UFCD 10793 - Fundamentos de Python

Notebook 08 - Ficheiros em Python

Sílvia Martins

AEFHP 2022

Leitura e escrita em ficheiros

É particularmente útil, em muitas situações, criar, ler e/ou escrever em ficheiros. O Python disponibiliza formas expeditas para tal.

Criar um novo ficheiro

Para trabalhar com ficheiros em Python, utiliza-se o método *open()* , que pode ter vários parâmetros (que podes consultar), sendo que se devem explicitar dois : O nome do ficheiro e o modo de abertura do mesmo (strings).

Sintaxe:

```
f = open('<filename>', '<mode>')
```

f será o nome que apontará para o ficheiro e que é defenido pelo utilizador.

Métodos (modos) de abertura de um ficheiro:

- `"r"` - Leitura - é a forma de abertura padrão (caso não seja indicado o modo de abertura será usada esta forma).
- `"x"` - Criar - Cria um ficheiro, devolve um erro se o ficheiro especificado existir.
- `"a"` - acrescentar (Append) - Abre um ficheiro para adição no final do mesmo. Cria o ficheiro especificado se este não existir.
- `"w"` - Escrever (Write) - Abre um ficheiro para escrita. Cria o ficheiro especificado se este não existir.

Pode-se também definir se o ficheiro será binário ou de texto:

- `"t"` - Text - Tipo padrão. Modo de texto
- `"b"` - Binary - Binary mode (x.plo. imagens)

Para ler dados de um ficheiro usa-se o método *read()* e para escrever dados no ficheiro o método *write()*
Associada a uma instrução de abertura de ficheiro deve-se sempre escrever-se uma instrução de fecho do mesmo, o que pode ser feito com o método *close()* .

In [1]:

```
%pwd
```

Out[1]:

```
'C:\\Users\\Sílvia Martins\\Desktop\\notebook_s'
```

Para criar um ficheiro de texto nesta localização, basta fazer o seguinte (daria um erro se o ficheiro já existisse):

In [2]:

```
f=open("exemplo.txt","x")
```

In [ ]:

```
%ls
```

In [3]:

```
type(f)
```

Out[3]:

```
_io.TextIOWrapper
```

In [4]:

```
f.close()
```

Depois de criado pode-se proceder à sua abertura para escrita:

In [5]:

```
f = open("exemplo.txt", "w")
f.write("Bolas! Gravei por cima do conteúdo!")
f.close()
```

Reabrindo o ficheiro para leitura e apresentação do seu conteúdo:

In [6]:

```
f = open("exemplo.txt", "r")
print(f.read()) # a leitura é feita de uma só vez
f.close()
```

```
Bolas! Gravei por cima do conteúdo!
```

Abertura do ficheiro para **acrescentar dados a partir do fim** do mesmo:

In [7]:

```
f = open("exemplo.txt", "a")
f.write("\nbom dia\nboa tarde\nboa noite")
f.close()
```

Abertura do ficheiro e **leitura de todo o seu conteúdo**:

In [8]:

```
f = open("exemplo.txt", "r")
print(f.read())
```

```
Bolas! Gravei por cima do conteúdo!
bom dia
boa tarde
boa noite
```

In [9]:

```
print(f.read()) # como a marca de ficheiro se encontra no fim do ficheiro, nada mais há par
```

In [10]:

```
f.close()
```

Abertura do ficheiro para **leitura linha a linha** usando o método *readline()*

In [11]:

```
f = open('exemplo.txt', 'r')
print(f.readline())  #que devolve a 1.ª linha do ficheiro
print(f.readline(3)) #que devolve os 3 primeiros carateres da 2.ª linha do ficheiro
f.close()
```

```
Bolas! Gravei por cima do conteúdo!

bom
```

Para a leitura de **todas as linhas do ficheiro** usando o método *readlines()*

In [12]:

```
f = open('exemplo.txt', 'r')
print(f.readlines())
f.close()
```

```
['Bolas! Gravei por cima do conteúdo!\n', 'bom dia\n', 'boa tarde\n', 'boa n
oite']
```

Para que o caráter \n não apareça na impressão podemos eliminá-los usando o método *strip()* , aplicado a

strings

In [13]:

```
with open('exemplo.txt') as f:
    seq = f.readlines()

seq = [linha.strip() for linha in seq]
print(seq)
f.close()
```

```
['Bolas! Gravei por cima do conteúdo!', 'bom dia', 'boa tarde', 'boa noite']
```

Para percorrer o ficheiro linha por linha, com um ciclo for:

In [14]:

```
f = open("exemplo.txt")
for x in f:
    print(x.strip())
f.close()
```

```
Bolas! Gravei por cima do conteúdo!
bom dia
boa tarde
boa noite
```

Com o método *enumerate()* são gerados os pares de valor- número da linha e conteúdo da linha:

In [15]:

```
f = open("exemplo.txt")
for linha, x in enumerate(f):
    print(linha,": ", x.strip())
f.close()
```

```
0 :  Bolas! Gravei por cima do conteúdo!
1 :  bom dia
2 :  boa tarde
3 :  boa noite
```

Para eliminar um ficheiro é necessário importar o módulo do SO e executar o método *os.remove()*

In [16]:

```
import os
os.remove("exemplo.txt")
```

Para evitar erros deve-se verificar se o ficheiro existe antes de tentar a eliminação:

In [17]:

```
import os
if os.path.exists("exemplo.txt"):
  os.remove("exemplo.txt")
else:
  print("O ficheiro não existe")
```

```
O ficheiro não existe
```

Pode-se também criar uma pasta ou removê-la:

In [ ]:

```
import os
os.mkdir("minha_pasta")
os.rmdir("minha_pasta")
```

Leitura e escrita em ficheiros csv

Leitura de ficheiros csv

A função *reader()* permite ler o devolver um objeto `\_csv.reader` que pode ser usado para iterar sobre o conteúdo de um arquivo CSV. A sintaxe da função *reader()* é a seguinte:

reader(fileobj [, dialect='excel' [, **fmtparam]]) -> _csv.reader

- `fileobj` - Refere-se ao objeto ficheiro (obrigatório)
- `dialect` - refere-se aos diferentes modos de formatar o documento CSV. Por padrão, o módulo csv usa o mesmo formato do Microsoft Excel.
- `fmtparam` - Refere-se ao conjunto de argumentos de palavras-chave para personalizar o dialeto como skipinitialspace (eliminação de espaços) ou lineterminator (que se refere ao carater usado para terminar uma linha, o padrão é `\r\n` .

reader é o objeto leitor.

Ler os dados de um ficheiro csv:

In [18]:

```
import csv
with open('empregados.csv') as f:
    for x in csv.reader(f, skipinitialspace=True):
        print(x)
```

```
['Nome', 'idade', 'cargo', 'departamento', 'salario']
['Ana', '34', 'formador', 'A', '1100']
['Rui', '45', 'cozinheiro', 'C', '900']
```

Observa que cada linha no ficheiro CSV é retornada como uma lista de strings.
Para obter os dados de determinados campos, podemos usar a indexação. Por exemplo:

In [19]:

```
import csv
with open('empregados.csv', 'r') as f:
    for x in csv.reader(f):
        print(x[0], x[1], x[2], x[3], x[4])
```

```
Nome idade cargo departamento salario
Ana 34  formador  A  1100
Rui  45  cozinheiro  C  900
```

Para que não seja impressa a linha de cabeçalho, com os campos,podemos usar o método *next*

In [20]:

```
import csv

with open('empregados.csv', 'rt') as f:
    csv_r=csv.reader(f)
    next(csv_r) # passa à linha seguinte
    for x in csv_r:
        print(x[0], x[1], x[2])
```

```
Ana 34  formador
Rui  45  cozinheiro
```

In [21]:

```
csv.reader?
```

Escrita em ficheiros csv

A função *writer()* permite escrever dados num ficheiro CSV e devolver um objeto _csv.writer . A sintaxe da função *writer()* é a seguinte:

writer(fileobj [, dialect='excel' [, **fmtparam]]) -> _csv.writer

- fileobj - Refere-se ao objeto ficheiro (obrigatório)
- dialect - refere-se aos diferentes modos de formatar o documento CSV. Por padrão, o módulo csv usa o mesmo formato do Microsoft Excel.
- fmtparam - Refere-se ao conjunto de argumentos de palavras-chave para personalizar o dialeto funcionando da mesma forma que para a função *reader()*. *writer* é o objeto gravador.

A instância do *writer* fornece os dois métodos a seguir para gravar dados:

- *writerow(row)* grava uma linha de dados e devolve o nº de carateres gravados no ficheiro.A linha tem de ser uma sequência de carateres.
- *writerows(row) _grava múltiplas linhas de dados e devolve _ none* . As linhas devem ser sequências.

In [22]:

```
import csv
f = open('numero_dobro_triplo.csv', 'w', newline='') # '' faz com que as linhas sejam grava
w = csv.writer(f)
# grava as 5 linhas inseridas sob a forma de lista
for i in range(5):
  w.writerow([i, i*2, i*3])
f.close()

#ler o ficheiro e exibi-lo no ecrã
f=open('numero_dobro_triplo.csv')
t=[]
r=csv.reader(f)
for x in r:
    t.append(x)
    print(x)

print(len(t))
```

```
['0', '0', '0']
['1', '2', '3']
['2', '4', '6']
['3', '6', '9']
['4', '8', '12']
5
```

In [23]:

```
campos = ['id', 'nome', 'email']
linhas = [
    [1, 'Ana', 'anouska@gmail.com'],
    [2, 'Watson', 'sherlock221b@iol.pt'],
    [3, 'Sara', 'sarinha@gmail.com'],
    [4, 'Carlos', 'reicharles@gmail.com'],
]
f=open('clientes.csv', 'wt')
x = csv.writer(f)
x.writerow(campos)  # grava os campos na 1.ª linha
x.writerows(linhas) # grava todas as linhas de dados dos clientes
f.close()
```

In [24]:

```
import csv
with open('clientes.csv','rt') as f:
    x= csv.reader(f, skipinitialspace=True)
    for linha in x:
        if linha !=[]:
            print(linha)
```

```
['id', 'nome', 'email']
['1', 'Ana', 'anouska@gmail.com']
['2', 'Watson', 'sherlock221b@iol.pt']
['3', 'Sara', 'sarinha@gmail.com']
['4', 'Carlos', 'reicharles@gmail.com']
```

Métodos DictReader e DictWriter

Estes métodos funcionam quase exatamente como reader() e writer(), mas em vez de reajustar uma linha como uma lista, é devolvido um dicionário ou são gravados dados usando um dicionário.

Método dictReader()

Syntaxe:

```
DictReader(fileobj, fieldnames=None, restkey=None, restval=None, dialect='excel', **fmtparam)
```

- `fieldnames` - (opcional) Refere-se à lista de chaves que serão usadas no dicionário a devolver. Se omisso, serão considerados para campos , a 1.ª linha do ficheiro csv.
- `restkey` - (opcional) Se a linha tiver mais campos do que o especificado no parâmetro fieldnames, os campos restantes serão armazenados como uma sequência codificada pelo valor do argumento restkey.
- `restval` - (opcional) Fornece valores aos campos que estão em falta na entrada. Os restantes parâmetros funcional de forma idêntica como para os métodos reader e writer.

Método dictWriter()

Syntaxe:

```
Dictwriter(fileobj, fieldnames, restval='', extrasaction='raise',  dialect='excel', **fmtparam)
```

- `fieldnames` - Refere-se aos nomes dos campos e à ordem em que serão escritos no ficheiro.
- `restval` - Fornece o valor ausente para as chaves que não existem no dicionário.
- `extrasaction` - Controla a ação a tomar se o dicionário contiver uma chave que não seja encontrada no argumento fieldnames. Por padrão, extrasaction é definido para aumentar, o que significa que uma exceção será gerada em tal evento. Caso de ignorem os valores extras, define-se extrasaction para ignorar.

Fornece os seguintes métodos para gravar dados:

- *`writeheader()`* - Grava o cabeçalho (ou seja, nomes de campo) no arquivo CSV e retorna none (nada)
- *`writerow(row)`* e *`writerows(rows)`* que têm a mesma função que para o *Writer()*

Nota: para mais informação consulta o bibliografia no final do notebook.

Escrita de dados no ficheiro usando *`dictwriter()`* :

In [25]:

```
import csv

header = ['id', 'name', 'address', 'zip']
#criação do dicionário
rows = [
    {'id': 1, 'name': 'Hannah', 'address': '4891 Black Street, Anchorage, Alaska', 'zip': 9
    {'id': 2, 'name': 'Sarah', 'address': '4223 Half  Drive, Lemoore, California', 'zip': 9
    {'id': 3, 'name': 'Sam', 'address': '3952 Little Street, Akron, Ohio', 'zip': 93700 },
    {'id': 4, 'name': 'Chris', 'address': '3192 Flint Road, Arlington Heights, Illinois', '
    {'id': 5, 'name': 'Doug', 'address': '3236 Walkers Way, Burr Ridge', 'zip': 61255},
]

with open('dictcustomers.csv', 'wt') as f:
    csv_writer = csv.DictWriter(f, fieldnames=header)
    csv_writer.writeheader() # write header
    csv_writer.writerows(rows)
```

Leitura de dados do ficheiro usando *dictreader()*:

In [26]:

```
import csv
with open('dictcustomers.csv', 'r') as f:
    csv_reader = csv.DictReader(f)
    for line in csv_reader:
        print(line)
```

```
{'id': '1', 'name': 'Hannah', 'address': '4891 Black Street, Anchorage, Alas
ka', 'zip': '99500'}
{'id': '2', 'name': 'Sarah', 'address': '4223 Half  Drive, Lemoore, Californ
ia', 'zip': '97400'}
{'id': '3', 'name': 'Sam', 'address': '3952 Little Street, Akron, Ohio', 'zi
p': '93700'}
{'id': '4', 'name': 'Chris', 'address': '3192 Flint Road, Arlington Heights,
Illinois', 'zip': '62777'}
{'id': '5', 'name': 'Doug', 'address': '3236 Walkers Way, Burr Ridge', 'zi
p': '61255'}
```

Utilização do parâmetro *restkey* :

In [27]:

```
import csv
with open('dictcustomers.csv', 'rt') as f:

    fields = ['id','name',]
    csv_reader = csv.DictReader(f, fieldnames=fields, restkey='extra', escapechar='\\')
    for row in csv_reader:
        print(row)
```

```
{'id': 'id', 'name': 'name', 'extra': ['address', 'zip']}
{'id': '1', 'name': 'Hannah', 'extra': ['4891 Black Street, Anchorage, Alask
a', '99500']}
{'id': '2', 'name': 'Sarah', 'extra': ['4223 Half  Drive, Lemoore, Californi
a', '97400']}
{'id': '3', 'name': 'Sam', 'extra': ['3952 Little Street, Akron, Ohio', '937
00']}
{'id': '4', 'name': 'Chris', 'extra': ['3192 Flint Road, Arlington Heights,
Illinois', '62777']}
{'id': '5', 'name': 'Doug', 'extra': ['3236 Walkers Way, Burr Ridge', '6125
5']}
```

Usar o parâmetro restval:

In [28]:

```
import csv
with open('dictcustomers.csv', 'rt') as f:
    fields = ['id','name', 'address', 'zip', 'phone', 'email'] # dois campos extra
    csv_reader = csv.DictReader(f, fieldnames=fields, restkey='extra', restval='NA')
    for row in csv_reader:
        print(row)
```

```
{'id': 'id', 'name': 'name', 'address': 'address', 'zip': 'zip', 'phone': 'N
A', 'email': 'NA'}
{'id': '1', 'name': 'Hannah', 'address': '4891 Black Street, Anchorage, Alas
ka', 'zip': '99500', 'phone': 'NA', 'email': 'NA'}
{'id': '2', 'name': 'Sarah', 'address': '4223 Half  Drive, Lemoore, Californ
ia', 'zip': '97400', 'phone': 'NA', 'email': 'NA'}
{'id': '3', 'name': 'Sam', 'address': '3952 Little Street, Akron, Ohio', 'zi
p': '93700', 'phone': 'NA', 'email': 'NA'}
{'id': '4', 'name': 'Chris', 'address': '3192 Flint Road, Arlington Heights,
Illinois', 'zip': '62777', 'phone': 'NA', 'email': 'NA'}
{'id': '5', 'name': 'Doug', 'address': '3236 Walkers Way, Burr Ridge', 'zi
p': '61255', 'phone': 'NA', 'email': 'NA'}
```

Bibliografia

Caleiro.C., Ramos.J.(2016). Notebook 01 - Conceitos básicos da linguagem Python Dep. Matemática - IST.Lisboa:IST https://www.w3schools.com/python/python_file_handling.asp (https://www.w3schools.com/python/python_file_handling.asp) https://webpages.ciencias.ulisboa.pt/~aeferreira/python/files.html (https://webpages.ciencias.ulisboa.pt/~aeferreira/python/files.html)

Carvalho. Adelaide. (2021). Práticas de Python - Algoritmia e programação. Lisboa: FCA

https://dicasdepython.com.br/como-criar-um-arquivo-csv-no-python/ (https://dicasdepython.com.br/como-criar-

um-arquivo-csv-no-python/)
https://thepythonguru.com/python-how-to-read-and-write-csv-files/ (https://thepythonguru.com/python-how-to-read-and-write-csv-files/)