Programação II

Exercícios 8
Programação de Sistema

Universidade de Lisboa
Faculdade de Ciências
Departamento de Informática
Licenciatura em Tecnologias da Informação

2020/2021

1. Escreva um *script* `para_minusculas` que imprima no ecrã o conteúdo do ficheiro com todas as letras convertidas para minúsculas. Exemplo:

   ```
   $ cat mensagem.txt
   A Europa jaz, posta nos cotovellos:
   De Oriente a Occidente jaz, fitando,
   $ python para_minusculas.py mensagem.txt
   a europa jaz, posta nos cotovellos:
   de oriente a occidente jaz, fitando,
   ```

   Sugestão: proceda em dois passos:

   (a) Escreva uma funcão `para_minusculas(nome_ficheiro)` que imprime no ecrã o conteúdo de um ficheiro com todas as letras convertidas para minúsculas.

   (b) Baseado na função anterior, escreva agora o *script* `para_minusculas.py` que lê da linha de comandos o nome do ficheiro e imprime no ecrã o conteúdo deste com todas as letras convertidas para minúsculas.

2. `more` é um filtro para visualizar o conteúdo de um ficheiro, uma página de cada vez. Na mais simples utilização, `more` é lançado da linha de comandos deste modo:

   ```
   $ python more.py o_meu_ficheiro
   ```

   Neste caso a primeira página do ficheiro aparece no ecrã. Depois, sempre que o utilizador carregar em `Enter` aparecerá uma nova página. Qualquer outra tecla seguida de `Enter` termina o programa.

(a) Implemente a versão simplificada do *script* `more.py` indicada acima.

(b) Um exemplo de interação mais complexo especifica o tamanho de cada página (`-c`):

```
$ python more.py -c 20 more.py
```

O valor por omissão para o comprimento da página é 10. Implemente esta funcionalidade no seu *script*. Sugestão: pode utilizar `len(sys.argv)` para obter o número de argumentos.

(c) Utilizando o módulo `argparse`, estenda o seu *script* de modo a possuir as seguintes opções:

- `-h, --help` mostra como utilizar o programa;
- `-c N, --comprimento N` indica o comprimento da página em número de linhas (valor por omissão: 10 linhas);
- `-l M, --linha M` indica o número da primeira linha a mostrar (valor por omissão: linha 1).

3. Escreva um *script* que, à semelhança do comando `wc` (*word count*) do sistema `Unix`, conte o número de palavras, caracteres, e bytes em vários ficheiros de entrada. Adicionalmente, o *script* deverá aceitar as seguintes opções:

   - `-h, --help` mostra como utilizar o programa;
   - `-c` mostra o número de bytes do ficheiro;
   - `-l` mostra o número de linhas.

   Por exemplo:

```
$ python conta_palavras.py mensagem.txt lusiadas.txt
2       12      73      mensagem.txt
1062    5347    30953   lusiadas.txt
1064    5359    31026   total

$ python conta_palavras.py -l mensagem.txt lusiadas.txt
2       mensagem.txt
1062    lusiadas.txt
1064    total

$ python conta_palavras.py -c mensagem.txt lusiadas.txt
73      mensagem.txt
30953   lusiadas.txt
31026   total

$ python conta_palavras.py -l -c mensagem.txt lusiadas.txt
2       73      mensagem.txt
```

```
1062    30953   lusiadas.txt
1064    31026   total

$ python conta_palavras.py mensagem.txt lusiadas.txt nao_existe.txt
2       12      73      mensagem.txt
1062    5347    30953   lusiadas.txt
conta_palavras.py: nao_existe.txt: open: No such file or directory
1064    5359    31026   total
```

4. Escreva um *script* para imprimir os ficheiros numa directoria e subdirectorias em forma de "árvore deitada". Para cada ficheiro, imprimos o seu nome; o nome de cada directoria deve ser precedido do sinal +. Cada nova directoria começa uma nova coluna dois espaços para a direita.

```
$ python pesquisa_arvore.py exemplos/
grafico.png
enunciado.tex
+codigo
  base_dados.csv
  +mais_codigo
    encontra_falhas.py
  analisa.py
  compila.py
calcula.py
```

5. Escreva um *script* que imprima o caminho absoluto de todos os ficheiros Python que se encontram numa dada directoria, incluindo as suas subdirectorias.

```
$ python todos_python.py exemplos
C:\ProgII\exemplos\codigo\mais_codigo\encontra_falhas.py
C:\ProgII\exemplos\codigo\analisa.py
C:\ProgII\exemplos\codigo\compila.py
C:\ProgII\exemplos\calcula.py
```

   Escreva duas versões:

   (a) Versão iterativa, utilizando `os.walk()`;

   (b) Versão recursiva, utilizando `os.listdir()`.

   Sugestão: para obter o caminho absoluto utilize a função `os.path.abspath()`.

6. (a) Escreva um *script* que imprima o maior ficheiro numa dada diretoria, incluindo as suas subdiretorias.

```
$ python maior_ficheiro.py .
8848798 ./olimpicos.csv
```

(b) Acrescente uma opção `-l` que mostre o tamanho desse ficheiro num número legível para humanos, seguindo as conversões:

- `1 TB = 1000 GB`;
- `1 GB = 1000 MB`;
- `1 MB = 1000 KB`;
- `1 KB = 1000 B`;

Utilize no máximo duas casas decimais. Exemplo:

```
$ python maior_ficheiro.py -l .
8.85MB  ./olimpicos.csv
```

7. (a) Escreva um *script* que imprima o caminho de todos os ficheiros numa certa diretoria cujo nome contenha uma *string* dada. Exemplo:

```
$ python encontra.py "ficheiro" .
ex01_ficheiros.py
maior_ficheiro.py
```

(b) Adapte a alínea anterior para que seja utilizada a diretoria corrente quando não for indicada a diretoria raiz.

(c) Acrescente duas opções `-e`, `-b` indicando que só devem ser retornados ficheiros que terminem (resp. comecem) com a *string* dada.

8. Este exercício tem como objectivo a escrita de *scripts* que imprimam o nome de todos os ficheiros Python numa certa diretoria que contenham alguma linha que emparelhe com uma certa expressão regular.

(a) Por exemplo, o comando seguinte lista todos os ficheiros Python no diretoria `src` que tenham alguma linha iniciada pela palavra "import":

```
$ python todos_regular.py '^import' src
src/cat.py
src/coop.py
src/frequentes.py
```

Sugestão: utilize a biblioteca para expressões regulares re, e adapte o exercício 5 acima.

(b) Adapte o exercício anterior para que seja utilizada a diretoria corrente quando não for indicada a diretoria raiz.

(c) Faça agora o seu *script* aceitar a opção `-r` indicando que a busca deve ser efectuada recursivamente nas subdiretorias.

9. Escreva um *script* que imprima a lista, sem repetidos e ordenada, de todos os módulos importados por todos os ficheiros Python constantes numa dada diretoria. Considere os padrões:

   - `import m1, ..., mn`
   - `from m import f`

   O nome de um módulo pode descrever submódulos utilizando um ponto: `from sound.effects.echo import echofilter`.

   ```
   >>> python todos_modulos.py sources
   [argparse, er, os, sys]
   ```

10. Escreva um *script* que dado um ficheiro comprimido no formato `.zip` mostre a lista de ficheiros nele contido, o tamanho original de cada ficheiro, e o seu tamanho comprimido.

    ```
    $ python lista_zip.py ex01_ficheiros_texto.zip

    Nome do ficheiro             Tamanho              Compressão
    frutas.txt                  26 bytes        26 bytes (100.0%)
    valores_comentarios.txt    157 bytes       121 bytes (77.1%)
    valores_vazio.txt           81 bytes        57 bytes (70.4%)
    valores.txt                 79 bytes        55 bytes (69.6%)
    ```

    Sugestão: utilize o módulo zipfile.

    (a) Para imprimir a lista de ficheiros de forma a que os nomes e tamanhos estejam alinhados, utilize 30 espaços para o nome do ficheiro, 10 espaços para o tamanho em bytes (antes e após a compressão), e 4 espaços para a taxa de compressão.

    (b) Adapte o exercício de forma a que o seu script percorra primeiro a lista de ficheiros para descobrir o número mínimo de espaços necessários para o nome do ficheiro e para os tamanhos em bytes.