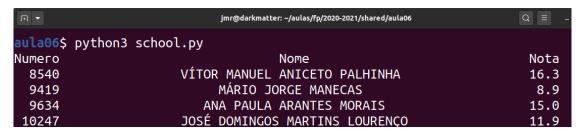
Leitura e escrita de ficheiros. Excepções.

Exercícios

1. O programa filesum.py pede um nome de um ficheiro e passa-o como argumento à função filesum. Complete essa função para calcular e devolver a soma dos valores do ficheiro. Considere que o ficheiro contém um valor por linha, como no exemplo abaixo. Na função main, experimente comentar a linha #A e descomentar #B.

nums.txt	
12.39	
1.93	

- 2. O ficheiro drawing.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. Cada par representa um ponto (x, y) para onde a tartaruga deve deslocar-se (com .goto). 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 [1].)
- 3. O ficheiro schooll.csv contém as notas dos alunos de uma turma. Cada linha tem o registo de um aluno e cada coluna tem um campo de informação. As colunas são separadas por caracteres TAB. A primeira linha contém um cabeçalho com os títulos dos campos. Os ficheiros schooll.csv e schooll.csv têm as notas dos alunos de outras turmas, com o mesmo formato. Complete o programa school.py para ler e processar esses ficheiros.
 - a) Complete a função loadFile(fname1, lst) para que, dado o nome de um ficheiro com este formato, leia o seu conteúdo e acrescente um tuplo por cada aluno à lista lst. Cada tuplo deve ter os campos (número, nome, nota1, nota2, nota3). Use o método .split('\t') para dividir cada linha e converta as notas e os números para os tipos adequados.
 - b) Crie uma função notaFinal (reg) que, dado um tuplo com o registo de um aluno, devolva a nota final calculada pela média das três notas no registo.
 - c) Crie uma função printPauta(lst) que, dada uma lista com registos de alunos, mostre uma tabela com os números, nomes e notas finais, formatados e alinhados como no exemplo abaixo. O nome deve aparecer centrado, enquanto o número e a nota devem aparecer ajustados à direita.
 - d) Usando estas funções, complete a função main () para ler os ficheiros, ordenar a lista com o método .sort () e mostrar a pauta.



- 4. Altere o programa anterior para gravar a pauta formatada num ficheiro de texto. Pode usar o método write ou a função print com o argumento file=. Sugestão: se generalizar a função printPauta para também receber um argumento file extra, pode reutilizá-la para gravar no ficheiro.
- 5. Execute o programa floatinput.py e experimente introduzir uma resposta não numérica. O programa terminará com uma excepção ValueError gerada pela chamada à função float dentro da função floatInput.

```
aula06$ python3 floatinput.py
a) Try entering invalid values such as 1/2 or 3,1416.
Value? 3,1416
Traceback (most recent call last):
   File "floatinput.py", line 27, in <module>
        main()
   File "floatinput.py", line 10, in main
        v = floatInput("Value? ")
   File "floatinput.py", line 4, in floatInput
        res = float(input(prompt))
ValueError: could not convert string to float: '3,1416'
```

a) Altere a função floatInput para validar o valor: pede um valor, *tenta* convertê-lo em float e, se falhar, avisa o utilizador e repete.

```
a) Try entering invalid values such as 1/2 or 3,1416.
Value? 3,1416
ERROR: Not a float!
Value? 355/113
ERROR: Not a float!
Value? 3.1416
v: 3.1416
```

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

```
b) Try entering invalid values such as 15%, 110 or -1.
Humidity (%)? 15%
ERROR: Not a float!
Humidity (%)? 110
ERROR: Value should be in [0, 100]!
Humidity (%)? 15
h: 15.0
```

c) *Torne os argumentos min e max opcionais. Quando omitidos, min deve assumir o valor $-\infty$ (-math.inf) e max deve assumir $+\infty$ (+math.inf).

```
c) Try entering invalid values such as 23C or -274.
Temperature (Celsius)? -274
ERROR: Value should be in [-273.15, inf]!
Temperature (Celsius)? -273.1
t: -273.<u>1</u>
```

- d) *Acrescente à função uma instrução assert para exigir que o argumento min não exceda max.
- 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 "*um 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.