armazém, e
- uma transação (que é uma tupla, conforme descrito acima) Esta função deve modificar o dicionário que é passado, mas não deve retornar nada.

Certifique-se de lidar com o caso de uma transação `receive` quando a mercadoria não estiver presente no dicionário; simplesmente trate o total atual para aquela mercadoria como zero. Se não houver estoque suficiente para completar uma transação `ship`, transfira todas as unidades disponíveis e imprima a string `'not enough'` (por exemplo, se 10 unidades de `'a'` foram requisitadas, mas há apenas 8 unidades no dicionário, o dicionário deve ser atualizado de modo que `'a'` mapeia a 0 e a mensagem `not enough` deve ser impressa).

Parte 2: Uma Solução Mais Elegante

Escreva a definição da classe `Warehouse` que possui os seguintes métodos:

- `__init__`, que inicializa um dicionário vazio para conter o inventário
- `process`, que processa uma transação, conforme descrito no problema anterior
- `lookup`, que retorna o suprimento total atual para uma determinada mercadoria (0 se não estiver presente) Uma interação típica pode ser assim:

```
w = Warehouse()
w.process(('receive', 'a', 10))
w.process(('ship', 'a', 7))
print(w.lookup('a'))  # imprime 3
print(w.lookup('b'))  # imprime 0
```

Você pode desejar incluir sua definição de `warehouse_process` para que possa fazer uso dela.

Submissão

Quando estiver pronto (depois de ter simulado manualmente e testado em sua própria máquina e estiver convencido de que seu programa fará a coisa certa), faça upload do seu arquivo Python no **Problema 4.1**