Folha prática 5 Introdução à Programação, DCC/FCUP, 2009/2010

Nota. Os exercícios 8 e 9 destinam-se a introduzir algoritmos que mais tarde serão explicados em mais pormenor nas aulas teóricas. Os exercícios 4, 5 e 6 usam listas em comprensão.

1. Onde está?
   Escreva uma função `ocorrencias(txt,c)` que retorna uma lista com os índices das ocorrências do caracter `c` na ´string´ `txt`. Por exemplo:

   ```
   >>> ocorrencias("banana", "a")
   [1, 3, 5]
   ```

2. Quem divide?
   Um número d é *divisor* de n (ou equivalente, n é *múltiplo* de d) se e só se o resto da divisão de n por d for zero. Escreva uma função `divs(n)` que calcula a lista dos divisores de n por ordem crescente. Exemplo:

   ```
   >>> divs(12)
   [1, 2, 3, 4, 6, 12]
   ```

   Utilizando esta função, como pode testar facilmente se um inteiro `n` é primo?

3. Em A mas não em B
   Escreva uma função `dif(A,B)` para construir a diferença

   $$A \setminus B = \{a \in A : a \notin B\}$$

   de dois conjuntos A e B representados como listas. Pode assumir que cada lista não tem elementos repetidos. A ordem dos elementos no resultado não é importante. Por exemplo:

   ```
   >>> dif([2, 3, 5], [1, 2, 3])
   [5]
   ```

4. Compreensão...
   Este exercício destina-se a introduzir o conceito de lista em compreensão.

   Vamos calcular o valor da seguinte lista (uma lista em compreensão)

   ```
         EXPR      GERADOR            FILTRO  FILTRO    GERADOR
        |___| |____________________| |_______| |____| |____________|
   >>> [x*x+y for x in [4,5,6,7,8] if x%2==0 if x>5 for y in [2,4]]
   x toma o valor 4
       verifica-se 4%2==0 mas não 4>5
   x toma o valor 5
       não se verifica 5%2==0
   x toma o valor 6
       verifica-se 6%2==0 e 6>5
          y toma o valor 2    --> 38 (=6*6+2) é o primeiro valor da lista
          y toma o valor 4    --> 40 (=6*6+4) é o segundo valor da lista
   x toma o valor 7
       não se verifica 7%2==0
   x toma o valor 8
       verifica-se 8%2==0 e 8>5
          y toma o valor 2    --> 66 (=8*8+2) é o terceiro valor da lista
          y toma o valor 4    --> 68 (=8*8+4) é o quarto valor da lista
   => O resultado é portanto [38, 40, 66, 68]
   ```

Uma lista em compreensão é da seguinte forma

```
[ <expressão> <geradores ou filtros> ]
<gerador> é: for <var> in <lista>  (ou outras formas do ´for´)
<filtro> é:  if <condição>
onde <condição> é evidentemente uma expressão com valor booleano.
```

As variáveis introduzidas pelos geradores ocorrem com frequência (mas não obrigatoriamente) na expressão.

Para cada alínea, e de acordo com o que conhece da função `range` e das listas em compreensão

(i) preveja o valor da expressão

(ii) teste a sua previsão no interpretador de comandos do python.

Nota. Inclui-se a função `divs` qua calcula a lista dos divisores de `n`.

```
def divs(n):
    li=[]
    for d in range(1,n+1):
        if n%d==0:
            li.append(d)
    return li
```

```
A) range(10)   B) len(range(1000))
C) range(0)    D) range(2,5)       E) range(2,5,12)
F) [x for x in [7,8,9]]   G) [x for x in [7,8,9] if x>8]
H) [x+y for x in range(10) for y in range(20) if x+y>=27]
I) [x+y for x in [7,8,9] for y in [10,20]]
J) [x for x in range(100) if len(divs(x))>=10]
K) [x for x in range(100) if len(divs(x))==2]
L) [hora for (hora,temperatura) in [(2,10),(3,15),(4,11),(5,16)]
      if temperatura>=12]
```

Descreva por palavras as listas obtidas nas alíneas `J)`, `K)` e `L)`,

5. Só uma lista
   Resolva o exercício 2, usando uma só instrução de `return`. Sugestão: use uma lista em compreensão.

6. Pautas...
   Pretende-se efectuar operações sobre pautas com classificações de alunos. A pauta é representada por uma lista de pares $(nome, nota)$ em que *nome* é uma ´string´ e *nota* é uma classificação de 0 a 20. Por exemplo:

```
>>> pauta = [("Ana",13.0), ("Joao",12.0), ("Pedro",8.0)]
```

   As alíneas (a)–(d) devem ser resolvidas com expressões usando listas em compreensão e funções pré-definidas como `len`, `max` e `sum`.

   (a) Escreva uma expressão para calcular a lista dos nomes de alunos aprovados (i.e. com pelo menos 9.5 valores).

   (b) Escreva uma expressão para calcular o número de alunos aprovados.

   (c) Escreva uma expressão para calcular a classificação média de todos os alunos.

   (d) Escreva uma expressão para calcular a classificação média dos alunos aprovados.

(e) Admita agora que cada aluno pode ocorrer em mais que um par (classificações nas diversas disciplinas) como no exemplo em baixo. Escreva uma função `melhor(nome, pauta)` que calcula a melhor classificação de um aluno. Por exemplo:

```
>>> melhor("Ana", [("Ana", 8.5), ("Pedro",8.0), ("Ana", 13.0)]
13.0
```

7. Polinómios

Um *polinómio* $p(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$ numa incógnita x pode ser representado pela lista de $n+1$ coeficientes $[a_0, a_1, a_2, \ldots, a_n]$ ordenada por potências crescentes de x. Verifica-se sempre $a_n \neq 0$; em particular, a lista que representa o polinómio 0 é []

(a) Escreva uma função `soma(p,q)` para calcular o polinómio soma de $p(x)$ e $q(x)$; os polinómios dados e o resultado devem ser listas de coeficientes.

(b) qual o resultado da função da alínea anterior se $p(x)$ e $q(x)$ tiverem graus diferentes (i.e. listas de coeficientes de comprimentos diferentes)? Em geral, que relação há entre os graus de $p(x)$, $q(x)$ e $(p+q)(x)$? *Atenção*: há um caso especial.

(c) Escreva uma função `calcular(p,x)` para calcular o valor do polinómio p em x, i.e.

$$p(x) = \sum_{i=0}^{n} a_i x^n$$

(d) Escreva uma função `horner(p,x)` para calcular o valor $p(x)$ usando a *fórmula de Horner*:

$$p(x) = a_0 + x(a_1 + x(\cdots(a_{n-1} + xa_n)\cdots))$$

Esta formula é mais eficiente do que a da alínea anterior: usa n somas e n multiplicações e não necessita de calcular potências.

(e) Escreva uma função `deriv(p)` que calcula os coeficientes do polinómio derivada de $p(x)$:

$$p'(x) = a_1 + 2a_2x + \cdots + na_nx^{n-1}$$

O resultado deve ser a lista de coeficientes de $p'(x)$.

8. Por ordem!

A *ordenação por selecção* é um algoritmo simples para colocar por ordem crescente uma sequência de elementos $[x_0, x_1, \ldots, x_{n-1}]$. A ideia é encontrar o menor valor da sequência e colocá-lo na primeiro posição; depois encontrar o segundo menor valor dos restantes e colocá-lo segundo lugar e assim sucessivamente.

1. para $i = 0$ até $n-2$:
   a) $k \leftarrow i$ (k vai guardar o índice do menor elemento)
   b) para $j = i+1$ até $n-1$: se $x_j < x_k$ então $k \leftarrow j$
   c) troca x_i e x_k

Implemente este método como uma função `ordena(xs)` para ordenar uma lista `xs`. Experimente a sua implementação com lista de números e strings, por exemplo:

```
>>> ordena([5, 3, -1, 3, 2, 7, 4])
    [-1, 2, 3, 3, 4, 5, 7]
>>> ordena(["bc", "aa", "a", "ab"])
    ["a", "aa", "ab", "bc"]
```

Exemplo.

0	1	2	3	(índices)
|4	6	2	3	$i = 0$, $k = 2$
2	|6	4	3	$i = 1$,$k = 3$
2	3	|4	6	$i = k = 2$
2	3	4	|6	(resultado final)

9. Onde está, depressa!

   Para testar se um valor ocorre numa sequência arbitrária necessitamos de comparar esse valor com cada um dos n elementos da sequência. Com uma sequência longa, esta operação pode ser lenta.

   Caso a sequência esteja ordenada é possível poupar trabalho fazendo uma *pesquisa binária* que divide sucessivamente a sequência ao meio até encontrar o valor. Seja $[x_1, x_1 \ldots, x_n]$ uma sequência de n elementos por ordem crescente e v um valor que queremos procurar; o algoritmo seguinte determina o índice de v (se v ocorrer na sequência):

   1. inicialmente $i \leftarrow 1$, $j \leftarrow n$ (i, j serão os limites dos índices onde v pode ocorrer)
   2. enquanto $i \leq j$:

      (invariante de ciclo: se v está na lista x, então tem um índice compreendido entre i e j, extremos incluídos).

      $m \leftarrow \lfloor (i + j)/2 \rfloor$ ($\lfloor x \rfloor$ é a parte inteira de x, i.e. tomamos o *quociente* da divisão inteira por 2);
      se $v = x_m$ então o algoritmo termina e a resposta é m
      se $v < x_m$ então $j \leftarrow m - 1$
      senão $i \leftarrow m + 1$
   3. o algoritmo termina sem encontrar o valor.

   Como em cada iteração dividimos o intervalo (i, j) ao meio, o número máximo de iterações para encontrar um valor numa sequência de n elementos é $\log_2 n$.

   Escreva uma função `pesquisa(xs,v)` que implemente a pesquisa binária. O resultado deve ser o índice do valor v ou -1 se v não ocorre na lista. Note que o primeiro índice da lista `xs` é 0 (e não 1).