Questão 1

Parcialmente correto

Atingiu 8,40 de 10,00

PyDiner Dash 2

Os proprietários de um restaurante estão com dificuldade de gerenciar os pedidos realizados pelas mesas, assim como também ter controle em relação aos ingredientes que estão no estoque do restaurante. Desta forma, vocês devem implementar um programa para ajudar a solucionar esses dois problemas.

Objetivo

O objetivo deste trabalho é criar um programa na linguagem de programação Python onde seja possível gerenciar um restaurante manipulando suas mesas, suas áreas, suas ocupações, seu cardápio, seus ingredientes, seus pedidos e seu histórico implementando conceitos de programação como entrada e saída de dados, variáveis, condicionais, laços de repetição, funções, listas, dicionários, leitura de arquivos e validação de entrada.

Neste trabalho, as informações relativas à configuração do restaurante (áreas, mesas e ocupação), itens do cardápio e estoque de ingredientes serão sempre obtidas de arquivos texto externos. Será necessário, portanto, implementar código para a leitura dos arquivos e atualização das informações correspondentes armazenadas em seu programa. Finalmente, o sistema deverá gerar uma série de relatórios para o acompanhamento do estado das informações armazenadas.

Inicialização

Inicializa o restaurante sem mesas, sem cardápio e sem estoque.

Imprime a seguinte mensagem no início do programa:

```
=  ante aberto
```

Aviso!

Para marcar o início da inserção de comandos que irão controlar o restaurante.

Comandos

1. Atualizar mesas

Esse comando tem como objetivo atualizar as configurações da mesa. Os dados serão lidos de um arquivo e em seguida as atualizações serão realizadas em variáveis do próprio programa.

Entrada:

Insira o comando:

```
+ atualizar mesas
```

Após o comando **deve-se ler o nome de um arquivo contendo uma configuração de mesas do restaurante.**

Saída:

Não imprime nenhuma mensagem.

ATENÇÃO

Cada número de mesa é único no restaurante. Logo, caso seja lido algum número de mesa já existente no restaurante, deve-se atualizar esta mesa com as novas informações de área e de ocupação.

Este comando pode:

- Inserir novas mesas no restaurante;
- Inserir novas áreas no restaurante caso seja lida alguma mesa em uma área nova;
- Atualizar informações de mesas já existentes.

2. Atualizar cardápio

Aviso!

O proprietário do restaurante pode modificar o cardápio do seu restaurante, por isso, o seu programa deve ser o comando de atualizar cardápio. Os cardápios apresentados no seu programa devem ser lidos a partir de um arquivo, e manipulados utilizando estruturas dentro do próprio programa.

Entrada:

Insira o comando:

```
+ atualizar cardapio
```

Após este comando, **deve-se ler um arquivo contendo uma configuração de cardápio contendo itens de cardápio e seus respectivos ingredientes.**

Saída:

Não imprime nenhuma mensagem.

ATENÇÃO

Caso seja lido um item que não exista no cardápio atual, deve-se adicioná-lo ao cardápio junto com seus respectivos ingredientes.

Caso seja lido um item que já exista no cardápio, deve-se atualizar este item com as novas informações de ingredientes, sobrescrevendo as antigas.

3. Atualizar estoque

O Controle de estoque é uma parte fundamental para o funcionamento de um restaurente, já que você precisa determinar se há ingredientes para montar os pedidos do cardápio. Como nos comandos anteriores o seu programa deve ler esses dados de um arquivo.

Entrada:

Insira o comando:

```
+ atualizar estoque
```

Após este comando, **deve-se ler um arquivo contendo uma configuração de estoque contendo ingredientes e suas respectivas quantidades.**

S Aviso!

Não imprime nenhuma mensagem.

*****ATENÇÃO*****

Caso seja lido algum ingrediente que não exista no estoque, deve-se adicioná-lo ao estoque com sua respectiva quantidade.

Caso seja lido algum ingrediente que já exista no estoque, deve-se adicioná-lo ao estoque somando sua respectiva quantidade com a quantidade já armazenada no estoque.

4. Relatório de mesas

O proprietário sempre precisará acessar informações sobre as mesas do seu restaurante. Sendo assim o comando Relatório de mesas apresentará essas informações.

Entrada:

Insira o comando:

```
+ relatorio mesas
```

Saída:

Imprime a configuração de mesas do restaurante **em ordem alfabética do nome da área e em ordem crescente de mesa,** no seguinte formato:

```
area: a_i
- mesa: m_1, status: s_1
.
.
- mesa: m_j, status: s_j
```

Onde
a_x corresponde às áreas existentes no restaurante, com $x \in \{1, 2, \ldots, i\}$, sendo i o número total de áreas;
m_y corresponde às mesas existentes em cada área, com $y \in \{1, 2, \ldots, j\}$, sendo j o número total de mesas;
s_y corresponde ao status da mesa m_y, ou seja, $s_y \in \{ocupada, livre\}$

- **Caso alguma área esteja sem mesas, imprima a seguinte mensagem nesta área:**

Aviso!

```
area: a_i
- area sem mesas
```

- Caso o restaurante não tenha mesas em nenhuma área, apenas imprima a seguinte mensagem:

```
- restaurante sem mesas
```

5. Relatório de cardápio

O proprietário do restaurante também pode acessar os dados do cardápio a qualquer momento. O Cardápio tem informações sobre os itens e seus ingredientes, assim como também a quantidade de cada item.

Entrada:

Insira o comando:

```
+ relatorio cardapio
```

Não são utilizadas linhas adicionais de input para o funcionamento deste comando.

Saída:

Imprime configuração dos itens do cardápio em ordem alfabética de nome e em ordem alfabética de ingredientes, no seguinte formato:

```
item: i_j
- ing_1: q_1
.
.
- ing_k: q_k
```

Onde
i_x corresponde a um item do cardápio, com $x \in \{1, 2, \ldots, j\}$, sendo j o número total de itens;
ing_y corresponde aos ingredientes de cada item, com $y \in \{1, 2, \ldots, k\}$, sendo k o número total de ingredientes;
q_y corresponde a quantidade do ingrediente ing_y necessária para produzir o item.

Aviso! o o cardápio esteja vazio, imprima a seguinte mensagem:

```
- cardapio vazio
```

6. Relatório de estoque

Seu programa deve permitir gerar um relatório com todos os ingredientes que estão no estoque.

Entrada:

Insira o comando:

```
+ relatorio estoque
```

Não são utilizadas linhas adicionais de input para o funcionamento deste comando.

Saída:

Imprime a relação de ingredientes do estoque **em ordem alfabética,** junto de suas respectivas quantidades, no seguinte formato:

```
ing1: qt1
.
.
.
ingk: qtk
```

Onde
ing_x correspondem aos ingredientes do estoque, com $x \in \{1, 2, \ldots, k\}$, sendo k o número total de itens;
qt_x corresponde a quantidade do item ing_x no estoque.

- **Caso o estoque esteja vazio, imprima uma mensagem.**

```
- estoque vazio
```

7. Fazer pedido

Neste comando, deve ser permitido definir a mesa e o seu respectivo pedido. Atenção as possibilidades neste comando, pois deve ser analisado se a mesa existe, se está ocupada, se o item do cardápio existe, e se há ingredientes suficientes no estoque para montar aquele pedido, como especificado a seguir.

Aviso!

Entrada:

Insira o comando:

```
+ fazer pedido
```

Para o funcionamento do comando, é necessária uma linha abaixo do comando especificando o pedido, da seguinte forma:

n, i

Onde:
n é um número inteiro, que indica o número da mesa que fez o pedido;
i é uma string, que indica o item do cardápio que foi pedido.

Saída:

Faça as verificações de erro nesta ordem:

1 - Mesa inexistente

Caso a mesa não exista no restaurante, imprima a seguinte mensagem de erro:

```
erro >> mesa n inexistente
```

Onde n é o número da mesa escrito no pedido.

2 - Mesa desocupada

Caso a mesa não esteja ocupada, imprima a seguinte mensagem de erro:

```
erro >> mesa n desocupada
```

Onde n é o número da mesa escrito no pedido.

3 - Item inexistente

Caso o item não exista no cardápio, imprima a seguinte mensagem de erro:

```
erro >> item i nao existe no cardapio
```

Onde i é o item do cardápio escrito no pedido.

Aviso!

redientes insuficientes

?

Caso falte ingredientes para produzir item, imprima a seguinte mensagem de erro:

```
erro >> ingredientes insuficientes para produzir o item i
```

Onde i é o item do cardápio escrito no pedido.

Deve ser impressa **no máximo** uma mensagem de erro.

- **Caso passe em todas as validações, imprima a seguinte mensagem de sucesso:**

```
sucesso >> pedido realizado: item i para mesa n
```

Onde n indica o número da mesa que fez o pedido e i indica o item do cardápio que foi pedido.

PARTICULARIDADES DO COMANDO

Ao realizar um pedido válido deve-se também executar as seguintes ações:
- Atualizar o registro de pedidos dessa mesa
- Atualizar o histórico de pedidos do restaurante
- Remover do estoque os ingredientes e suas respectivas quantidades que foram utilizados no preparo do pedido

ATENÇÃO

Ao realizar um pedido com sucesso, caso este pedido esgote algum ingrediente do estoque, deve-se deletar este ingrediente do estoque ao invés de deixá-lo zerado.

8. Relatório de pedidos

O seu programa também deve gerar relatório de pedidos de cada mesa do restaurante.

Entrada:

Insira o comando:

```
torio pedidos
```

Aviso!

Não são utilizadas linhas adicionais de input para o funcionamento deste comando.

Saída:

Imprime os pedidos feitos pelo restaurante **em ordem crescente de número de mesa e em ordem alfabética do nome dos itens pedidos,** no seguinte formato:

```
mesa: mj
- i1
.
.
- ik
```

Onde:
m_x corresponde ao número das mesas existentes, com $x \in \{1, 2, \ldots, j\}$, sendo j o número total de mesas
i_y corresponde aos itens pedidos por cada mesa, com $y \in \{1, 2, \ldots, k\}$, sendo k o número total de pedidos feitos por uma mesa

- **Caso nenhum pedido tenha sido feito, apenas imprima a seguinte mensagem:**

```
- nenhum pedido foi realizado
```

9. Fechar restaurante

Após o encerramento do expediente, o proprietário deve fechar o seu restaurante. A seguir são apresentados os comandos que o seu programa deve seguir nesta etapa.

Entrada:

Insira o comando:

```
+ fechar restaurante
```

Não são utilizadas linhas adicionais de input para o funcionamento deste comando.

Saída:

Imprima o histórico de pedidos do restaurante **em ordem cronológica**, ou seja, na ordem em que foram pedidos, no seguinte formato:

Aviso!

```
1. mesa m1 pediu i1
2. mesa m2 pediu i2
.
.
.
n. mesa mj pediu ik
```

Onde m são as mesas, e p são os pedidos que cada mesa fez.

- Caso não tenha sido feito nenhum pedido no restaurante, imprima a seguinte mensagem:

```
- historico vazio
```

Após imprimir o histórico, imprima na tela uma mensagem de fim do programa:

```
=> restaurante fechado
```

10. Comando inexistente

O seu programa deve identificar comandos não existentes. Uma mensagem deve ser enviada neste caso como apresentada no exemplo.

Entrada:

Quando for inserido um comanndo que não existe, ou seja, que não se encaixa em nenhum dos comandos mencionados acima.

Exemplo:

```
+ comer macarrao
```

Saída:

Imprima a seguinte mensagem de erro:

```
erro >> comando inexistente
```

Arquivos

Mesas

Aviso! o .csv contendo várias linhas de informação onde cada linha contém o número da mesa, sua respectiva área e ocupação.

Os arquivos .csv das mesas, que serão utilizados nos casos de teste, estão disponíveis para download neste link

Cardápio

Arquivo .csv contendo várias linhas de informação onde cada linha contém o nome de um item do cardápio e seus respectivos ingredientes de preparo.

Os arquivos .csv dos cardápios, que serão utilizados nos casos de teste, estão disponíveis para download neste link

PARTICULARIDADES DO ARQUIVO

Um item pode ter mais de uma unidade do mesmo ingrediente para ser preparado. Deve-se contabilizar todas as aparições deste ingrediente para preparar o item.

Estoque

Arquivo .csv contendo várias linhas de informação onde cada linha contém o nome de um ingrediente do estoque e sua respectiva quantidade.

Os arquivos .csv dos estoques, que serão utilizados nos casos de teste, estão disponíveis para download neste link

For example:

Input	Result

`+ relatorio pedidos`
`+ relatorio mesas`
`+ relatorio cardapio`
`+ relatorio estoque`
`+ fechar restaurante`
	`=> restaurante aberto`
	`- nenhum pedido foi realizado`
	`- restaurante sem mesas`
	`- cardapio vazio`
	`- estoque vazio`
	`- historico vazio`
	`=> restaurante fechado`

Aviso!