Programação I

Listas (ficha 10)

v2.0

1. Determine, no papel, o valor resultante de cada uma das seguintes expressões. Valide os resultados no interpretador de Python.
   (a) `a=[1,2,3,4,5,6,7,8]; a[2]`
   (b) `a=[1,2,3,4,5,6,7,8]; a[7]`
   (c) `a=[1,2,3,4,5,6,7,8]; a[-1]`
   (d) `a=[1,2,3,4,5,6,7,8]; a[-4]`
   (e) `a=[1,2,3,4,5,6,7,8]; a[4:]`
   (f) `a=[1,2,3,4,5,6,7,8]; a[:4]`
   (g) `a=[1,2,3,4,5,6,7,8]; a[2:8:2]`
   (h) `a=[1,2,3,4,5,6,7,8]; a[2:2:2]`
   (i) `a=[1,2,3,4,5,6,7,8]; x=0; x in a`
   (j) `a=[1,2,3,4,5,6,7,8]; x=8; x in a`
   (k) `a=[1,2,3,4,5,6,7,8]; b=a; b is a`
   (l) `a=[1,2,3,4,5,6,7,8]; b=a; b[4]=9; b==a`
   (m) `a=[1,2,3,4,5,6,7,8]; b=a[:]; b[4]=9; b==a`
   (n) `a=[1,2,3,4,5,6,7,8]; b=a[:]; b[4]=9; a[4]=b[4]; b==a`
   (o) `a=[1,2,3,4,5,6,7,8]; a[0:4]= []; a`
   (p) `a=[1,2,3,4,5,6,7,8]; a[0:1]= [99]; a`
   (q) `a=[1,2,3,4,5,6,7,8]; a[1:1]= [99]; a`
   (r) `a=[1,2,3,4,5,6,7,8];  a[1:1]= range(100,115,5); a`

2. Crie uma função que verifica se os elementos de uma lista estão ordenados por ordem crescente (devolve `True` se estiver ordenada e `False` caso contrário. Pode assumir que os elementos da lista podem ser comparados com os operadores relacionais `<`, `>`, ...

```
print(verifica_ordem([1,2,2]))
True
print(verifica_ordem([‘b’,‘a’]))
False
```

3. Implemente as seguintes funções:
   (a) `corta(lista)` que recebe uma lista e modifica-a, removendo o primeiro e o último elemento.

```
l = [1, 2, 3, 4]
corta(l)
print(l)
[2, 3]
```

   (b) `meio(lista)` que recebe uma lista e retorna uma nova lista que contém todos os elementos, excepto o primeiro e o último.

```
l1 = [1, 2, 3, 4]
l2 = meio(l1)
print(l1)
[1, 2, 3, 4]
print(l2)
[2, 3]
```

4. Crie uma função que permita contar o número de elementos comuns entre 2 listas. Considere listas apenas compostas por elementos únicos.

```
print(conta_elementos([1,2,3,4,5],[2,4]))
2
print(conta_elementos([1,2,3,4,5],[7,8]))
0
```

5. Crie uma função que permita realizar a soma dos valores das colunas de uma matriz.

```
print(soma_colunas([[1,2,3],[3,4,5],[6,7,8]]))
[10,13,16]
```

6. Crie uma função que permita verificar se uma matriz quadrada (representada através de uma lista de listas) é a matriz identidade.

```
print(matriz_identidade([[1,0,0],[0,1,0],[0,0,1]]))
True
```

7. Implemente a função `combina(pals1, pals2)` que, dadas duas listas (de strings), imprime as strings resultantes de concatenar cada uma das palavras da primeira lista com cada uma das palavras da segunda (separadas por espaço).

```
print(combina(['um','pequeno'],['teste','exemplo']))
um teste
um exemplo
pequeno teste
pequeno exemplo
```

8. Implemente a função `descodifica(pals, ordem)` que, dadas duas listas (uma de palavras e outra de inteiros), devolve uma string com as palavras concatenadas (com espaço como separador) ordenadas pela ordem correspondente.

```
print(descodifica(['exemplo','Isto','um','pode','ser'],[4,0,3,1,2]))
Isto pode ser um exemplo
```

9. Implemente um programa em Python para ler valores (não ordenados) correspondentes ao tempo em segundos que que vários atletas necessitam para completar 100m e mostrar a lista com o top3 onde estão os tempos mais baixos. Para isso deve ter um programa principal, onde lê quantos atletas existem, e o tempo de cada um, separadamente. A lista dos tempos deve ser passada como argumento a uma função `top3(lista)`, que ordena os valores e retorna uma nova lista com os 3 valores mais baixos.

```
quandos tempos pretende inserir? 4
tempo do atleta 1: 11
tempo do atleta 2: 9
tempo do atleta 3: 12
tempo do atleta 4: 11.5
o top3 de [11, 9, 12, 11.5] é [9, 11, 11.5]
```

10. Implemente uma função que, dada uma lista devolve outra lista cujos elementos em cada índice `i` são o acumulado dos valores dos elementos na lista original entre o índice `0` e `i`.

```
print(acumulado([1,2,3]))
[1, 3, 6]
```

11. Suponha que a tabela de preços dos artigos de uma loja. O preço de cada artigo é guardado numa lista com 2 elementos: o artigo e o preço (`['jarro',20.6]`); a tabela contém uma sequência de vários artigos (`[['jarro',20.6],['almofada',18],['candeeiro',45]]`). Implemente as funções:
    (a) `consulta_preco(artigo, tabela)`. Dado o nome de um artigo uma tabela, a função deve procurar o artigo e devolver o respetivo preço.
    (b) `artigo_mais_caro(tabela)`. A função deve imprimir o preço do artigo mais caro e devolver como resultado o nome desse artigo.

12. O rececionista de um restaurante toma nota do nome dos clientes, à chegada; depois, cada cliente espera que haja vaga numa mesa. Implemente um programa em Python que possa ser usado para a gestão da fila de espera para almoço. As operações importantes são:
    - adicionar um nome à fila
    - consultar o comprimento da fila de espera e imprimir todos os nomes
    - retirar os primeiros `n` nomes da fila, porque surgem `n` lugares sentados

    O programa deve funcionar em ciclo, devendo o rececionista escrever a opção que pretende executar (1,2,3) ou 0 para sair. Use uma lista para controlar a fila; para separar o tratamento de cada operação deve recorrer aos condicionais, dentro do corpo do ciclo.

```
indique a sua opcao:
adicionar nome (1), consultar lista (2),  retirar os primeiros n (3), sair (0)
opcao -> 1
nome a adicionar -> Ana 1

indique a sua opcao:
adicionar nome (1), consultar lista (2),  retirar os primeiros n (3), sair (0)
opcao -> 1
nome a adicionar -> Bruno 2

indique a sua opcao:
adicionar nome (1), consultar lista (2),  retirar os primeiros n (3), sair (0)
opcao -> 2
comprimento da fila: 2
- Ana 1
- Bruno 2

indique a sua opcao:
adicionar nome (1), consultar lista (2),  retirar os primeiros n (3), sair (0)
opcao -> 3
```

```
quantos nomes conseguem vaga e saem da fila? -> 1

indique a sua opcao:
adicionar nome (1), consultar lista (2),  retirar os primeiros n (3), sair (0)
opcao -> 0
fim de execucao
```

Utilize, como ponto de partida, o código abaixo onde já tem o menu com as opções:

```
# gestao da fila de espera para o almoco
# ciclo com menu de opcoes: adicionar, consultar, retirar

mensagem="\nindique a sua opcao:\
\nadicionar nome (1), consultar lista (2),  retirar os primeiros n (3), sair (0)"
while True:
  print(mensagem)
  op= int(input("opcao -> "))
  if op==0:
    break
  # ver que operacao sera realizada
  # ... se op == 1 invocar funcao para consultar lista...
print("fim de execucao")
```

13. Pense numa estrutura baseada em listas para guardar o conteúdo de um carrinho de compras. No início não tem compras. As operações possíveis são:

    - adicionar produto com nome P, em N unidades (se o produto já estiver no carro, apenas incrementa N à quantidade existente)
    - remover todas as unidades do produto com o nome P
    - mostrar o valor pagar pelo conteúdo actual do carrinho de compras. Para isto deve multiplicar as unidades pelo preço unitário. Este valor deve ser encontrado numa "tabela" independente do carrinho de compras, que tem um valor para cada produto, identificado pelo seu nome, e que já está inicializada no início do programa - como no exercício 11.)

    Implemente um menu, através de um ciclo que sistematicamente mostra as opções e lê a intenção do utilizador (ver exercicio 12). Para processar cada opção, implemente uma função própria, invocada com os argumentos que achar necessários.

14. Considere um conjunto de `M` palavras chave e `N` documentos. Apresente uma forma de guardar a matriz de ocorrências de cada palavra em cada documento (uma tabela `MxN`), recorrendo a listas de listas.

    (a) Implemente a função `mostraMatriz(matriz)` para imprimir tal matriz de ocorrências na forma de uma tabela. Terá de percorrer as listas...

    (b) Implemente a função `ocorreEmDoc(matriz, palavra, doc)`, que devolverá True se a palavra ocorre dentro do documento doc, ou False em caso contrário.

    (c) Como obtem as ocorrências da palavra "mesa" no documento "d1"? Implemente uma função para devolver as ocorrências de uma palavra num determinado documento: `getOcorrencias(matriz,palavra,doc)`

    (d) Com a representação escolhida, implemente outra função para obter o total de palavras num dado documento, dados dois argumentos: a matriz e o nome do documento.