Universidade do Minho
Licenciatura em Física

Tarefa 7

Professores:	Cristiana Neto, Pedro Oliveira, Vitor Alves
Disciplina:	Linguagens para Computação Numérica
Tema:	Ficheiros
Data:	Abril de 2022

1 Introdução

No desenvolvimento de programas informáticos na maioria das vezes é necessário importar e guardar dados (por exemplo, resultados ou amostras) de ou para ficheiros. Além de possibilitar a reutilização dos dados processados permite a portabilidade dos mesmos, para utilização noutros programas. Atualmente, existe uma variedade de formatos padronizados para este propósito, por exemplo o formato xml (definido para guardar informação de forma estruturada), csv, JSON, etc... A forma mais simples de guardar informação de modo legível para o ser humano é através de ficheiros de texto. Neste formato, é possível estruturar informação através de delimitadores ou espaçamentos.

1.1 Conceitos Básicos

O Python tem funções nativas que permitem abrir, ler, manipular e escrever sobre ficheiros. No entanto existe uma função principal `open()` que guarda uma referência a um objecto do tipo arquivo, sobre o qual as restantes funções podem ser utilizadas. A função `open()` define-se do seguinte modo:

```
open('ficheiro','modo')
```

O campo "modo"pode receber os parâmetros apresentados na Tabela 1. 0 campo "ficheiro"vai depender da localização do ficheiro que pretender abrir em relação ao script que estiver a correr. Se o ficheiro estiver no mesmo diretório do script, apenas é necessario descrever o nome do ficheiro, caso esteja num diretório diferente tem que colocar o caminho absoluto para este. Isto vai depender do sistema operativo utilizado. Por exemplo, se tiver um ficheiro com nome "resultados.txt"num diretório diferente do script, então terá que definir do seguinte modo:

```
open('C:\\caminho para o ficheiro\\resultados.txt','r')
```

Poderá encontrar com mais detalhe métodos para manipulação de caminhos para directórios ou ficheiros na documentação do Python.

Modo	Descrição
r	Abre um ficheiro apenas em modo de leitura
r+	Abre um ficheiro em modo de leitura e de escrita. O apontador inicia no início do ficheiro.
w	Abre um ficheiro apenas em modo de escrita.
w+	Abre um ficheiro em modo de escrita e de leitura. Sobre-grava um ficheiro existente, ou cria um novo caso não exista.
a	Abre um ficheiro para anexação. O apontador inicia no final do ficheiro.
a+	Abre um ficheiro para leitura e anexação. Caso já exista o ficheiro o apontador inicia no final, caso não exista cria um novo em modo de leitura e escrita.

Tabela 1: *Modos da função `open()`*

Após ler ou escrever ficheiros existe uma funcao `close()` para fechar esses mesmos ficheiros. É importante aplicar esta função após uma tarefa de leitura ou escrita porque enquanto o ficheiro estiver aberto, está desnecessariamente a ocupar recursos do sistema, principalmente memória. A função é utilizada do seguinte modo:

```
ficheiro.close()
```

1.2 Leitura de ficheiros

O processo de ler um ficheiro é dividido em três etapas:

1. Abrir um ficheiro com o método `open()`, descrito em cima.
2. Ler o ficheiro com um dos métodos descritos no seguimento desta secção;
3. Fechar o ficheiro com o método `close()`.

Existem três métodos distintos que poderão ser utilizados na leitura do ficheiro:

- `read()` - Retorna todo o conteúdo do ficheiro como uma única string. Este metodo incorpora um parâmetro `size`, com o qual se pode definir a quantidade de dados que se quer ler. Quando este parâmetro é omitido é lido todo o conteúdo do ficheiro e colocado em memória.
- `readline()` - Retorna uma linha completa incluindo os caracteres (\n) que indicam uma nova linha.
- `readlines()` - Retorna uma lista com todas as linhas presentes no ficheiro. Cada linha corresponde a um elemento da lista. Tal como o método `read()` o todo conteúdo do ficheiro é colocado em memória.

Após abrir um ficheiro com o método `open()` pode-se então ler o seu conteúdo, como mostrado no seguint exemplo (supondo que se tem um ficheiro com o nome "frase"e extensão .txt):

```
#Por exemplo, escreva num ficheiro com o nome "frases.txt" o seguinte texto (sem as aspas):
# "Vou abrir esta frase no Python."
# "Esta é a segunda linha."
# Nota: coloque o ficheiro no mesmo diretório onde escreveu o seu script/notebook

#abre o ficheiro
ficheiro = open('frases.txt','r')
#lê o ficheiro e imprime
print(ficheiro.read())
#fecha o ficheiro
ficheiro.close()
```

```
Vou abrir esta frase no Python.
Esta é a segunda linha.
```

No entanto o utilizador pode querer ler apenas uma ou um determinado número de linhas presentes no ficheiro. Assim pode utilizar um dos métodos descritos em cima, `readline()` ou `readlines()`. Se aplicar estes dois métodos no exemplo anterior, poderá ver quais as diferenças entre eles método `read()`, como exemplificado a seguir:

```
[2] ficheiro = open('frases.txt','r')
    print(ficheiro.readline())
    ficheiro.close()
```

```
Vou abrir esta frase no Python.
```

```
[4] ficheiro = open('frases.txt','r')
    print(ficheiro.readlines())
    ficheiro.close()
```

```
['Vou abrir esta frase no Python.\n', 'Esta é a segunda linha.']
```

Por exemplo se quiser ler a segunda linha do ficheiro, pode fazer da seguinte forma:

```
[5] # imprime a segunda linha presente no ficheiro
    # 1- As linhas sao colocadas num objecto lista
    # 2- imprimi-se o elemento(linha) no index 1 correspondente à segunda linha

    ficheiro = open('frases.txt','r')
    print(ficheiro.readlines()[1])
    ficheiro.close()
```

```
Esta é a segunda linha.
```

Os métodos mencionados para abrir e fechar ficheiros não são totalmente seguros pois se uma exceção ocorrer quando alguma operação estiver a ser feita com o ficheiro, o código vai sair sem fechar o ficheiro. Uma maneira mais segura seria usar um bloco `try... finally`.

```
[6] try:
      #Colocar aqui as instruções
      #Se ocorrer uma exceção executar esta instrução pode não ser executada.
       ficheiro = open('frases.txt','r')
       print(ficheiro.read())

    finally:
      #Esta instrução será sempre executada.
      ficheiro.close()
```

Contudo, a melhor maneira de fechar um ficheiro é usando o `with`. Este garante que os ficheiros são fechados quando o bloco dentro do `with` termina, ou seja, não é necessário chamar explicitamente o método `close()` pois este é chamado internamente.

```
[7] with open('frases.txt','r') as ficheiro:
       print(ficheiro.read())
```

1.2.1 Aplicação de ciclos na leitura de ficheiros

Quando se esta a trabalhar com ficheiros que contêm muita informação, é necessário aceder a essa informação de forma eficiente, gastando o mínimo de recursos computacionais. Ao aplicar ciclos `for` e `while` na leitura, o utilizador poderá reduzir a quantidade de informação que tem de ser guardada em memória, retirando apenas a informação essencial. Existem métodos para melhorar a eficiência na leitura de ficheiros e caso queira aprofundar esta matéria pode ler a documentação do Python. Com os ciclos poderá fazer a leitura de todo o arquivo ou parte do arquivo de um modo mais controlado. Por exemplo ao aplicar os métodos `readline()` com recurso a ciclos, o utilizador pode ler apenas uma linha ou linhas pretendidas. Nos exemplos seguintes são demonstradas algumas formas de aplicar os ciclos na leitura de ficheiros. Supondo que tem um ficheiro de texto com o nome "linhas.txt"no qual o seu conteúdo é:

```
linha1
linha2
...
...
linha9
```

Pode-se então imprimir o conteúdo deste ficheiro de acordo com alguns dos exemplos seguintes:

Ciclo for

```
[ ]  # Utilização direta.
     with open('linhas.txt','r') as ficheiro:
       for linha in ficheiro:
         print(linha)

     #método readlines com cliclo for
     with open('linhas.txt','r') as ficheiro:
       linhas = ficheiro.readlines()
       for linha in linhas:
         print(linha)

     #seleção de linhas
     with open('linhas.txt','r') as ficheiro:
       linhas = ficheiro.readlines()
       for i in range(2,5):
         print(linhas[i])
```

Ciclo while

```
[4]  #Utilização do método readline() com o ciclo while e condicional if.
     with open('linhas.txt','r') as ficheiro:
       while 1:
         linha = ficheiro.readline()
         if not linha:
           break
         print(linha)

     #Utilização do método readline() com ciclo while.
     with open('linhas.txt','r') as ficheiro:
       linha = ficheiro.readline()
       while linha:
         print(linha)
         linha = ficheiro.readline()

     #Utilização do método readline() com o ciclo while e condição de desigualdade.
     with open('linhas.txt','r') as ficheiro:
       linha = ficheiro.readline()
       while linha != '':
         print(linha)
         linha = ficheiro.readline()
```

1.3 Escrita em ficheiros

Os procedimentos para escrita em ficheiros é muito semelhante aos de leitura porém, o passo intermédio é diferente. Os passos são os seguintes:

1. Abrir um ficheiro com o método `open()`, semelhante à leitura de ficheiros, porém deverá ser em modo de escrita (w). Caso já tenha um ficheiro com conteúdo e queira apenas adicionar informação, devera abrir em modo de anexação (a).

2. Escrever os dados pretendidos para o ficheiro através dos métodos descritos de seguida.

3. Fechar o ficheiro com o método `close()` descrito anteriormente (ou usar o `with`).

Ao contrário do processo de leitura, existem apenas dois métodos para escrita no ficheiro:

- `write()` - Escreve uma string num ficheiro (grava apenas texto como uma única string). Este método não pode gravar objetos iteráveis (por exemplo: listas).
- `writelines()` - Escreve uma sequência de strings para um ficheiro, porém a diferenca, é que este método pode escrever objetos iteráveis (listas).

Supondo que se quer escrever um determinado ficheiro com o nome 'omeuficheiro.txt' contendo a seguinte frase "Este é o meu primeiro programa para escrever num ficheiro", então terá de proceder como exemplificado a seguir:

```
[2] try:
      #Colocar aqui as instruções
      #Se ocorrer uma exceção executar esta instrução pode não ser executada.
       ficheiro = open('omeuficheiro.txt','w')
       ficheiro.write('Primeiro programa de escrita em ficheiros.')
    finally:
      #Esta instrução será sempre executada.
      ficheiro.close()

    #alternativa com with
    with open('omeuficheiro.txt','w') as ficheiro:
      ficheiro.write('Primeiro programa de escrita em ficheiros.')
```

Mas caso o utilizador quisesse guardar o ficheiro num diretório específico (por exemplo no disco C do Windows) então teria que alterar o método `open()` para:

```
1 ficheiro = open('C:\\o_meu_ficheiro.txt','w')
```

Supondo agora que a frase está dividida por palavras numa lista, e quer guardar cada uma das palavras por cada linha, então poderá fazer:

```
[5] # Criar um ojecto ficheiro
    with open('o_meu_ficheiro.txt','w') as ficheiro:
      palavras=['Este\n','é\n','o\n','meu\n','primeiro\n','programa\n','para\n','escrever']
      ficheiro.writelines(palavras) #cada elemento da lista fica uma linha no ficheiro
```

1.3.1 Aplicação de ciclos na escrita de ficheiros

Tal como na leitura de ficheiros a aplicação de ciclos na escrita de ficheiros é essencial. Deste modo é possível organizar a informação de acordo com uma formatação e estrutura que o utilizador pretender. A seguir são expostos vários exemplos para realçar a importância de ciclos na escrita de ficheiros.

```
[8] matriz = [[3,5,6,8],[5,9,2,3],[6,9,10,1]]

    with open("matriz.txt","w") as ficheiro:
      for linha in matriz:
        for coluna in linha:
          ficheiro.write(str(coluna)+" ")
        ficheiro.write("\n")
```

```
[21] #Adicionar dados de uma matriz 2 a uma matriz 1 já presente num ficheiro.
     matriz2 = [[1,2,9,8],[2,4,2,2]]

     with open("matriz.txt","a") as ficheiro:
       for linha in matriz2:
         for coluna in linha:
           ficheiro.write(str(coluna)+"\t")
         ficheiro.write("\n")
```

2 Exercícios

Com esta ficha de trabalho pretende-se apresentar alguns problemas exemplificativos da utilização de ficheiros.

1. Ler o ficheiro nomes.txt em formato texto, com uma lista de nomes não ordenados e criar um novo ficheiro com esses mesmos nomes ordenados por ordem alfabética.

2. Ler o ficheiro programação.txt em formato texto e criar um novo ficheiro com uma lista de todas as palavras e número de ocorrências de cada uma.

3. Escreva um programa que leia uma sequência de valores numéricos de um ficheiro (um valor por linha) e que os escreva num outro ficheiro (um por linha) de forma ordenada (do mais pequeno para o maior).

4. Escreva um programa que grave num ficheiro uma matriz 10x10 com a diagonal preenchida com o valor 1 e o restante com o valor 0.

5. O ficheiro sistema_solar.txt apresenta informações sobre a distância média do sol, a massa e o raio de uma seleção de corpos celestiais do nosso sistema solar.

   (a) Leia os conteúdos do ficheiro e escreva os conteúdos em três dicionários: `distancia`, `massa` e `raio` com o nome dos objetos celestiais como key.

   (b) Crie um novo dicionário, `densidades`, que contém a densidade de cada objeto celestial. A densidade é a massa a dividir pelo volume em kg/m^3. Atenção que o raio no ficheiro não está em metros. Podemos assumir que todos objetos são esferas perfeitas, o que significa que os seus volumes podem ser calculados através da formula $v = \frac{4}{3} \times \pi \times r^3$.

   (c) Escreva todas as informações, incluindo as densidades, num novo ficheiro com mais ou menos o mesmo formato que o original. O programa da alínea a) também deve ser capaz de abrir este ficheiro.

6. Quando trabalhamos com o valor da aceleração da gravidade é comum desprezarmos o local da Terra onde nos encontramos. Na verdade, o valor de g depende de onde estamos localizados, mais precisamente da altitude e latitude do local. O ficheiro valores_g.txt apresenta os valores arredondados de g numa seleção de cidades.

   Escreva um programa que leia o ficheiro e crie um dicionário onde será guardado o g associado a cada cidade. Por exemplo, a aceleração da gravidade em Estolcomo é $g = 9.818 m/s^2$. Ou seja, o dicionário deve ser feito de modo a que s mandarmos Estocolmo como key, o dicionário deve retornar 9.818.

   Atenção que as cidades podem ter mais que um nome, portanto é aconselhado o uso do `split()` e posterior verificação da lista obtida (se o comprimento da lista for mais que

dois, quer dizer que a cidade tem mais de uma palavra). Por fim, pretende-se que o utilizador insira um cidade e a massa de um objeto e obtenha a força gravítica calculada com o respetivo valor de g.