Aula prática nº 7 – Leitura e escrita de ficheiros. Exceções.

Exercícios

1) Crie um programa que calcule a soma de uma lista de valores gravados num ficheiro. Considere que o ficheiro contém apenas um valor por linha, como no exemplo abaixo. O nome do ficheiro deve ser pedido ao utilizador.

nums.txt
```
12.39
1.93
7.85
...
```

2) O ficheiro `mystery.txt` contém linhas com as palavras UP, DOWN ou com um par de números. UP e DOWN são instruções para a tartaruga levantar ou pousar o pincel. Os pares são coordenadas x, y de pontos. Complete o programa `turtledraw.py` para ler as instruções do ficheiro e usar uma tartaruga para traçar o desenho. *Sugestão: use o método `.split()` para dividir as linhas que têm coordenadas. (Adaptado do exercício 11.5 do livro "How to think like a computer scientist".)*

3) O ficheiro `school.csv` contém uma tabela com notas de uma disciplina. Cada linha tem o *registo* de um aluno e cada coluna tem um *campo* de informação. As colunas são separadas caracteres TAB '\t'. A primeira linha contém um cabeçalho com os títulos dos campos.
   a) Crie uma função que leia o ficheiro e devolva uma lista com um tuplo por aluno. Cada tuplo deve ter os campos (numero, nome, nota1, nota2, nota3). Use o método `.split()` para dividir cada linha e converta as notas e os números para os tipos adequados.
   b) Crie uma função `notaFinal(reg)` que, dado o registo de um aluno, devolva a nota final calculada pela média das três notas no registo.
   c) Usando estas funções, escreva um programa que leia o ficheiro, ordene a lista (com `.sort()`) e mostre uma pauta com o seguinte aspeto:

```
Numero                      Nome                      Nota
  8540          VÍTOR MANUEL ANICETO PALHINHA         16.3
  9419               MÁRIO JORGE MANECAS               8.9
```

   O nome deve aparecer centrado, enquanto o número e a nota devem aparecer ajustados à direita. Use o método `.format`.

4) Altere o programa anterior para gravar a pauta formatada num ficheiro de texto. Pode usar o método `write` ou o argumento `file` na função `print`.

5) Sempre que executamos `float(input(...))`, corremos o risco de o utilizador introduzir um texto que gere um `ValueError` na conversão.

   a) Para resolver o problema, crie uma função `floatInput(prompt)` que faça a leitura com validação: pede um valor, tenta convertê-lo e, se falhar, avisa o utilizador e repete.

```
>>> floatInput("val? ")
val? pi
Not a float!
val? 3.1416
3.1416
```

   b) Acrescente dois argumentos `min` e `max` e valide se o valor está no intervalo [`min`, `max`]. Se não estiver, a função deve avisar e repetir.

```
>>> floatInput("val? ", min=1.3, max=2.1)
val? two
Not a float!
val? 2.2
Value should be in [1.3, 2.1]!
val? 1.3
1.3
```

   c) Torne os argumentos `min` e `max` opcionais. Quando omitidos, a função deve aceitar qualquer valor real. (Pode usar ±`math.inf`.)

6) Escreva uma função `compareFiles` que verifica se dois ficheiros são iguais. Para poupar tempo e memória, leia e compare blocos de 1 KiB de cada vez, e termine logo que descubra diferenças. Abra os ficheiros em modo binário e use a função `read`. Teste a função num programa que recebe os nomes dos ficheiros como argumentos. (1 KiB lê-se "1 kibibyte" e corresponde a 1024 bytes.)

7) Para saber o tamanho em bytes de um ficheiro, pode usar a função `os.stat("ficheiro").st_size`. Crie uma função que percorra um diretório (com `os.listdir`) e mostre o tamanho de cada ficheiro.