



PROJECTO 2

PyCoder e Syms

(Beta)

INTRODUÇÃO E OBJECTIVOS

A finalidade deste projecto passa por aplicar os conhecimentos adquiridos até agora no desenvolvimento aplicações em Python 3. Em particular, pretende-se que desenvolva código modular, decomposto em funções, utilizando os tipos de dados apropriados e, sempre que possível, recorrendo a bibliotecas que implementem as funcionalidades de que precisa.

Na primeira parte do projecto, pretende-se que desenvolva uma aplicação para codificação e decodificação de ficheiros utilizando o método *Run-Length Encoding*. Na segunda parte, deve desenvolver uma aplicação comando para detectar ficheiros semelhantes num sistema de ficheiros. Todas as partes do projecto a desenvolver são mencionadas na secção **AValiação**. Nesta secção encontra também uma explicação sobre o que se entende por "extra" no contexto deste projecto.

PARTE I - PYCODER: CODIFICADOR/DESCODIFICADOR RLE

INTRODUÇÃO

Pretende-se que implemente um codificador e decodificador do tipo *Run-Length Encoding* (RLE), que podemos traduzir para algo como "*Codificador de Séries por Comprimento*". O RLE é muito utilizado para **comprimir blocos repetidos de informação** definidos consecutivamente, e integra outros métodos de compressão como, por exemplo, o DEFLATE (zip/gzip) ou o JPEG. Designamos estes blocos por "séries". O seu compressor deverá codificar séries de bytes, porém vamos ilustrar a sua aplicação na codificação de séries de caracteres. Considere a seguinte sequência de texto (tirada da Wikipedia):

WWWWWWWWWWBWWWWWWWWWWBWWWWWWWWWWBWWWWWWWWWWBWWWWWWWWWWBWWWWWWWWWWBWWWWWWWWWWB

Existem várias codificações RLE possíveis para esta sequência. As pretendidas neste exercício são as originam o seguinte *output*:

Método RLE A: 12W1B12W3B24W1B14W

Método RLE B: WW12BWW12BB3WW24BWW14

No **primeiro método** de codificação, cada série de caracteres consecutivos é resumida indicando o número ocorrências do caractere seguido do caractere.

No **segundo método**, apenas uma ocorrência de um caractere leva a que este seja passado para a saída inalterado. Uma série de duas ou mais ocorrências do mesmo caractere é codificada com a dimensão precedida de dupla ocorrência do caractere. Ou seja, o caractere é colocado duas vezes na saída, seguindo-se um número que indica o comprimento da série. Deste modo, quando o decodificador detecta um mesmo caractere duas vezes seguidas, sabe que o caractere foi codificado com RLE e que deve consultar o número que segue esta dupla ocorrência do caractere.

O algoritmo a implementar deve ser orientado ao byte. Para distinguir um byte da sequência original, da própria dimensão da série, esta dimensão deve ser codificada com um byte também. Isto leva a que sequências de mais de 255 ocorrências de um mesmo byte tenham que ser divididas em duas. Por exemplo (utilizando uma notação semelhante à do Python para representar bytes), a sequência

```
\x82\x7F\x7F\x7F\x7F\x9B\x9B... série de 300 ocorrências deste último byte...\x9B
```

é codificada da seguinte forma através do segundo método de codificação:

```
\x82\x7F\x7F\x04\x9B\x9B\xff\x9B\x9B\x2D
```

A negrito estão destacadas as dimensões das séries. Em decimal essas dimensões são 4, 255 e 45, respectivamente.

Consultar: https://en.wikipedia.org/wiki/Run-length_encoding

INTERFACE DA LINHA DE COMANDOS

O programa a desenvolver pode ser invocado sem opções ou utilizando as seguintes opções:

```
$ python3 pycoder.py (-c [-t TYPE] | -d) [-p PASSWD] FILE
```

As opções `-c` e `-d` (ou `--encode` e `--decode`) indicam se vai codificar ou decodificar o ficheiro dado por `FILE`. `TYPE` pode ser 1 (método A) ou 2 (método B), sendo 2 o valor por omissão. A versão longa desta opção é `--type=TYPE`.

À semelhança do comando `gzip`, após codificação (opção `-c`) o ficheiro `FILE` passa a possuir a extensão `.rle`. Esta extensão é removida após decodificação, tal como sucede com o comando `gunzip`.

Com a opção `-p` pode introduzir uma palavra-passe que será utilizada para para encriptar, através de criptografia simétrica, o ficheiro resultante da compressão RLE. Pode utilizar um módulo da biblioteca do Python ou uma biblioteca externa instalável com `pip` (eg, *cryptography*). Como alternativa, pode também desenvolver um algoritmo de encriptação. Um algoritmo pedagogicamente interessante, ainda que pouco seguro, e que se enquadra no espírito deste projecto, é a Cifra de Vigenère.

Utilize o módulo **docopt** para ler as opções da linha de comandos.

Cada ficheiro RLE deve ser precedido de um **cabeçalho** com 5 bytes:

- Byte 1: **método de compressão**. Para que o descompressor saiba o método de codificação RLE utilizado, o compressor deve guardar no primeiro byte do cabeçalho um identificador do método. O método 1 é identificado pelo valor 33, ao passo que o método 2 é identificado pelo valor 138. O descodificador decide então o método a utilizar em função deste identificador, assinalando um erro caso não tenha nenhum destes valores.
- Bytes 2-5: a **data/hora (timestamp)** da compressão do arquivo em segundos. Este é um número inteiro de 32 bits que pode ser obtido a partir de `time.time`. Deve ser gravado em formato *big endian* (veja `int.to_bytes` ou o módulo `struct`).

Pode utilizar o módulo **struct** para gerar e ler este cabeçalho, ou outro método que prefira.

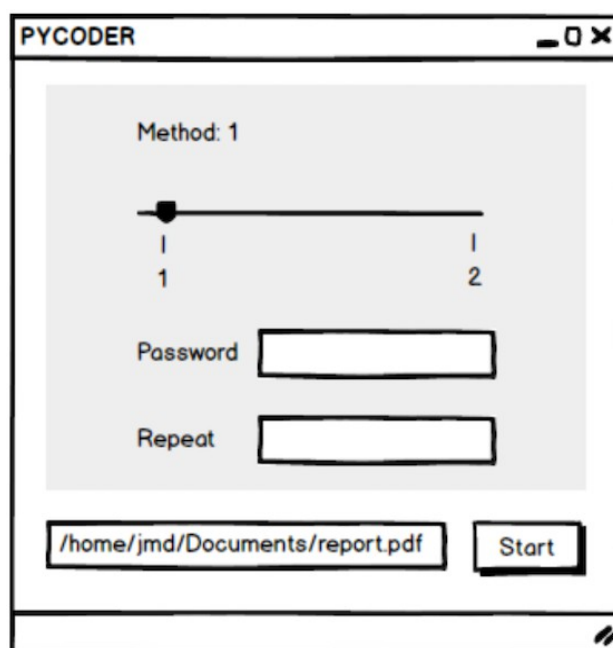
Após descomprimir o ficheiro `info.pdf.rle` (por exemplo) para `info.pdf` na linha de comandos, o pycoder exibe:

```
Decompressed 'info.pdf.rle' into 'info.pdf' using method 2 (opcode 138)
Compression date/time      : 2020-11-15 19:21
```

INTERFACE GRÁFICA

Quando invocado sem opções, o PyCoder deve abrir uma interface gráfica (GUI) desenvolvida com uma das seguintes *frameworks*: PySide2/6 (Qt for Python), PyQt5/6, PySimpleGUI ou Kivy.

Eis a maquete da GUI do PyCoder:



A operação a executar - comprimir ou descomprimir - através de `Start` depende da extensão do caminho indicado no campo apropriado.

OUTRAS CONSIDERAÇÕES

Deve desenvolver o código num ambiente virtual criado com o módulo `venv`. Aqui é que deve instalar todas as bibliotecas de que necessita para a sua aplicação.

Além do mais deve criar um repositório GIT para gestão de versões. Pode criar um repositório local ou utilizar um serviço de alojamento (*cloud*) destes repositórios como o GitHub.

Deve ser possível produzir uma distribuição executável do PyCoder (`pycoder.exe` em Windows, `pycoder` apenas noutros SOs).

Consultar: https://en.wikipedia.org/wiki/Run-length_encoding
<https://www.pythonguis.com/>
<https://doc.qt.io/qtforpython/quickstart.html>
<https://www.pythonguis.com/pyside2-tutorial/>
<https://realpython.com/python-virtual-environments-a-primer/>
<http://docopt.org/>

[PARTE III - SYMS]

Vamos agora fazer um programa **extra** para detectar ficheiros com "semelhanças". Através da linha de comandos, o seu programa recebe um caminho para uma directoria e "desce" essa directoria procurando por ficheiros que apresentem uma determinada semelhança. No final exibe uma listagem com os grupos de ficheiros semelhantes.

De seguida, precedidas da respectiva opção da linha de comandos, indicamos as semelhanças a detectar:

- `-c, --contents`: detecta ficheiros com conteúdo binário igual
- `-n, --name`: detecta ficheiros com o mesmo nome (incluindo a extensão)
- `-e, --extension`: detecta ficheiros com a mesma extensão
- `-r PATTERN, --regex==PATTERN`: detecta ficheiros cujo o nome verifica a expressão regular dada por `PATTERN`; os ficheiros são depois agrupados pela ocorrência concreta do padrão

A sintaxe do comando deve ser a seguinte:

```
$ symps [-c] [-n] [-e] [-r PATTERN] [caminho_dir]
```

Utilize a biblioteca **argparse** para leitura das opções da linha de comandos. Se mais do que uma opção for especificada, então o seu programa deve exibir uma listagem por opção. Se nenhuma opção for especificada, o programa assume a opção `-n`. O valor por omissão para `caminho_dir` é a directoria corrente.

Sugestões:

1. Utilize `os.walk` para percorrer árvore de directorias dentro do caminho fornecido
2. No caso da detecção de ficheiros com o mesmo conteúdo, para evitar ter que comparar ficheiros, processo dispendioso em termos de CPU e de utilização de memória, utilize `hashlib.md5` para obter de forma eficiente um resumo de cada ficheiro. É praticamente impossível encontrar dois ficheiros para os quais se obtenham resumos idênticos (ou seja, nestas circunstâncias, e para efeitos práticos, um *hash* é um resumo unívoco).

Consultar: <https://docs.python.org/3/library/os.html?#os.walk>
<https://docs.python.org/3/library/hashlib.html#module-hashlib>
https://www.tutorialspoint.com/python/python_reg_expressions.htm
<https://docs.python.org/3/howto/regex.html>
<https://towardsdatascience.com/learn-enough-python-to-be-useful-argparse-e482e1764e05>
<https://docs.python.org/3/library/argparse.html>

AVALIAÇÃO

Devem entregar até às 23h59 do dia **25/07/2022** o que está descrito na secção **INTERFACE DA LINHA DE COMANDOS**. Para tal, devem submeter um **ZIP** com todo o projecto. Este **ZIP** deve ser tirado do repositório de GitHub que criaram para o efeito. Indiquem no ficheiro **README.MD** o endereço do repositório bem como os nomes dos membros do grupo, caso tenham trabalhado em grupo.

Serão dados pontos extra a quem desenvolver o programa **SYMS**.

NOTAS

Aqui ficam algumas noções relacionados com esta temática, em inglês, por falta de tempo para traduzir, mas também para fortalecer o nosso inglês técnico.

Encoding	<i>Provide a 'physical' representation for data. This can be motivated by the fact that there is no such representation available yet. But it can also be done in order to obtain different representations for the same data with the purpose of gaining efficiency in processing the data, or reducing the size (compression), or "shield it" from interference when sending it through a communication channel, or obscuring the data (encryption), etc.</i>
Encryption	<i>The process of encoding (obscuring) data to make it unreadable.</i>
Decryption	<i>The process of decoding data to make it readable again.</i>
Cipher	<i>Encoding algorithm for performing encryption and decryption.</i>

Plaintext	<i>The original data.</i>
Ciphertext	<i>The encrypted data.</i>
Compression	<i>An encoding process which aims to eliminate redundant information allowing for the original data to be compressed.</i>

Observe que todos os codificadores podem possuir uma interface comum assente em dois métodos: `encode` para codificar os dados, e `decode` para decodificar os dados. Uma das vantagens de definir uma interface passa pela possibilidade de substituir um codificador por outro sem que o código cliente tenha que ser alterado.