Folha prática 3 Introdução à Programação, DCC/FCUP, 2009/2010

Nota importante. Nos exercícios desta disciplina, sempre que se pedir para escrever uma função, esta deve em princípio ser “pura”, isto é, limitar-se a calcular e devolver (através da instrução `return`) um valor que depende exclusivamente dos parâmetros (e não de outras variáveis globais). Em particular as funções não devem incluir instruções de entrada/saída como `print` ou `input`, a menos que isso seja explicitamente dito no enunciado do exercício – tal como acontece nesta folha com o exercício 3.4.(a).

1. *Testar divisibilidade*

- (a) Escreva uma função `testaDivisivel(n,m)` que testa se `n` é divisível por `m` (ambos inteiros). O resultado deve ser `True` em caso afirmativo e `False` em caso negativo. A definição da função deverá conter uma só instrução (do tipo `'return'`).
- (b) Experimente a função com alguns valores. O que acontece quando `n = 0`? E quando `m = 0`? Interprete o resultado.

2. *No intervalo?*

Escreva uma função `contido(x,a,b)` que verifica se o valor real `x` está contido no intervalo real `[a, b]`. O resultado deverá ser `True` se `x ∈ [a, b]` e `False` caso contrário. Pode assumir que `a ≤ b`. A definição da função deverá conter uma só instrução (do tipo `'return'`).

3. *Como pode ser $x > 0$ e $1 + x = x$?!?!*

Por causa dos erros de arredondamento, a seguinte sequência de testes dá `True` a partir de certo `eps`

			eps	p
<code>1 + 1</code>	<code>== 1</code>	<code>-> False</code>	1	0
<code>1 + 0.5</code>	<code>== 1</code>	<code>-> False</code>	0.5	1
<code>1 + 0.25</code>	<code>== 1</code>	<code>-> False</code>	0.25	2
<code>1 + 0.125</code>	<code>== 1</code>	<code>-> False</code>	0.125	3
<code>1 + ?????</code>	<code>== 1</code>	<code>-> True</code>	???	?

Determine, (escrevendo um programa!) qual o menor valor de `p` tal que o teste `1+eps==1` dá o resultado `True`, onde `eps` tem o valor 2^{-p} .

4. *Aproximadamente iguais...*

Por causa dos erros de arredondamento, não se deve testar a igualdade entre dois números representados em vírgula-flutuante. É preferível comparar o valor absoluto da diferença entre os dois valores. Por exemplo, o erro absoluto entre 0.33330 e 0.33335 é $|0.33330 - 0.33335| = 5.0 \times 10^{-5}$.

Escreva uma função `aprox(x,y,eps)` cujo valor é `True` se o erro absoluto entre `x` e `y` é inferior a `eps` e `False` no caso contrário.

5. *Quatro quadrantes*

- (a) Escreva uma função `quadrante(x,y)` imprime qual o quadrante (I, II, III ou IV) em que se encontra o ponto (x,y) no plano (esta função não é pura). Assuma-se que $x \neq 0$ e $y \neq 0$.
- (b) Modifique a função anterior para que seja devolvido um valor inteiro associado ao quadrante (1, 2, 3 ou 4)... e nada seja impresso.

6. *Primos...*

Considere a seguinte função que testa a primalidade de um inteiro $n \geq 2$

```
def primo(n):
    d=2
    while d*d <= n:
        # Neste ponto n nao e' divisivel por 2,3,...,d-1
        if n%d==0:
            return False
        d=d+1
    return True
```

- (a) Justifique informalmente a correção da função. Mostra a validade do comentário incluído na função (invariante de ciclo); use indução matemática na variável `t`.
- (b) Escreva uma função (não pura) `ate(m)` que chama a função `primo` (alínea anterior) para imprimir todos os primos até ao limite `m` (inclusivé).
- (c) Escreva uma função (pura) `soma(m)` que chama a função `primo` que retorna `True` se `m` pode ser escrito como a soma de 2 primos e `False` caso contrário. Assim, `soma(32) -> True` pois, por exemplo, $32=3+29$.

7. *Triângulos...*

Um triângulo com lados a , b e c diz-se *equilátero* se os lados são todos iguais, *isósceles* se apenas dois lados são iguais e *escaleno* se os três lados são diferentes. Escreva uma função `tipoTriangulo(a,b,c)` que retorna 'equilatero', 'isosceles' ou 'escaleno' conforme o tipo do triângulo (em minúsculas e sem acentos).

8. *Existe tal triângulo?*

Para que se possa construir um triângulo é necessário que a medida de qualquer um dos lados seja menor que a soma das medidas dos outros dois.

Escreva uma função `testaTriangulo(a,b,c)` que verifica essas condições. O resultado deve ser um valor booleano (`True` ou `False`).

9. *Dias no mês*

Escreva uma função `dnm(mes,ano)` que calcule o número de dias no mês (inteiro de 1 a 12) num determinado ano. O resultado deve ser o número de dias (entre 28 e 31), p.ex.

```
>>> dnm(2,2009)
28
```

porque Fevereiro de 2009 teve 28 dias. Recorde que Fevereiro tem 29 dias apenas nos anos bissextos e que um ano é bissexto se for múltiplo de 4 e não ser múltiplo de 100 (excepção) e não ser múltiplo de 400 (excepção à excepção); por exemplo são bissextos 2000 e 2004 mas não 1900.

Janeiro	31 dias
Fevereiro	28 ou 29 dias
Março	31 dias
Abril	30 dias
Maió	31 dias
Junho	30 dias
Julho	31 dias
Agosto	31 dias
Setembro	30 dias
Outubro	31 dias
Novembro	30 dias
Dezembro	31 dias

Comece por escrever uma função `bissexto(n) -> {False,True}`. De preferência a definição da função deverá conter uma só instrução (do tipo 'return').

10. *Combinações...*

Para $n, m \geq 0$ o número $\binom{m}{n}$ de combinações de m objectos tomados n a n é definido indutivamente por (propriedade simples do triângulo de Pascal)

$$\binom{m}{n} = \begin{cases} 0 & n > m \\ 1 & n = 0 \\ \binom{m-1}{n} + \binom{m-1}{n-1} & \text{nos outros casos} \end{cases}$$

Sem usar o computador, use esta definição para desenhar o triângulo de Pascal para $0 \leq m, n \leq 5$. Escreva uma função `comb(m,n)` baseada directamente na definição dada que retorna $\binom{m}{n}$. Forma pretendida do triângulo de Pascal:

```

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
... ..
```

11. *Cenas dos próximos episódios...*

Escreva um ficheiro com o nome 'xxx' e o conteúdo

```
As armas e os baroes assinalados
que, da ocidental praia lusitana
partiram, um dia, talvez, por mares nunca dantes navegados
etc, etc...
```

Execute o seguinte programa, onde a parte indicada deve conter o caminho completo da raíz para o seu fiheiro

```
#          |-----|
f = open("/home/bernardina/AULAS/IP10/xxx","r")
ch = f.read(1)
while len(ch)>0:
    if ch in "aeiouAEIOU":
        ch="u"
    print ch,
    ch = f.read(1)
f.close()
```

Interprete os resultados que obteve. Comente-os. Tente explicar o funcionamento do programa, indicando em particular o efeito das expressões read(1) e ch in "aeiouAEIOU".