Folha 7 - Strings, tuples, lists

Python v 3.*

Exercícios

1) Considere a seguinte representação de uma string:

`schoolName = "`Escola Professor Doutor Aníbal Cavaco Silva`"`

Indique o valor das seguintes expressões:

a) `schoolName[0]`

b) `schoolName[6]`

c) `schoolName[3:11]`

d) `schoolName[:7]`

e) `schoolName[17:]`

f) `schoolName[24]`

g) `len(schoolName)`

h) `len("")`

i) Quais os índices que a expressão `schoolName[???]`deve ter para representar "Cavaco" ?

j) `schoolName[:7] + schoolName[-5:]`

k) `'a' in schoolName`

l) `'Doutor' in schoolName`

m) `schoolName.index('Cavaco')`

n) `schoolName[schoolName.index('Cavaco'):]`

o) `[x for x in schoolName if not x in ['a','e','i','o','u']]`

2) Considere a frase `"`*Eu nunca me engano e raramente tenho dúvidas*`"`,

a) Crie uma variável `quote` com esta frase

b) Imprima a segunda letra de `quote` usando parênteses rectos.

c) Escreva uma expressão que obtenha a palavra "engano" de `quote`.

d) Escreva uma expressão que, tirando partido de `quote`, obtenha a frase a partir da palavra "raramente".

e) Construa uma expressão, com base em `quote`, que tenha como resultado a frase `"`*Eu raramente me engano e nunca tenho dúvidas*`"`

3) Considere o seguinte tuplo Python:

```
t = (1,'a',(2,3),4.3,'olá',10)
```

a) Quais os índices com que se pode aceder ao tuplo t. Faça também referência aos índices negativos.
b) Como pode aceder à componente que tem "olá" usando índices positivos?
c) E como aceder à mesma componente usando índices negativos?
d) Como determinar o comprimento do tuplo correspondente à terceira componente do tuplo t?
e) E como aceder à componente de valor 3 do tuplo `(2,3)` do tuplo t?
f) Qual a expressão que representa o tuplo `(4.3,'olá',10)` a partir de t?
g) E a expressão que representa o tuplo `(1,'a',(2,3))` a partir de t?
h) Indique duas forma de obter o tuplo `('a',(2,3),4.3,'olá')` a partir de t?
i) Como determinar se 4.3 faz parte do tuplo t? E como comprovar que 'b' não faz parte?
j) Verifique que 2 não faz parte do tuplo t, mas que faz parte do tuplo na terceira componente de t.
k) Como obter o tuplo `('a', 4.3, 10)` a partir de t?
l) E como obter o tuplo (10, 4.3, 'a')?
m) Junte o tuplo t ao tuplo `('b',3)` e atribua-lhe o nome t1.
n) Qual o comprimento de t1?
o) Junte ao tuplo t o tuplo `('b',3)` como um único elemento, de forma a que a última componente do tuplo resultante seja o tuplo `('b',3)`.

4) Considere o seguinte tuplo Python:

```
t = ([1,2,3],4,(2,3))
```

a) Qual é o efeito das seguintes atribuições:

```
 i) t = (1,2)
ii) t[1] = 5
iii) t[2][0] = 1
 iv) t[0][1] = 6
```

b) Em que consiste as operações de *packing* e *unpacking* de um tuplo? Qual o resultado das seguintes instruções:

```
 i) a, b = (1,[1,2])
ii) a, b, c = t
iii) for i in range(10):
         t = t + (i,)
```

5) Quantos elementos têm as seguintes listas:

a) [’a’,’b’]

b) [[’a’,’b’]]

c) [[’a’,’b’], [’c’,’d’]]

d) [[[’ a ’,’ b ’]]]

e) []

f) [[]]

g) [[],[]]

6) Considere a seguinte lista:

```
numeros = [1, 4, 9, 16, 32, 64]
```

a) Indique o valor das seguintes expressões:

```
numeros[0]
numeros[3]
numeros[6]
numeros[-2]
```

b) Indique o efeito das seguintes instruções:

```
numeros[0] = 3
numeros[3] = [2,3]
numeros[6] = 1
numeros[-2] = -1.0
numeros += [128,256]
```

7) Supondo que

lst1 = [1,2,3,4,5,6] e lst2 = [‘a’,’b’,’c’,’d’,’e’,’f’]

Qual o resultado das seguintes expressões:

a) `lst1 + lst2`

b) `lst1[1::4]`

c) `lst1[1::1] + lst2[2::2]`

d) `lst2 + lst1[-1::]`

e) `lst1.append(lst2)`

f) `lst1.append(lst2[2::3])`

8) Indique o valor das seguintes expressões:

a) `range(10)`

b) `range(0,10,2)`

c) `range(10) + range(10,20,3)`

d) `range(10,0,-1)`

e) `range(0,10,-1)`

9) No seguinte programa, identifique o efeito de cada instrução.

```
lt = ['huey', 'dewey', 'louie']
lt.append('scrooge')
lt.insert(0, 'donald')
lt.extend(['daisy', 'gearloose'])
print(lt)
print(lt.index('dewey'))
lt.remove('louie')
lt.pop(1)
other = lt
other.append(lt.pop(0))
print(lt)
lt.append('von drake')
print(lt)
another = other + []
another.pop(0)
print(lt, other, another)
```

10) Qual a lista resultante das seguintes instruções?

a) `[x for x in [1,2,3]]`

b) `[xyz for xyz in [4,6,2,3,7,4,1,2] if xyz % 2 == 0]`

c) `[y for y in range(10) if y > 9]`

d) `[y for y in [1,2,3,4] for z in [6, 4, 2, 8] if z == y]`

e) `[y+1 for y in [z*2 for z in [6, 4, 2, 8]]]`

f) `[y for y in [6, 4, 2, 8] for z in range(y) if z != y]`

11) Escreva uma lista em compreensão equivalente às seguintes instruções:

a)
```
squares = []
for i in range(10):
    squares.append(i**2)
```

b)
```
doubles = []
for i in range(10):
    doubles.append((i,2*i))
```

c)
```
combs = []
for x in [1,2,3]:
      for y in [3,1,4]:
            if x != y:
                  combs.append((x,y))
```

12) Identifique o efeito de cada instrução nos seguintes programas:

a)
```
a = [1,2,3]
b = [-1,0,a]
c = b
d = b[:]
a.append(4)
print(a,b,c,d)
```

b)
```
from copy import copy
a = [1,2,3]
b = [-1,0,a]
c = copy(b)
a = [1,2]
a.append(4)
b.append(5)
c.append(6)
print(a,b,c)
```

c)
```
a = [1,2,3]
b = [-1,0,a]
c = a
a = [1,2]
a.append(4)
c.append(5)
print(a,b,c)
```

d)
```
from copy import deepcopy
a = [1,2,3]
b = [-1,0,a]
c = deepcopy(b)
d = a
a = [1,2]
d.append(4)
b.append(5)
c.append(6)
print(a,b,c,d)
```

Problemas

1) Considere a frase `"`*Há limites para aquilo que o POVO português pode aguentar.*`"` e crie uma variável `famous_quote` com esta frase. Usando métodos da classe `String`, construa um programa que

a) Imprima o número de vezes que a sequência `"po"` ocorre em `famous_quote`.

b) Imprima o resultado de passar todos os caracteres de `famous_quote` a minúsculas.

c) Imprima o resultado de substituir todas as ocorrências em `famous_quote` de `"a"` por `"o"` e todas as ocorrências de `"o"` por `"a"`.

d) Imprima a lista de subsequências que resultam de quebrar `famous_quote` nos espaços em branco.

e) Depois de remover espaços em branco repetidos, imprima a lista de subsequências que resultam de quebrar `famous_quote` nos caracteres `"u"` e `"t"`.

f) Imprima o resultado de substituir em `famous_quote` os caracteres nas posições pares por `"2"`.

2) Escreva um programa com a função `tdiv` em linguagem Python que receba um número inteiro positivo e devolva um tuplo com os seus divisores. Nesse programa adicione interacção com o utilizador de forma a testar essa função.

3) Escreva um programa com a função `tsearch` que receba um tuplo de inteiros e determine o índice em que se encontra um dado valor. No caso do valor não se encontrar no tuplo, a função deve devolver `None`. Nesse programa adicione interação com o utilizador de forma a testar essa função.

4) Escreva um programa com a função `tascii` que construa um tuplo com os caracteres que têm o código ASCII entre 32 e 127 inclusive. Nesse programa adicione interação com o utilizador de forma a testar essa função.

5) Escreva um programa com a função `trepeat` que receba como parâmetro o número de componentes e um valor e devolva um tuplo com a dimensão especificada em que todas as componentes têm o valor passado como parâmetro. Nesse programa adicione interação com o utilizador de forma a testar essa função.

6) Escreva um programa com a função `trange` que se comporte como a função `range`, mas que construa um tuplo em vez de uma lista. Tenha em atenção que o limite inferior e o passo são opcionais, assumindo, por omissão, o limite inferior o valor zero e o passo o valor 1. Nesse programa adicione interação com o utilizador de forma a testar essa função.

7) Escreva um programa com a função `get_max(l)`, que recebe uma lista `l` de inteiros e devolve o maior número existente na lista. Nesse programa adicione interação com o utilizador de forma a testar essa função.

8) Escreva um programa com a função `find(n,l)` que recebe um elemento `n` e uma lista `l`, e devolve `True` se `n` existir em `l`. Nesse programa adicione interação com o utilizador de forma a testar essa função.

9) Escreva um programa com a função `choose_in_collection(l)`, que recebe uma lista `l` e devolve um elemento aleatório da lista (pode usar a função `randrange(n,m)` do módulo `random`). Nesse programa adicione interação com o utilizador de forma a testar essa função.

10) Escreva um programa com a função `count_even(l)` que recebe uma lista de inteiros e devolve quantos números pares existem nessa lista. Nesse programa adicione interação com o utilizador de forma a testar essa função.

11) Escreva um programa com a função `add_even(l)`, que recebe uma lista `l` de inteiros e devolve o soma dos pares existente na lista. Nesse programa adicione interação com o utilizador de forma a testar essa função.

12) Escreva um programa com a função `same_first_last(l)` que recebe uma lista e devolva `True` se a lista tem mais de um elemento e o primeiro e o último elemento são iguais, e `False` caso contrário. Nesse programa adicione interação com o utilizador de forma a testar essa função.

13) Usando a função `same_first_last(l)` escreva um programa também com a função `symmetrical(l)` que verifica se uma dada lista é igual à sua inversa. Nesse programa adicione interação com o utilizador de forma a testar essa função.

14) Escreva um programa com a função que dada uma lista de `strings`, devolva uma lista com as `strings` que verificam as seguintes condições:
- Tem mais de três elementos.
- É um palíndromo.

15) Escreva um programa com a função `histogram(l)` que recebe uma lista de inteiros e escreve o seu histograma no ecrã. Por exemplo `histogram([4,11,9])` deve imprimir o seguinte:

```
****
**********
*********
```

16) Escreva um programa com a função `remove_adjacent`, que recebe uma lista de inteiros e devolve outra lista ajustada onde todos os números adjacentes iguais foram reduzidos a um único número. Por exemplo,

```
remove_adjacent([1,2,2,3,4,4,4,5,3,3,3,6]) = [1,2,3,4,5,3,6]
```

17) Escreva um programa com a função `count_adjacent(l)`, que recebe uma lista `l` de inteiros ordenada e devolve uma outra lista de tuplos onde o primeiro elemento é o inteiro e o segundo o número de vezes que ele ocorre na lista. Por exemplo:

```
count_adjacent([1,2,2,2,2,3,4,4,4]) = [(1,1),(2,4),(3,1),(4,3)]
```

18) Escreva um programa com a função `removeAllcopy(l, n)`, que recebe uma lista `l` de inteiros ordenada e devolve uma outra lista onde todas as ocorrências de n foram removidas. Para efeitos pedagógicos, neste exercício use a função `copy.deepcopy(l)` e faça o ajuste sobre o seu output. Por exemplo:

```
removeAllcopy([1,2,3,2,3,4,4,4], 2) = [1,3,3,4,4,4]
```

19) Escreva um programa com a função `common_count(l1,l2)`, que recebe duas listas `l1` e `l2` e devolve quantos elementos têm em comum (as listas não têm elementos repetidos). Nesse programa adicione interação com o utilizador de forma a testar essa função.

20) Escreva um programa com a função `linear_merge(l1,l2)` que recebe duas listas ordenadas de inteiros e devolve uma outra lista ordenada resultante da fusão das duas listas originais.

```
linear_merge([1,3,4,6],[2,3,5]) = [1,2,3,3,4,5,6]
```

21) Escreva um programa com a função `append_lists(l1,l2)`, que recebe duas listas `l1` e `l2` e devolve uma nova lista resultado da junção das duas. Nesse programa adicione interação com o utilizador de forma a testar essa função. Não se esqueça de testar com listas de listas.

22) Escreva um programa com a função `swapMaxMin(l)`, que recebe uma lista `l` de inteiros e a altera da seguinte forma: o maior e o menor elemento trocam de posições. Nesse programa adicione interação com o utilizador de forma a testar essa função.

23) Escreva um programa com a função `swapMaxMinCopy(l)`, que recebe uma lista `l` de inteiros e devolve uma cópia da lista onde o maior e o menor elemento trocaram de posições. Nesse programa adicione interação com o utilizador de forma a testar essa função.

24) Escreva um programa com a função `swapSubLists(l)`, que recebe uma lista `l` de listas de inteiros e a altera da seguinte forma: duas das suas listas trocaram de posições, nomeadamente aquela cuja soma dos seus inteiros tem maior valor por aquela outra cuja soma dos seus inteiros tem menor valor. Nesse programa adicione interação com o utilizador de forma a testar essa função.

25) Escreva um programa com a função `swapSubListsCopy(l)`, que recebe uma lista `l` de listas de inteiros e devolve uma cópia de `l` onde duas das suas listas trocaram de posições, nomeadamente aquela cuja soma dos seus inteiros tem maior valor por aquela outra cuja soma dos seus inteiros tem menor valor. Nesse programa adicione interação com o utilizador de forma a testar essa função.

26) Escreva um programa com a função `replaceChars(l)`, que recebe uma lista `l` cujos elementos são caracteres e substitui cada caracter pelo caracter seguinte no alfabeto que fica a três posições de distância (considere o alfabeto como sendo circular). Nesse programa adicione interação com o utilizador de forma a testar essa função.

27) Escreva um programa com a função `replaceCharsCopy(l)`, que recebe uma lista `l` cujos elementos são caracteres e substitui cada caracter pelo caracter seguinte no alfabeto que fica a três posições de distância (considere o alfabeto como sendo circular). Nesse programa adicione interação com o utilizador de forma a testar essa função.