Folha 8 - Dictionaries

Python v 3.*

Exercícios

1) Considere o seguinte excerto de código:

```
d = {}
d['a'] = 'alpha'
d['g'] = 'gamma'
d['o'] = 'omega'
```

Qual o efeito das seguintes instruções:

a)
```
print(d)
print(d['a'])
d['a'] = 6
print('a' in d)
d['b'] = 'beta'
print(d['z'])
if 'z' in d: print(d['z'])
print(d.get('z'))
print(d.get('z','não existe!'))
```

b)
```
for key in d:
   print(key)
for key in d.keys():
   print(key)
for item in d.items():
   print(item)
print(list(d.keys()))
print(list(d.values()))
print(list(d.items()))
print(len(d))
print(d.len())
```

c)
```
d['b'] = 'beta'
d['e'] = 'epsilon'
del d[0]
del d[-2:]
print(d)
del d['b']
print(d)
```

d)
```
d1 = d.copy()
d2 = d
d1['b'] = 'beta'
del d1['o']
d['e'] = 'epsilon'
del d2['g']
print(d, d1, d2)
```

2) Indique quais os erros no seguinte excerto de código:

```
d={'a':1,'b':2,'c':3,'d':'e',4:(2,”dois”)}
d['a'] = d['o']

for a in d:
      d['b'] = d[a]+1

for b in d:
      if d[b] = -1:
            del d[b]

for c in d:
      if c < 'c':
            d['o'] = 'c'
```

3) Considere o seguinte excerto de código

```
d={}
for x in range(2,4):
    for y in range(4,7):
        d[(x,y)]=x*y
```

Identifique o resultado das seguintes sequências de instruções (se estas forem executadas depois do código inicial)

a)
```
print(d)
```

b)
```
values = list(d.values())
d={}
print(values, list(d.values()))
```

c)
```
for (x,y) in d:
    d[(2,y)]=0
print(d)
```

d)
```
for (x,y) in d.keys():
    d[(2,y)]=0
print(d)
```

e)
```
for (x,y) in d.keys():
    d[d[(x,y)]]=(x,y)
    del d[(x,y)]
print(d)
```

4) Considere o seguinte excerto de código:

```
import copy
simple1 = {}
simple1 ['a'] = 'alpha'
simple1 ['g'] = 'gamma'
simple1 ['o'] = 'omega'
simple2= {}
simple2 ['b'] = 'beta'
simple2 ['d'] = 'delta'
simple2 ['e'] = 'epsilon'
```

Indique, justificando, o resultado das seguintes instruções:

i)
```
d = {1:simple1, 2:simple2}
d1 = d
del d1[2]['b']
del d1[1]['a']
print(simple1, simple2, d1)
```

ii)
```
d = {1:simple1, 2:simple2.copy()}
d1 = d
del d1[2]['b']
del d1[1]['a']
print(simple1, simple2,d1)
```

iii)
```
d = {1:simple1, 2:simple2}
d1 = d.copy()
del d1[2]['b']
del d1[1]['a']
print(simple1, simple2,d1)
```

iv)
```
d = {1:simple1, 2:simple2}
d1 = copy.deepcopy(d)
del d1[2]['b']
del d1[1]['a']
print(simple1, simple2,d1)
```

Problemas

1) Considere duas variantes de uma loja: na primeira (alínea a)), a loja tem *stock* ilimitado; na segunda (alíneas b) e c)), a loja tem *stock* limitado.

   a) Escreva um programa que represente o preçário de uma loja (produto, custo) num dicionário e que, dado um pedido na forma de uma lista de compras (produtos e quantidades), consiga calcular o custo total do pedido. Assuma que a quantidade de cada produto, existente em armazém, é ilimitada (ou, equivalentemente, que pode ser encomendada qualquer quantidade necessária). Implemente interacção com utilizadores humanos.

   b) Escreva um programa que represente o inventário de uma loja (produto, custo e quantidade) num dicionário e que, dado um pedido na forma de uma lista de compras (produtos e quantidades), consiga calcular o custo total do pedido. Assuma que, se a quantidade pedida ultrapassar a existente, apenas será fornecida a que está em inventário. Implemente interacção com utilizadores humanos.

   c) Melhore o programa da alínea b) escrevendo uma versão que actualize o inventário após a satisfação do pedido do cliente. Note que o inventário é actualizado de forma passiva, ou seja, o programa não gera um pedido de produtos para repor o *stock*.

2) Defina um módulo `caixaSup.py` que apoie uma caixa de supermercado. Escolha uma representação para o conteúdo da caixa tendo como base um dicionário.

   a) Defina uma função que calcule o dinheiro em caixa de um dado tipo de denominação. Por exemplo, saber o valor existente em notas de €10.

   b) Defina uma função que calcule a totalidade do dinheiro em caixa.

   c) Escreva uma função que calcule trocos, recebendo o custo da aquisição, quantia entregue e a caixa. Com estes valores a função deve calcular menor conjunto de notas e moedas a entregar de troco tendo em atenção a variedade de dinheiro em caixa.

   d) Suponha agora que não deve esgotar as denominações em caixa, por exemplo, se tem 2 notas de €10 e 8 de €5 em caixa, não deve usar as duas de €10 no mesmo troco. Pode usar a regra que a denominação inferior não deve ultrapassar o dobro da denominação superior. Reescreva a função anterior com esta condicionante.

3) Escreva um programa que recorra a e teste o módulo `caixaSup.py`, fazendo uso de todas as funções do módulo.

4) Construa uma função que receba um ficheiro de texto como parâmetro e devolva um dicionário com cada uma das palavras do texto associada à lista de linhas em que ocorre no texto. Implemente instruções que testem a função.

5) Construa uma função que devolva um dicionário de "mimica" que associa a cada palavra que ocorre num ficheiro de texto uma lista de todas as palavras que a seguem nesse texto. Por exemplo, a palavra "carro" pode ter associada a lista ["amarelo", "veloz", "antigo"] indicando as palavras que surgem depois de "carro" no texto. Implemente instruções que testem a função.

6) Usando a função da alínea anterior, escreva uma função geradora automática de texto, que dada um ficheiro de texto inicial substitua cada palavra por outra obtida aleatoriamente do dicionário de "mimica". Se a palavra não existir no dicionário é substituída pela string vazia. Implemente instruções que testem a função.

7) Escreva um módulo `AddBook.py` que suporte as funcionalidades associadas a uma agenda telefónica. Defina o dicionário a usar de modo a conter o nome e o número de telefone de cada contacto, e as seguintes funções:
   a) Uma função `getPhone` que dado um nome e uma agenda devolve o número de telefone desse nome nessa agenda.
   b) Uma função `getName` que dado um número telefónico e uma agenda devolve o nome associado ao número nessa agenda.
   c) Funções que permitam adicionar e remover contactos de uma agenda dada.

8) Escreva um programa que recorra a, e teste, o módulo `AddBook.py`, fazendo uso de todas as funções do módulo.

9) Escreva um módulo `BigAddbook.py` que apoie as funcionalidades associadas a uma agenda telefónica com maior capacidade (use o módulo `AddBook.py`). Redefina o seu dicionário agrupando os contactos pela letra inicial do seu nome. Que tipo de estrutura seria a mais indicada supondo que esta agenda pode ter uma grande dimensão?

   a) Defina uma função `getList` que dada a letra inicial devolva um dicionário com todos os contactos cujo nome começa por essa letra.

   b) Defina uma função `getPhone` que dado um nome e uma `BigAddbook` devolva o respectivo número de telefone.

   c) Defina uma função `getName` que dado um número telefónico e uma `BigAddbook` devolva o nome associado.

   d) Defina funções que permitam adicionar e remover contactos da `BigAddbook`.

   e) Defina uma função `addbook2BigAddbook` que constrói uma `BigAddbook` a partir de uma `Addbook`.

   f) Defina uma função big`Addbook2Addbook` que constrói uma `Addbook` a partir de uma `BigAddbook`.

10) Escreva um programa que recorra a, e teste, o módulo Big`AddBook.py`, fazendo uso de todas as funções do módulo.

11) Redefina a `Addbook` de modo a poder incluir informação sobre a morada do contacto (rua, localidade e código postal). Adicione funções à agenda e à `BigAddbook` que permitam adicionar, remover o obter esta informação. Escreva um programa que recorra a e teste os módulos, fazendo uso de todas as suas funções.