Conjuntos e Dicionários

Exercícios

1. Tente prever o resultado e os efeitos de cada uma das instruções abaixo. Algumas não têm resultado e outras dão erros. Use o Python em modo interativo para confirmar. Discuta e compare os resultados com os colegas e com o professor.

```
shop = {'eggs':6, 'sugar':1.0}
shop
{'sugar':1, "eggs":6} == shop
type(shop)
len(shop)
shop['eggs']
'eggs' in shop
6 in shop
shop[6]
shop.get(6, 'nada')
shop['sugar'] = 2.0
shop
shop.append('beer')
shop['beer'] = 6*0.33
shop
len(shop)
shop['beer'] += 0.33
shop
shop.keys()
shop.values()
shop.items()

d = {}
type(d); len(d); d
d[93542] = ('maria', 'P1')
d[95612] = ('daniel', 'P2')
d[76367] = ('john', 'P1')
len(d); d
d[95612][1]
```

```
for x in d:
    print(x, d[x])

for x,y in d.items():
    print(x, y, sep='->')

t = {'P1':[], 'P2':[]}
for x in d:
    t[d[x][1]].append(d[x][0])

len(t['P1'])
t
t.pop('P2')
t


set(['ana','diogo','marta'])
nums = {1, 2, 1, 3, 2}; len(nums)
{1, 2, 3} == {2, 1, 3, 2}
nums[1]
nums+{4}
2 in nums
S1 = {1, 2}; S2 = {2, 3}
S1 < S2
S1 == S2
S1 > S2
S1.add(4); S1
S1.remove(3)
S1.discard(5)
```

2. 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.) *Opcional: modifique o programa para mostrar o resultado por ordem alfabética (a ordenação será tratada noutra aula).*

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

3. 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`.* (Isto equivale à alínea 3a da aula 05, mas agora usando um dicionário.)

   d) Complete a função `filterPartName`, que dada uma string, deve devolver um dicionário com os contactos {número: nome} cujos nomes incluam essa string. (Isto é semelhante à alínea 3b da aula 05.) Use essa função para implementar a operação "Procurar Parte do nome", que deve pedir um nome parcial e listar os contactos que o contêm.

4. 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
```

5. O ficheiro `names.txt` tem uma lista de nomes completos de pessoas, com um nome por linha. Escreva um programa que mostre, para cada apelido (último nome), o conjunto de primeiros nomes encontrados na lista, sem repetições. O excerto abaixo é um exemplo do resultado pretendido. *Sugestão: construa um dicionário com chave = último nome e vá acrescentando os primeiros nomes ao conjunto associado a cada chave. Este é um problema que não se consegue reduzir facilmente a uma definição por compreensão.*

```
FIGUEIREDO : {'RUI', 'LUIS'}
SOARES : {'ELISABETE', 'VITOR', 'JENNIFER', 'RUBEN'}
MIRANDA : {'JOEL'}
```

6. No programa `primesupto.py`, defina a função `primesUpTo(n)` para devolver um conjunto com todos os números primos até `n`. Use o algoritmo do *crivo de Eratóstenes*: comece com o conjunto {2, 3, …, n}, depois elimine os múltiplos de 2 a começar em 2^2, depois os múltiplos de 3 a começar em 3^2, pode saltar o 4 porque já foi eliminado (bem como todos os seus múltiplos), depois continue eliminando os múltiplos de cada número que ainda se mantenha no conjunto. No fim, o conjunto conterá apenas os primos.

7. 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}`.

8. 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 aula06 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.