https://codecheck.io/private/resume/2111231919dyor78uacxk2tqaxpg6sq15ec/ebur-xawy-hini-emof/CEKJ88YQA50VXO3Q97BIFHT80

Dicionários

Preparação (fazer antes da aula)

Tente prever o que acontece nestes excertos no PythonTutor: Dicionários1, Dicionários2.

Faça estes exercícios CodeCheck: Dicionários3.

Exercícios

1. Escreva um programa que determine a frequência de ocorrência de todas as letras que ocorrem num ficheiro de texto. (Pode usar `c.isalpha()` para verificar se um caráter `c` é uma letra). O nome do ficheiro deve ser passado como argumento na linha de comando (use `sys.argv`). Descarregue "Os Lusíadas" (documento 3333 do Projeto Gutenberg) e faça a contagem. Ajuste o programa para não distinguir maiúsculas de minúsculas. (Pode usar `s.lower()` para converter uma string `s` para minúsculas.) Finalmente, modifique o programa para mostrar o resultado por ordem alfabética.

```
$ python3 countLetters.py pg3333.txt
a 32088
b 2667
c 7672
d 12846
e 33406
...
```

2. O programa `telefone.py` simula a lista de contactos de um telemóvel, implementada com um dicionário. O programa apresenta um menu com cinco operações. A operação "Listar contactos" já está implementada. Experimente e analise o programa.

   a) Acrescente a operação de "Adicionar contacto". Deve pedir um número e nome, e acrescentá-los ao dicionário.

   b) Acrescente a operação de "Remover contacto". Deve pedir o número e eliminar o item correspondente. (Use o operador `del` ou o método `pop`.)

   c) Acrescente a operação "Procurar Número". Deve pedir um número e mostrar o nome correspondente, se existir, ou o próprio número, caso contrário. *Sugestão: pode recorrer ao método `get`.* Repare que o dicionário permite uma solução mais simples e eficiente do que a solução que fez em `aula05/telephones.py`.

   d) Complete a função `filterPartName`, que dada uma string, deve devolver um dicionário com os contactos {número: nome} cujos nomes incluam essa string. Use essa função para implementar a operação "Procurar Parte do nome", que deve pedir um nome parcial e listar os contactos que o contêm. Aqui tem de fazer uma solução análoga à da função `nameToTels` de `aula05/telephones.py`.

3. Adapte o programa anterior para ser possível associar a morada a um contacto. Sugere-se que altere o dicionário para ter pares (`nome, morada`) como *valores* associados às chaves. Altere a função de listagem para mostrar os dados em 3 colunas com larguras fixas, como se vê abaixo: número ajustado à direita, nome centrado na coluna, morada ajustada à esquerda. Faça também as adaptações necessárias nas restantes operações.

```
   Numero :           Nome           : Morada
234370200 :  Universidade de Aveiro  : Santiago, Aveiro
876111333 :      Carlos Martins      : Porto
887555987 :        Marta Maia        : Coimbra
```

4. *Crie um programa que permita gerir um campeonato de futebol.

   a) O programa deverá pedir ao utilizador os nomes das equipas e guardá-los numa lista.

   b) Use a função criada no exercício 4 da aula05 para gerar uma lista com todos os jogos. Cada jogo é representado por um par (equipa1, equipa2).

   c) O programa deverá perguntar ao utilizador o resultado de cada jogo (golos de cada equipa) e registar essa informação num dicionário indexado pelo jogo. Por exemplo: `resultado[('FCP', 'SLB')] -> (3, 2)`.

   d) O programa deve manter uma tabela com o registo do número de vitórias, de empates, de derrotas, o total de golos marcados e sofridos, e os pontos de cada equipa. Com o resultado de cada jogo, deve atualizar os registos das duas equipas envolvidas. O melhor é manter os registos noutro dicionário indexado pela equipa. Por exemplo: `tabela['SLB'] -> [0,0,1, 2,3, 0]`.

   e) No final, apresente a tabela classificativa com as seguintes colunas: equipa, vitórias, empates, derrotas, golos marcados, golos sofridos e pontos. *Desafio: consegue ordenar a tabela por ordem decrescente de pontos? Faremos isso noutra aula*.

   f) Finalmente, deverá apresentar a equipa campeã. A campeã é a equipa com mais pontos ou, em caso de empate, a que tiver maior diferença entre golos marcados e sofridos.

5. *O ficheiro `nasdaq.csv` tem um registo das transações das ações de algumas empresas ao longo de um mês na bolsa de valores NASDAQ. Cada linha do ficheiro tem os campos seguintes, separados por TABs:

   ```
   Empresa Data ValorAbertura ValorMaximo ValorMinimo ValorFecho Volume
   ```

   O programa `stocks.py` tem uma função que lê esse ficheiro e devolve essa informação numa lista de tuplos. Complete as funções que faltam para colocar o programa a funcionar corretamente, respeitando as invocações feitas na função `main()`.

   a) Complete a função `totalVolume(lst)` para devolver um dicionário com a estrutura `{empresa: volumeTotal}`, que indique para cada empresa, qual o volume total transacionado no período completo.

   b) Complete `maxValorization(lst)` para devolver um dicionário com a estrutura `{data: (empresa, valorização)}` que, para cada data, indica qual a empresa com maior valorização diária relativa (ValorFecho/ValorAbertura - 100%) e qual essa valorização.

   c) Complete `stocksByDateByName(lst)` para devolver a informação num dicionário indexado por data e por nome da empresa.

d) Complete a função que calcula o valor de uma dada carteira de ações (um *portfólio*) de um investidor no fecho de uma dada data. A carteira de ações deve ser um dicionário com o número de ações de cada título, e.g.: `{'NFLX':100, 'CSCO':80}`.

6. O programa `coins.py` contém um conjunto de funções para gerir carteiras de moedas. Cada carteira (*bag*) é representada por um dicionário que a cada tipo de moeda associa o número dessas moedas na carteira. A lista `COINS` contém os tipos de moedas válidas, por ordem decrescente de valor (em cêntimos).

   a) Complete a função `value(b)` para devolver o montante total na carteira `b`.

   b) Complete a função `transfer1coin(b1, c, b2)` para tentar transferir uma moeda de tipo `c` da carteira `b1` para a `b2`. Se `b1` não tiver moedas do tipo `c`, a função deve devolver `False` e deixar as carteiras inalteradas. Se tiver, deve devolver `True` e atualizar o número de moedas nas duas carteiras.

   c) Altere a função `strbag(bag)` para devolver uma string com uma representação mais "amigável", com as quantidades de moedas por ordem decrescente do tipo de moeda, por exemplo.

   d) **Complete a função `transfer(b1, a, b2)` para tentar transferir um montante `a` de `b1` para `b2`. Deve fazê-lo à custa de várias transferências de uma moeda de cada vez. Se conseguir, a função deve devolver `True` e alterar as carteiras. Se não, deve devolver `False` e manter as carteiras intactas. *Atenção: este é um problema complexo*.