PENSAMENTO COMPUTACIONAL

Cleiton Silvano Goulart

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS

Leitura de arquivos em Python

Objetivos de aprendizagem

Ao final deste texto, você deve apresentar os seguintes aprendizados:

- Explicar a leitura de arquivos no formato txt nos diferentes modos de abertura.
- Definir a leitura de arquivos no formato csv nos diferentes modos de abertura.
- Ler arquivos em diferentes problemas computacionais.

Introdução

Neste capítulo, você vai estudar os modos de abertura utilizados para abrir um arquivo na linguagem Python. Você vai verificar como abrir e ler arquivos no formato txt e arquivos separados por vírgula, no formato csv. Os arquivos csv são um formato muito versátil e simples para a troca de informações entre diferentes programas e equipamentos de automação industrial. Por fim, você vai analisar a leitura de arquivos em diferentes problemas computacionais.

Arquivos de texto puro (txt)

O computador armazena todas as informações na forma de arquivos. Os **arquivos** consistem em uma sequência de caracteres, que são armazenados nas mídias convencionais dos computadores, isto é, no disco rígido, no *pen-drive*, nos discos ópticos (CD-ROM, DVD-ROM, Blu-Ray), etc. Esses arquivos servem para armazenar diferentes tipos de informações.

Em primeira instância, os arquivos podem ser divididos em arquivos de texto e arquivos binários. Quando nos referimos aos **arquivos de texto**, queremos dizer que o conteúdo desse arquivo é composto por uma sequência de caracteres legíveis em algum idioma, como um texto de um livro. Já os **arquivos binários** são compostos por uma sequência de caracteres que não são facilmente compreensíveis. Os arquivos binários servem para armazenar programas de computador, arquivos de imagens, arquivos de vídeos e, até mesmo, algum formato especial para algum aplicativo específico, conforme lecionam Forbellone e Eberspächer (2005).

Modos de abertura de arquivos

Para poder acessar o conteúdo de um arquivo, seja para leitura ou para a escrita nele, é preciso primeiro **abrir o arquivo**. Façamos uma analogia: se você vai ler um livro, primeiro você precisa localizar ele na estante de livros, depois, você deve abrir o livro, para, só então, começar a ler seu conteúdo. Com os arquivos de computadores, o processo é praticamente o mesmo. Quando você precisa acessar algum arquivo do seu computador, primeiro, o sistema operacional localiza ele no disco rígido (ou no disco em que ele estiver armazenado). Então, ao ser localizado, o arquivo estará disponível para que você possa trabalhar com ele.

Porém, antes de começar a ler o arquivo, da mesma forma como ocorre com o livro, você deve abrir o arquivo. Quando você vai abrir um arquivo, existem alguns modos de abertura. O **modo de abertura** é uma configuração que você deve especificar para informar ao sistema operacional qual é o nível de controle que você precisa ter sobre o arquivo. Se você vai apenas ler seu conteúdo, você pode abrir o arquivo em **modo de leitura**. Agora, se você precisa abrir o arquivo para modificar seu conteúdo, seja escrevendo novas linhas no final do arquivo ou alterando alguma linha existente, você deve abrir o arquivo em **modo de escrita**. O Quadro 1 apresenta os modos de abertura que estão disponíveis na linguagem Python.

Quadro 1. Principais modos de abertura de arquivos da linguagem Python

MODO DE ABERTURA	DESCRIÇÃO
`'r'`	Abre o arquivo somente para a leitura do seu conteúdo
`'w'`	Abre o arquivo para escrita a partir do início do arquivo
`'a'`	Abre o arquivo para escrita a partir do final do arquivo
`'t'`	Abre o arquivo como um arquivo de texto
`'b'`	Abre o arquivo como um arquivo binário

Fonte: Adaptado de Python Software Foundation (2019a).

Abrindo arquivos com a linguagem Python

Para abrir um arquivo, a linguagem Python conta com o comando `open`. Esse comando deve receber como parâmetro, no mínimo, o nome do arquivo que será aberto. Outro parâmetro opcional do comando `open` é o modo de abertura do arquivo. Caso não seja especificado nenhum modo, ele vai assumir que você quer abrir um arquivo texto em modo de leitura, isto é, '**rt**', conforme leciona Perkovic (2016). Veja na Figura 1 um exemplo de como abrir um arquivo.

```
[1]: arquivo = open("ArquivoExemplo.txt", 'rt')
```

Figura 1. Exemplo de abertura de arquivo na linguagem Python.

A variável `arquivo` é um tipo especial de variável, que serve para especificar o arquivo que está sendo acessado. Todas as funções que serão usadas para ler seu conteúdo ou para modificá-lo deverão ser feitas a partir dessa variável identificadora do arquivo. Tome cuidado para não tentar abrir um arquivo inexistente. Caso isso ocorra, a linguagem Python vai gerar um erro semelhante ao erro ilustrado na Figura 2 (PYTHON SOFTWARE FOUNDATION, 2019a).

```
[1]: arquivo = open("ArquivoInexistente.txt", 'rt')

---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-1-c55e352c21bb> in <module>
----> 1 arquivo = open("ArquivoInexistente.txt", 'rt')

FileNotFoundError: [Errno 2] No such file or directory: 'ArquivoInexistente.txt'
```

Figura 2. Mensagem de erro que pode ocorrer no caso de arquivos inexistentes.

Acessando o conteúdo dos arquivos texto

Depois de abrir o arquivo, você estará pronto para acessar seu conteúdo. Veja na Figura 3 um exemplo de como acessar o conteúdo do arquivo. Foi usado o comando `for` para percorrer todas as linhas do arquivo. A função `readlines()` do objeto arquivo retorna uma lista com todas as linhas que o arquivo possui. Assim, é possível acessar linha por linha do arquivo de forma simples e prática (PYTHON SOFTWARE FOUNDATION, 2019a).

```
[1]: arquivo = open("ArquivoExemplo.txt", 'rt')
     for linha in arquivo.readlines():
         print(linha)

     Este é um arquivo de exemplo para leitura de arquivos na Linguagem Python.

     Esta é a segunda linha do arquivo de teste.

     Aqui temos a terceira.
```

Figura 3. Exemplo de acesso do conteúdo de um arquivo texto na linguagem Python. Nesse exemplo, cada linha do arquivo é lida e, depois, impressa na tela.

É possível ainda acessar uma linha de cada vez, sem usar o comando `for`. Na Figura 4, você pode observar como é possível ler uma linha de cada vez com o uso da função `readline()`. Após você usar essa função, ela vai automaticamente ler a próxima linha.

```
[1]: arquivo = open("ArquivoExemplo.txt", 'rt')
     arquivo.readline()

[1]: 'Este é um arquivo de exemplo para leitura de arquivos na Linguagem Python.\n'

[2]: arquivo.readline()

[2]: 'Esta é a segunda linha do arquivo de teste.\n'

[3]: arquivo.readline()

[3]: 'Aqui temos a terceira.'

[4]: arquivo.readline()

[4]: ''
```

Figura 4. Exemplo de acesso do conteúdo de um arquivo texto na linguagem Python. Nesse exemplo, cada linha é lida de uma vez.

Arquivos separados por vírgulas (csv)

Os **arquivos separados por vírgulas** (csv) são arquivos texto com vários dados separados, como se fossem uma tabela. Cada linha do arquivo é como se fosse uma linha da tabela. Os campos em uma mesma linha são separados usando, tradicionalmente, a vírgula; ou seja, cada coluna da tabela é separada da outra por meio de um caractere separador.

Fique atento

Nos arquivos csv, a vírgula é o caractere separador mais usado; porém, é possível usar, por exemplo, ponto e vírgula ou caractere de tabulação (tecla Tab).

Esses arquivos são muito úteis quando queremos levar um conjunto de dados de um programa para outro. Por exemplo, você abre uma planilha eletrônica contendo vários dados e exporta esses dados para um arquivo csv. Depois, você pode abrir esses dados na linguagem Python, para fazer algum processamento com eles. Veja na Figura 5 um exemplo de arquivo csv.

```
1 ColunaA, ColunaB, ColunaC
2 campo 1, 123.01, Texto um pouco mais longo
3 Campo 2, -123.99, Texto curto
4  , 0, Linha com a coluna A em branco
5
```

Figura 5. Exemplo de um arquivo separado por vírgulas (csv).

Acessando o conteúdo dos arquivos csv

Para acessar um arquivo csv, é preciso, antes de tudo, que o arquivo seja aberto. Para isso, você pode usar o mesmo comando usado para abrir um arquivo de texto — isto é, o comando `open`. Repare que um arquivo csv é, antes de tudo, um arquivo de texto. Logo, o modo de abertura do arquivo deve ser especificado da mesma forma como para os arquivos texto. Veja o Quadro 1 para relembrar os modos de abertura possíveis. Para a maioria das situações, o arquivo csv será aberto no modo padrão, que é **'rt'** (PYTHON SOFTWARE FOUNDATION, 2019b).

A linguagem Python possui o módulo **csv**, que facilita o processo de leitura e interpretação dos campos separados por vírgulas. Para usar esse módulo, é preciso informar que será utilizado esse módulo antes de digitar os comandos, utilizando o comando `import csv`. O módulo csv possui o comando `reader`, que cria um objeto capaz de ler e interpretar cada linha do arquivo csv. Por exemplo, com o comando `leitorCSV = csv.reader(arquivoCSV)`, será criada a variável `leitorCSV`, pela qual será possível acessar o `arquivoCSV`. A vantagem de usar esses comandos é que, por meio deles, você pode ler um campo por vez (PYTHON SOFTWARE FOUNDATION, 2019b).

Link

Acesse o *link* a seguir para verificar mais funções do módulo csv da linguagem Python.

https://qrgo.page.link/c6aTR

No exemplo da Figura 6, um arquivo csv é aberto usando o comando `open`; depois, seu conteúdo é acessado linha por linha, por meio do objetivo criado pelo comando `csv.reader`. Observe nesse exemplo que a primeira instrução orienta a linguagem Python a usar o módulo csv. Esse exemplo se limita a ler linha por linha do arquivo e imprimir o conteúdo lido na tela. Repare que cada linha lida é, na verdade, uma lista, em que os elementos são os campos existentes no arquivo. Ainda nesse exemplo, é possível observar que o comando `csv.reader` possui o parâmetro opcional `delimiter=','`, que serve para especificar qual é o caractere que será usado como delimitador de campos do arquivo csv (PYTHON SOFTWARE FOUNDATION, 2019b).

```
[1]: import csv

     arquivoCSV = open("ArquivoCSVExemplo.csv", 'rt')
     reader = csv.reader(arquivoCSV, delimiter=',')
     for linha in reader:
         print(linha)

     ['ColunaA', ' ColunaB', ' ColunaC']
     ['campo 1', ' 123.01', ' Texto um pouco mais longo']
     ['Campo 2', ' -123.99', ' Texto curto']
     [' ', ' 0', ' Linha com a coluna A em branco']
```

Figura 6. Algoritmo em linguagem Python capaz de ler um arquivo csv. Nesse exemplo, é ilustrada a leitura do arquivo csv reproduzido na Figura 5.

Na Figura 7, verifica-se outra forma de acessar o conteúdo do arquivo csv. Repare que agora foi utilizado como separador de campos o caractere ponto e vírgula. Outro aspecto muito importante desse exemplo é que, para que os dados do arquivo csv fossem exibidos, a variável `linha` foi acessada da mesma forma que uma variável do tipo lista. Usando os colchetes e o índice 0, pode-se acessar o conteúdo da primeira coluna do arquivo csv; usando o índice 1, pode-se acessar a segunda coluna, e, assim, sucessivamente.

ARQUIVO CSV

```
1 A;10
2 B;11
3 C;13
4 D;14
```

ALGORITMO

```
[1]: import csv

     arquivoCSV = open("ArquivoExemploB.csv", 'rt')
     reader = csv.reader(arquivoCSV, delimiter=';')
     for linha in reader:
         print("Elemento: " + str(linha[0]) + " = " + str(linha[1]))

     Elemento: A = 10
     Elemento: B = 11
     Elemento: C = 13
     Elemento: D = 14
```

Figura 7. Exemplo de leitura de um arquivo csv cujo separador é o ponto e vírgula. Nesse exemplo, é ilustrado o acesso aos campos como listas.

Aplicações práticas de manipulação de arquivos

Agora que você já viu como acessar o conteúdo de arquivos csv e de arquivos texto, vamos conhecer um pouco as aplicações que podemos fazer usando esse recurso. Os arquivos de texto e csv são a forma mais simples de se transportar alguma informação de um programa para outro. Com dois exemplos simples, você vai ver como é possível usar os métodos de leitura de arquivos csv e arquivos texto para a manipulação e o tratamento de dados.

Tabela de vendas de impressora fiscal

Suponha que uma impressora de cupom fiscal registre internamente, em uma memória dedicada, todos os cupons que ela emitiu desde que foi fabricada. Com alguns comandos do programa de controle dessa impressora, pode-se enviar um arquivo csv para o computador, contendo um relatório resumido dos cupons que foram emitidos. Um algoritmo em linguagem Python pode ser criado para acessar o conteúdo desse equipamento, a fim de mostrar quantos cupons a impressora já emitiu desde que foi criada, o valor total acumulado de cupons emitidos e o valor médio por cupom.

A Figura 8 ilustra o conteúdo de um arquivo csv e um algoritmo em linguagem Python. Esse algoritmo faz a leitura do arquivo csv e, com o auxílio do comando `for`, da variável contadora `noCupons` e da variável `valorAcumulado`, ele é capaz de determinar quantos cupons foram emitidos, qual é o valor acumulado de cupons emitidos e qual é o valor médio dos cupons que foram emitidos. Observe que, para a determinação do valor acumulado, foi usada a função `int` para converter o conteúdo da variável `cupom` para um número inteiro. Isso foi necessário porque o módulo csv considera que todo campo é uma *string*.

ARQUIVO CSV

```
001,100
002,150
003,54
004,99
005,356
006,1230
007,600
008,120
009,84
010,32
011,86
012,99
013,230
014,73
015,20
016,123
017,54
```

ALGORITMO

```
[1]: import csv

arquivoCSV = open("Cupom.csv", 'rt')
leitorCSV = csv.reader(arquivoCSV)

noCupons = 0
valorAcumulado = 0

for cupom in leitorCSV:
    noCupons += 1
    valorAcumulado += int(cupom[1])

valorMedio = valorAcumulado / noCupons

print("Foram processados " + str(noCupons) + " cupons.")
print("Valor acumulado de " + str(valorAcumulado))
print("Valor médio por cupom é de " + str(valorMedio))
```

```
Foram processados 17 cupons.
Valor acumulado de 3510
Valor médio por cupom é de 206.47058823529412
```

Figura 8. Exemplo de processamento de quantidade de cupons emitidos, valor máximo acumulado e valor médio por cupons a partir do arquivo csv gerado por uma impressora fiscal.

Processamento de dados

Uma situação em que a leitura dos arquivos csv se torna muito interessante é realizar algum processamento a partir dos registros de um programa. Suponha que, por exemplo, um determinado programa de controle de estoque é capaz de gerar um arquivo csv com toda a relação de estoque da empresa. O arquivo gerado possui o código do produto, um nome curto para identificar, a quantidade de peças em estoque e a quantidade máxima suportada em estoque. Agora, suponha que você precise enviar um novo pedido de compras para o fornecedor das peças, mas você só vai pedir as peças cujos estoques tenham zerado, ou que estejam abaixo de 5% do valor de estoque máximo comportado pela empresa. Assim, será empregado um algoritmo em linguagem Python para fazer esse processamento.

A Figura 9 ilustra um exemplo do relatório de estoque, em que as colunas são: código do produto, descrição simples, quantidade de estoque e capacidade máxima de estoque. Nessa mesma figura, foi ilustrado um algoritmo em Python que faz a leitura desse arquivo csv; a partir dos dados lidos para cada produto, o algoritmo verifica se o estoque é 0 e se o estoque está inferior a 5% do estoque máximo. Caso alguma dessas condições sejam satisfeitas, o algoritmo vai imprimir na tela os produtos que devem ser solicitados para repor o estoque.

ARQUIVO CSV

```
001, Lanterna,  10, 1000
002, Caneta,    50,   80
003, Garrafa,    0,   10
004, Régua,     15,   16
005, Borracha, 200,  400
```

ALGORITMO

```
[1]: import csv

     arquivoCSV = open("Estoque.csv")
     leitorCSV = csv.reader(arquivoCSV)

     print("Lista de itens a repor:")

     for item in leitorCSV:
         codigo      = item[0]
         nome        = item[1]
         estoque = int(item[2])
         estoqueMax = int(item[3])

         if (estoque == 0) or (estoque < (estoqueMax * 5 / 100)):
             print(codigo + " - " + nome)

Lista de itens a repor:
001 -  Lanterna
003 -  Garrafa
```

Figura 9. Exemplo de processamento para levantamento de lista de peças a comprar, com base em critérios preestabelecidos. Como base de dados para o processamento, foi considerado um relatório em formato csv com os dados de estoque da empresa.

Referências

FORBELLONE, A. L. V.; EBERSPÄCHER, H. F. *Lógica de programação*: a construção de algoritmos e estruturas de dados. 3. ed. São Paulo: Pearson Prentice Hall, 2005. 218 p.

PERKOVIC, L. *Introdução à computação usando Python*: um foco no desenvolvimento de aplicações. Rio de Janeiro: LTC, 2016. 516 p.

PYTHON SOFTWARE FOUNDATION. Built-in Functions: open. *In*: PYTHON SOFTWARE FOUNDATION. *The Python tutorial*. Wilmington, 2019a. Disponível em: https://docs.python.org/3/library/functions.html#open. Acesso em: 11 jun. 2019.

PYTHON SOFTWARE FOUNDATION. CSV File Reading and Writing. *In*: PYTHON SOFTWARE FOUNDATION. *The Python tutorial*. Wilmington, 2019b. Disponível em: https://docs.python.org/3/library/csv.html. Acesso em: 11 jun. 2019.

Leituras recomendadas

BANIN, S. L. *Python 3*: conceitos e aplicações: uma abordagem didática. São Paulo: Érica, 2018, 264 p.

BARRY, P. *Use a cabeça! Python*. 2. ed. Rio de Janeiro: Alta Books, 2018. 616 p. (Série Use a Cabeça/Head First).

READING and Writing Files in Python. *Python for Beginners*, [*S. l.*], 8 Jul. 2013. Disponível em: https://www.pythonforbeginners.com/files/reading-and-writing-files-in-python. Acesso em: 11 jun. 2019.

USING the CSV module in Python. *Python for Beginners*, [*S. l.*], 18 Jan. 2013. Disponível em: https://www.pythonforbeginners.com/systems-programming/using-the-csv-module-in-python/. Acesso em: 11 jun. 2019.

Encerra aqui o trecho do livro disponibilizado para esta Unidade de Aprendizagem. Na Biblioteca Virtual da Instituição, você encontra a obra na íntegra.

Conteúdo:

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS