

Introdução à Programação

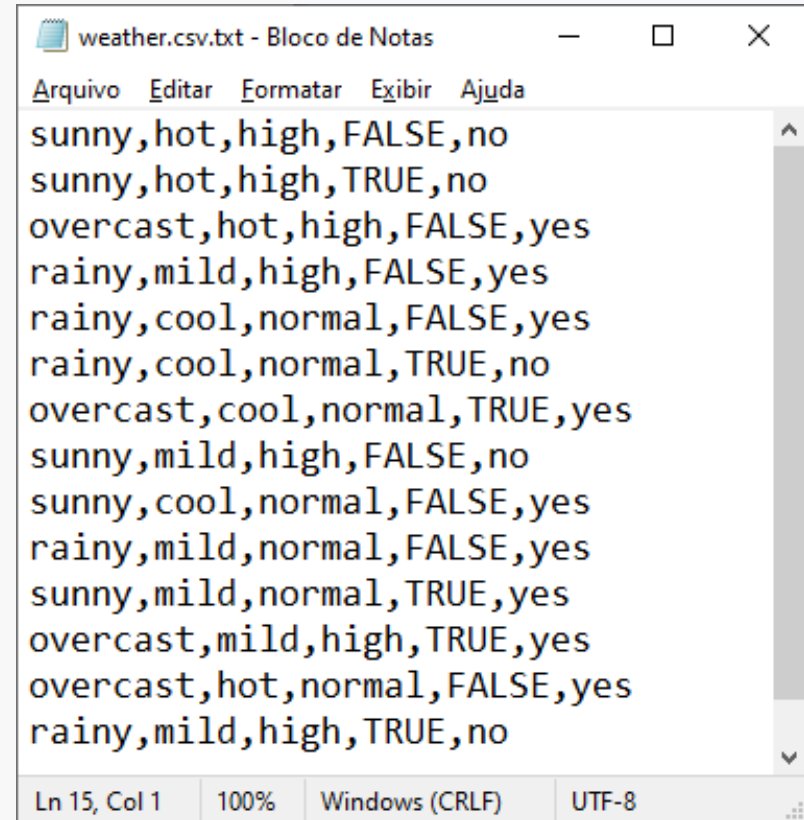
Aula 17: Arquivos Texto

Prof. Eduardo Corrêa Gonçalves

07/07/2020

Introdução (1/2)

- O que é um Arquivo?
 - Um arquivo é um conjunto de contíguo de bytes relacionados, mantido em algum dispositivo de armazenamento permanente (HD, pen drive, cartão de memória etc.).
 - Há vários tipos de arquivo.
 - Porém, neste curso trabalharemos apenas com os **arquivos texto**.
 - Trata-se do tipo de arquivo cujo conteúdo pode ser **visualizado** em **qualquer editor de textos**, como o bloco de notas.



```
weather.csv.txt - Bloco de Notas
Arquivo  Editar  Formatar  Exibir  Ajuda
sunny,hot,high,FALSE,no
sunny,hot,high,TRUE,no
overcast,hot,high,FALSE,yes
rainy,mild,high,FALSE,yes
rainy,cool,normal,FALSE,yes
rainy,cool,normal,TRUE,no
overcast,cool,normal,TRUE,yes
sunny,mild,high,FALSE,no
sunny,cool,normal,FALSE,yes
rainy,mild,normal,FALSE,yes
sunny,mild,normal,TRUE,yes
overcast,mild,high,TRUE,yes
overcast,hot,normal,FALSE,yes
rainy,mild,high,TRUE,no
```

Ln 15, Col 1 100% Windows (CRLF) UTF-8

Introdução (2/2)

- **Arquivos Texto**

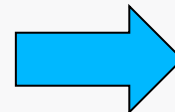
- Como estatístico, uma das tarefas mais comuns que você realizará será ler, analisar e escrever (gravar) **arquivos texto**
- Nesta aula, trabalharemos principalmente com um **tipo específico de arquivo texto** conhecido como **arquivo CSV**.
 - CSV é um dos formatos mais utilizados para armazenar **bases de dados** (BDs) no formato tabular (dados dispostos em **linhas** e **colunas**).
 - Há vários sites na Internet que fornecem bases de dados nesse formato. Elas podem ser estudadas pelos estatísticos.

Arquivos CSV (1/3)

- Um CSV é um **arquivo texto** contendo um conjunto de dados a respeito de “alguma coisa” (algum objeto de estudo).
- Exemplos:** país, animal, pessoa, carro, etc.

1	sigla	nome	continente	extensao
2	AR	Argentina	América	2780
3	BR	Brasil	América	8511
4	FR	França	Europa	644
5	IT	Itália	Europa	301
6	UK	Reino Unido	Europa	244

Conjunto de dados
representado como tabela



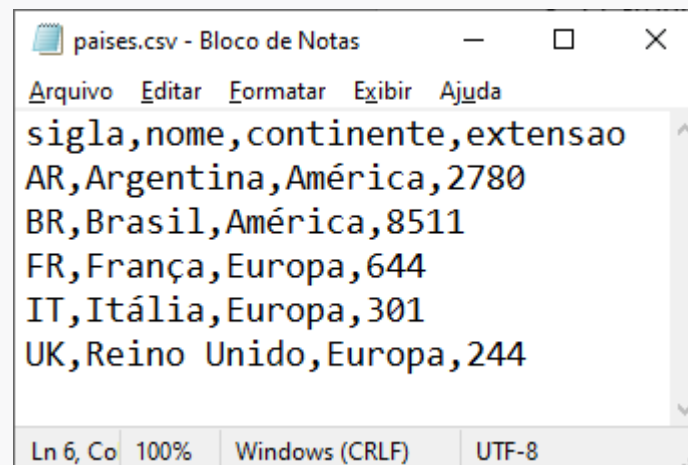
```
países.csv - Bloco de Notas
Arquivo  Editar  Formatar  Exibir  Ajuda
sigla,nome,continente,extensao
AR,Argentina,América,2780
BR,Brasil,América,8511
FR,França,Europa,644
IT,Itália,Europa,301
UK,Reino Unido,Europa,244
Ln 6, Co 100%  Windows (CRLF)  UTF-8
```

Mesmo conjunto de dados
representado no formato CSV

Arquivos CSV (2/3)

Formato CSV

- Cada **linha** corresponde a um **registro** (ou **observação**, no vocabulário dos estatísticos).
- As **colunas** são separadas por vírgula e representam os **atributos** (propriedades, ou **variáveis** no vocabulário dos estatísticos) dos registros.
- O arquivo pode ter uma 1ª linha de **cabeçalho**, com os nomes dos atributos. Isso torna o arquivo autodescritivo.



```
países.csv - Bloco de Notas
Arquivo  Editar  Formatar  Exibir  Ajuda
sigla,nome,continente,extensao
AR,Argentina,América,2780
BR,Brasil,América,8511
FR,França,Europa,644
IT,Itália,Europa,301
UK,Reino Unido,Europa,244
Ln 6, Co 100%  Windows (CRLF)  UTF-8
```

Arquivos CSV (3/3)

- **EXERCÍCIO**: digite e grave no HD ou pen-drive a base “países.csv”
 - Possui 5 registros. Cada um corresponde a um país.
 - Possui 4 atributos para descrever cada país: “sigla”, “nome”, “continente” e “extensão”.
 - A 1ª linha é a linha de cabeçalho
- Utilizaremos esse CSV para os exemplos da aula.

```
países.csv - Bloco de Notas
Arquivo  Editar  Formatar  Exibir  Ajuda
sigla,nome,continente,extensao
AR,Argentina,América,2780
BR,Brasil,América,8511
FR,França,Europa,644
IT,Itália,Europa,301
UK,Reino Unido,Europa,244
```

linha de cabeçalho

5 registros

Ln 6, Co 100% Windows (CRLF) UTF-8

Outros exemplos de arquivos CSV (1/2)

- **Exemplo 2:** “animais.csv” (base de dados de animais)
 - 14 registros (cada um referente a um animal)
 - 3 atributos:
 - “animal” - nome do animal
 - “fofura” – escala de fofura (0 a 100)
 - “tamanho” – escala de tamanho (0 a 100)

animal,fofura,tamanho
gatinho filhote,95,15
hamster,80,5
tarântula,8,3
cachorrinho filhote,90,20
crocodilo,5,40
golfinho,60,45
urso panda,75,45
lagosta,2,15
capivara,70,35
elefante,65,90
mosquito,1,1
peixinho dourado,25,2
cavalo,50,50
galinha,25,15

Outros exemplos de arquivos CSV (2/2)

- **Exemplo 3:** “notas.csv” (base de dados de notas de alunos)
 - Veja que esse BD é um pouco diferente.
 - Não possui cabeçalho. *Problema: ele deixa de ser autodescritivo.*
 - Os dados são separados por ponto-e-vírgula em vez de vírgula.
 - Mesmo assim, pode ser considerado um CSV !
 - 4 registros (cada um referente a um aluno)
 - 3 atributos:
 - Matrícula do aluno
 - Nota na 1ª prova
 - Nota na 2ª prova

M0012023;9.8;9.5

M0022023;5.3;4.1

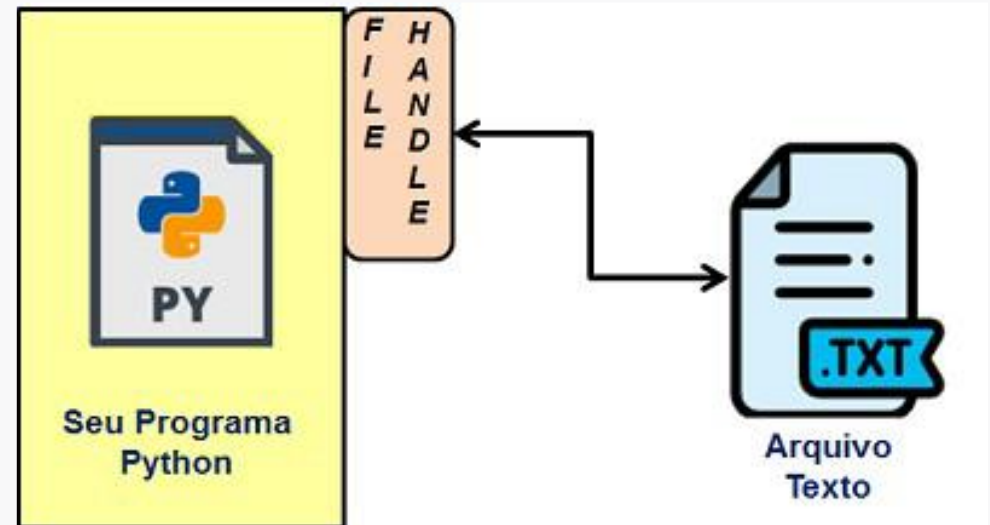
M0032023;2.5;8.0

M0042023;7.5;7.5

Arquivos Texto no Python (1/11)

- **Função open()**

- No Python padrão, é necessário usar a função **open()** para comandar a **abertura** de um arquivo.
- **Abrir** significa pedir ao SO para **encontrar o endereço** de localização do arquivo no HD, pen drive, etc.
- Ao encontrar o endereço do arquivo, o SO retornará um **file handle** para o programa.
 - O file handle **não** é a mesma coisa que conteúdo (dados) do arquivo.
 - Na verdade, é uma ferramenta que permite ao programador “manejar” os dados do arquivo.



Função open() (1/2)

- **Função open() - Importância**

- Trabalhar com arquivos via **open()** / **file handle** **não** é tão “confortável” como trabalhar com um Data Frame da linguagem R ou do pacote pandas do Python (*como você verá no futuro...*).
- No entanto, **é importante** saber lidar com a função **open()** pelos seguintes motivos:
 - É uma ferramenta **eficiente** para realizar o **acesso sequencial** (linha por linha). Muitas vezes, esta é a única opção viável para lidar com bases muito grandes.
 - É a ferramenta mais simples para lidar com **arquivos separados por colunas** (formato raro hoje em dia, mas ainda usado).
 - Trata-se de uma função do **Python padrão**. Ou seja: permite com que você trabalhe com arquivos sem precisar instalar ou importar nenhum pacote adicional.

Função open() (2/2)

- Função open() – Sintaxe

f = open(nome_arq, modo)

- **nome_arq**: nome do arquivo (incluindo especificação do diretório). Único parâmetro obrigatório.
- **modo**:
 - **'r'**: abre o arquivo para leitura (default)
 - **'w'**: abre o arquivo para escrita, eliminando a versão anterior.
 - **'a'**: abre o arquivo no modo append. Se o arquivo existir, as novas linhas serão inseridas no final.
 - **'rb', 'wb', 'ab'** nos permitem trabalhar com arquivos binários, em vez de arquivos texto.
- Por exemplo, comando abaixo, abre para leitura o arquivo 'weather.csv', localizado na pasta "c:\bases"
 - **f = open('C:/bases/weather.csv')**

Acesso Sequencial (1/6)

- O que é Acesso Sequencial?
 - A forma mais comum de acessar arquivos texto no Python padrão é **processando linha por linha**, sequencialmente.
 - Neste modelo, o Python permite com que um arquivo seja visto como uma **coleção de linhas**.
 - É como se fosse uma fila de linhas. Podemos “desenfileirá-las” uma por uma na ordem em que estão dispostas no arquivo.
 - Muito **eficiente** em termos de **ocupação de memória**. A cada iteração, apenas uma linha reside na memória.

arq_teste.csv

```
Rakesh,25  
Vijay,43  
Yash,18  
Juily,51  
Vidur,39
```



```
f = open('arq_teste.csv')  
for linha in f:  
    print(linha)
```

```
f.close()
```

```
>>>  
Rakesh,25
```

```
Vijay,43
```

```
Yash,18
```

```
Juily,51
```

```
Vidur,39
```

Acesso Sequencial (2/6)

- **Acesso Sequencial – Sintaxe:**
 - A sintaxe básica para ler um arquivo sequencialmente é muito simples:

for linha **in** f:
 comandos

- A cada iteração do **for**, a próxima linha será armazenada como uma **string** na variável “linha”.
- Veja que inicialmente, precisamos abrir o arquivo com o método **open()**.
- No final, fechá-lo com o método **close()**.
 - Na operação de leitura isso não é obrigatório, mas é uma boa prática.

arq_teste.csv

```
Rakesh,25  
Vijay,43  
Yash,18  
Juily,51  
Vidur,39
```



```
f = open('arq_teste.csv')  
for linha in f:  
    print(linha)
```

```
f.close()
```

```
>>>  
Rakesh,25
```

```
Vijay,43
```

```
Yash,18
```

```
Juily,51
```

```
Vidur,39
```

Acesso Sequencial (3/6)

- Problema: o caractere “\n”

- Ao executar o exemplo, você deve ter percebido que uma linha em branco foi impressa depois de cada `print()`.
- Isto ocorreu porque o último caractere de uma linha do arquivo é na verdade “invisível”.
- Ele se chama **fim de linha** ou **newline** e é representado por “\n” nas linguagens de programação.
 - Na verdade o fim de linha no Windows é representado por dois caracteres invisíveis (CR + LF). No Windows e Mac, apenas um (LF).
 - Para simplificar, referenciaremos o fim de linha sempre como \n

```
f = open('arq_teste.csv')
for linha in f:
    print(linha)
```

```
f.close()
```

```
>>>
Rakesh,25
```

```
Vijay,43
```

```
Yash,18
```

```
Juily,51
```

```
Vidur,39
```

Acesso Sequencial (4/6)

- Problema: o caractere “\n”
 - A primeira linha do arquivo é **Rakesh,25**
 - Veja que quando mandamos imprimir o comprimento dessa linha, o valor retornado é 10, embora a gente só enxergue 9 caracteres.
 - A mesma coisa ocorre para as demais linhas !!!

```
f = open('arq_teste.csv')
for linha in f:
    print("comprimento = ", len(linha))
    print(linha)
```

```
f.close()
```

```
>>>
comprimento = 10
Rakesh,25
```

```
comprimento = 9
Vijay,43
```

```
comprimento = 8
Yash,18
```

```
comprimento = 9
Juily,51
```

```
comprimento = 9
Vidur,39
```

Acesso Sequencial (5/6)

- Solução: usar o método `rstrip()`

- Para descartar o “\n” após ler uma linha, você pode utilizar:

- `linha.rstrip()`:
 - Este método **remove** caracteres do tipo whitespace que estiverem à direita de uma string.
 - Caracteres como espaço em branco, “\t” e “\n” são considerados whitespace.

```
f = open('arq_teste.csv')
for linha in f:
    print(linha.rstrip())

f.close()

>>>
Rakesh,25
Vijay,43
Yash,18
Juily,51
Vidur,39
```

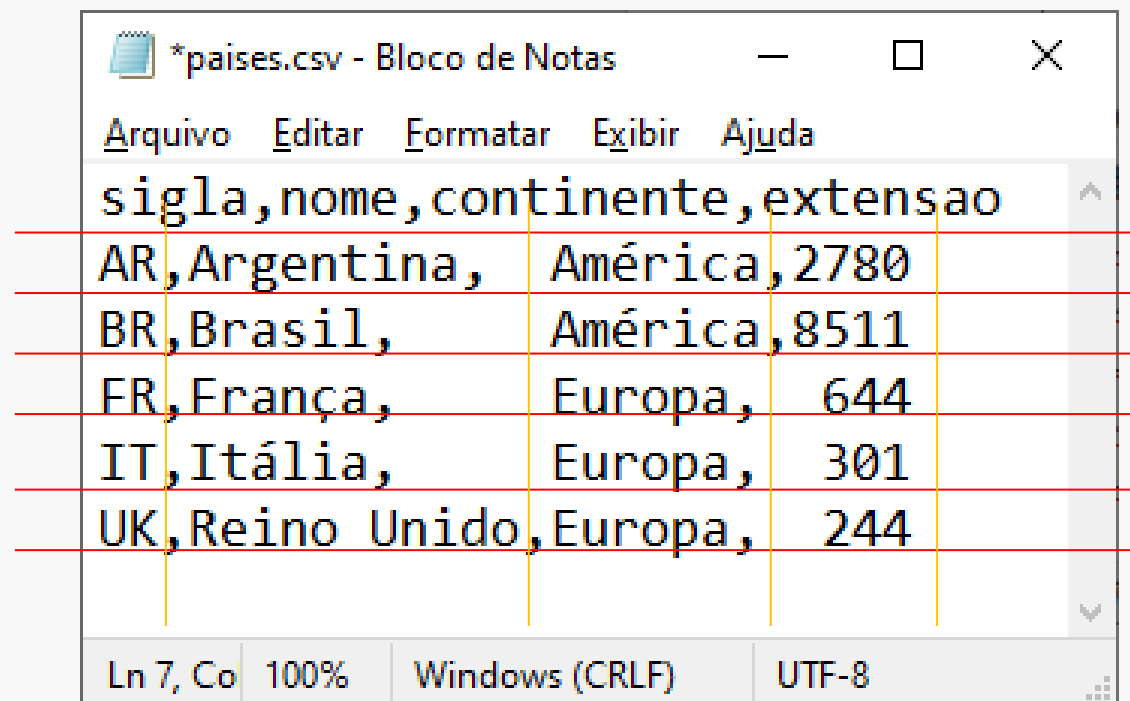

Acesso Sequencial (6/6)

- **Acesso Sequencial - Resumo**

- É o modo de acesso a arquivos mais usado no Python padrão.
- Recupera as linhas do arquivo, **uma por uma**, da primeira à última, em um laço **for**.
- Cada linha é sempre lida para uma variável **string**.
- Só permite avançar as linhas, **nunca recuar**.
- **Eficiente**, especialmente na questão da ocupação de memória.
- Nos próximos slides, mostraremos a **receita** para processar sequencialmente bases de dados texto em dois diferentes formatos:
 - Arquivo CSV;
 - Arquivo separado por colunas;
- Para rodar os exemplos, digite as bases de dados, salve com o nome indicado e as coloque na mesma pasta dos programas.

Processando Arquivos CSV (1/8)

- Como processar um arquivo CSV?
 - Primeiro precisamos abrir o arquivo com `open()` especificando o nome e encoding do arquivo corretamente.
 - Depois fazer o acesso sequencial, linha por linha, usando `for`.
 - Para cada linha, é preciso separar as informações em cada coluna.



sigla	nome	continente	extensao
AR	Argentina	América	2780
BR	Brasil	América	8511
FR	França	Europa	644
IT	Itália	Europa	301
UK	Reino Unido	Europa	244

Processando Arquivos CSV (2/8)

- **Passo 1: abrir o arquivo com open()**

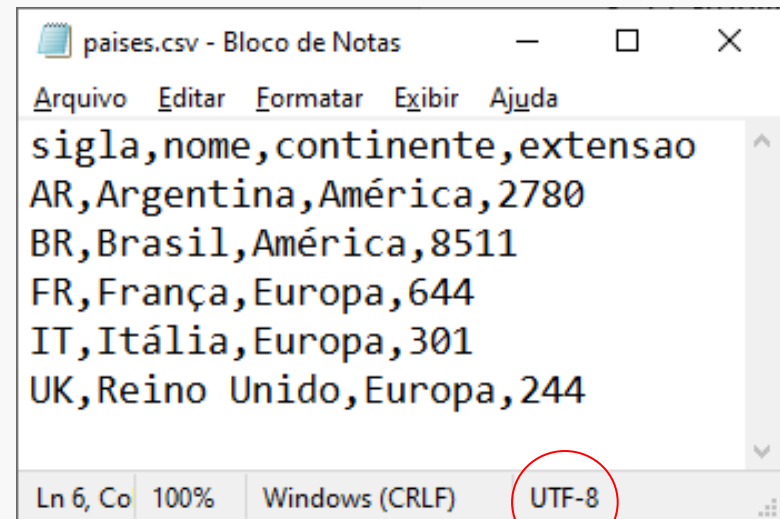
- Basta especificar:

(1) o nome caminho e nome do arquivo.

(2) O *encoding* do arquivo

- ***Mas o que é encoding????***

```
f = open('países.csv', encoding="utf-8")
```



encoding

Processando Arquivos CSV (3/8)

- **Encoding**

- Encoding é o método que foi usado para converter códigos Unicode para bytes ao salvar o arquivo texto.
- Existem muitos encodings, mas os mais usados são “ansi” e “utf-8”.
- Por padrão, a função `open()` considera que o encoding do arquivo é “ansi”. Se for utf-8, você tem que indicar, como foi feito no programa ao lado.

```
f = open('paises.csv', encoding="utf-8")
for linha in f:
    print(linha.rstrip())

f.close()

>>>
sigla,nome,continente,extensao
AR,Argentina,América,2780
BR,Brasil,América,8511
FR,França,Europa,644
IT,Itália,Europa,301
UK,Reino Unido,Europa,244
```

Processando Arquivos CSV (4/8)

- **Encoding**

- E o que acontece se o encoding errado for usado?
- Nesse caso, alguns caracteres (ex.: caracteres acentuados) vão ser lidos de forma incorreta.
- Ao tentar exibi-los, vai aparecer “lixo” na tela.
- Veja ao lado, onde um arquivo utf-8 é lido como ansi (ansi é o padrão default, mas é o encoding errado para esse arquivo).

```
f = open('paises.csv')
for linha in f:
    print(linha.rstrip())

f.close()

>>>
sigla,nome,continente,extensao
AR,Argentina,AmÃ©rica,2780
BR,Brasil,AmÃ©rica,8511
FR,FranÃ§a,Europa,644
IT,ItÃ¡lia,Europa,301
UK,Reino Unido,Europa,244
```

Processando Arquivos CSV (5/8)

- **Encoding – Receita de Bolo**
 - Ao tentar abrir um arquivo, se você não souber o encoding, você pode primeiro usar o `open()` especificando `encoding = 'utf-8'`
 - Isso porque o UTF-8 é o encoding mais comum.
 - Se der problema (aparecer os caracteres lixo) simplesmente remova o parâmetro.

```
f = open('paises.csv', encoding="utf-8")
for linha in f:
    print(linha.rstrip())

f.close()

>>>
sigla,nome,continente,extensao
AR,Argentina,América,2780
BR,Brasil,América,8511
FR,França,Europa,644
IT,Itália,Europa,301
UK,Reino Unido,Europa,244
```

Processando Arquivos CSV (6/8)

- **Passo 2: fazer o acesso sequencial separando linhas e colunas**
 - Ao lado, a receita completa:
 - O **for linha in f:** percorre as linhas
 - Para cada linha, usamos **linha = linha.rstrip()** para remover o '\n' à direita.
 - Depois usamos **lst = linha.split(",")** para transformar cada linha em uma **lista de strings** (basta mandar separar pela “,” no caso desse arquivo).
 - Assim cada atributo é separado em uma posição da lista.

```
f = open('países.csv', encoding="utf-8")

for linha in f:
    linha = linha.rstrip()
    colunas = linha.split(",")
    print(colunas)

f.close()

>>>
'sigla', 'nome', 'continente', 'extensao']
['AR', 'Argentina', 'América', '2780']
['BR', 'Brasil', 'América', '8511']
['FR', 'França', 'Europa', '644']
['IT', 'Itália', 'Europa', '301']
['UK', 'Reino Unido', 'Europa', '244']
```

Processando Arquivos CSV (7/8)

- **Passo 2: fazer o acesso sequencial separando linhas e colunas**
 - Observe que, para qualquer linha, na lista **colunas** teremos:
 - A **sigla** na **posição 0**
 - O **nome** na **pos. 1**
 - O **continente** na **pos. 2**
 - A **extensão** na **pos. 3**
 - Sendo assim, basta observar o padrão acima para fazer qualquer programa que faça um estudo no arquivo.

```
f = open('países.csv', encoding="utf-8")
```

```
for linha in f:
```

```
    linha = linha.rstrip()
```

```
    colunas = linha.split(",")
```

```
    print(colunas)
```

```
f.close()
```

```
>>>
```

```
'sigla', 'nome', 'continente', 'extensao']
```

```
['AR', 'Argentina', 'América', '2780']
```

```
['BR', 'Brasil', 'América', '8511']
```

```
['FR', 'França', 'Europa', '644']
```

```
['IT', 'Itália', 'Europa', '301']
```

```
['UK', 'Reino Unido', 'Europa', '244']
```


Processando Arquivos CSV (8/8)

- Agora é só usar a receita!

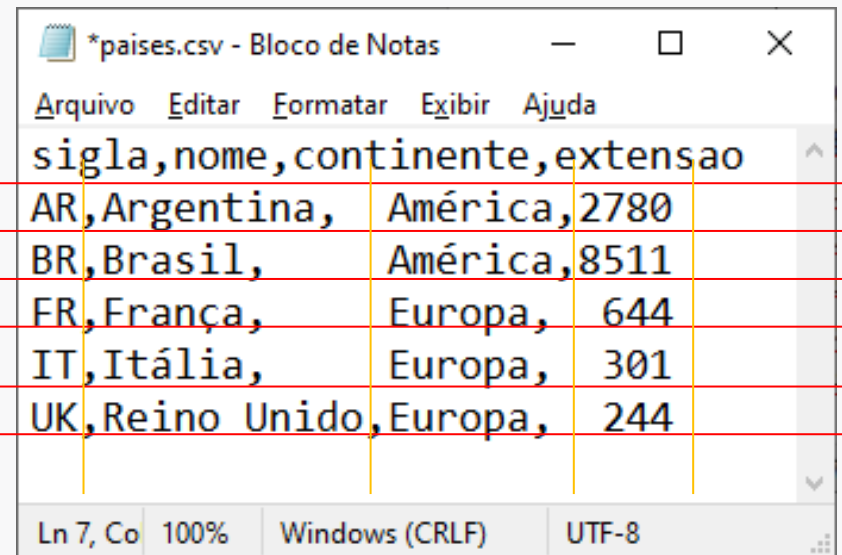
- Acabamos de ver a receita básica para processar um arquivo CSV:

- **PASSO 1 – SEPARAR AS LINHAS:**

- usar `open()` com `for` nas linhas para fazer o acesso sequencial.

- **PASSO 2 – SEPARAR AS COLUNAS:**

- Dentro do laço `for`, usar `split()` para separar as colunas de cada linha, indicando o separador adequado.
- Nesse caso é vírgula, mas pode ser outro.



sigla	nome	continente	extensao
AR	Argentina	América	2780
BR	Brasil	América	8511
FR	França	Europa	644
IT	Itália	Europa	301
UK	Reino Unido	Europa	244

- Nos exercícios seguintes, veremos como aplicar essa receita na prática.
- Aproveitaremos para mostrar mais algumas coisas importantes, como a forma de desconsiderar o cabeçalho e a necessidade de conversão de atributos numéricos.

Exercícios Resolvidos (1/5)

- Exercício Resolvido 1: contar o total de países da Europa e da América

```
1 # contar total de países da Europa e da América
2 f = open('países.csv', encoding="utf-8")
3 f.readline() # usado para pular a linha de cabeçalho
4 tot_eu = tot_am = 0
5
6 for linha in f: # para cada linha...
7     linha = linha.rstrip() # tira o \n do final
8     colunas = linha.split(",") # separa as colunas em uma lista
9     if colunas[2] == 'Europa': tot_eu += 1
10    elif colunas[2] == 'América': tot_am += 1
11
12 f.close()
13 print(f'Total de países da Europa: {tot_eu}')
14 print(f'Total de países da América: {tot_am}')
15
```

Shell x

```
>>> %Run conta_total_eu_am.py
```

```
Total de países da Europa: 3
Total de países da América: 2
```

Python 3.10.4

Exercícios Resolvidos (2/5)

- Exercício Resolvido 1: contar o total de países da Europa e da América

```
1 # contar total de países da Europa e da América
2 f = open('países.csv', encoding="utf-8")
3 f.readline() # usado para pular a linha de cabeçalho
4 tot_eu = tot_am = 0
5
6 for linha in f: # para cada linha...
7     linha = linha.rstrip() # tira o \n do final
8     colunas = linha.split(",") # separa as colunas em uma lista
9     if colunas[2] == 'Europa': tot_eu += 1
10    elif colunas[2] == 'América': tot_am += 1
11
12 f.close()
13 print(f'Total de países da Europa: {tot_eu}')
14 print(f'Total de países da América: {tot_am}')
```

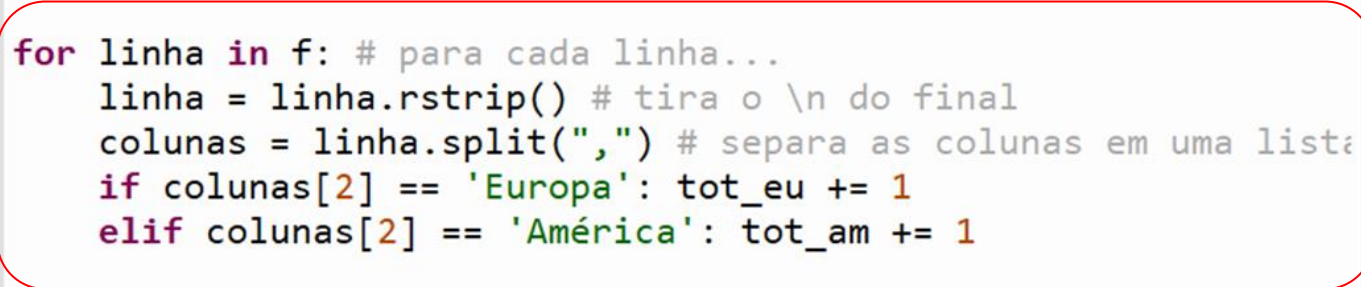
- **IMPORTANTE !!!**

- Antes do for, usamos `f.readline()` para forçar a leitura da linha de cabeçalho.
- `readline()` é um método que lê uma linha e avança para a próxima
- Isso é uma forma de “descartar” a linha, já que ela não possui dados.

Exercícios Resolvidos (3/5)

- Exercício Resolvido 1: contar o total de países da Europa e da América

```
1 # contar total de países da Europa e da América
2 f = open('países.csv', encoding="utf-8")
3 f.readline() # usado para pular a linha de cabeçalho
4 tot_eu = tot_am = 0
5
6 for linha in f: # para cada linha...
7     linha = linha.rstrip() # tira o \n do final
8     colunas = linha.split(",") # separa as colunas em uma lista
9     if colunas[2] == 'Europa': tot_eu += 1
10    elif colunas[2] == 'América': tot_am += 1
11
12 f.close()
13 print(f'Total de países da Europa: {tot_eu}')
14 print(f'Total de países da América: {tot_am}')
```



- **IMPORTANTE (2) !!!**
 - Os resultados são **computados dentro do for** (onde processamos cada linha)

Exercícios Resolvidos (4/5)

- Exercício Resolvido 2: qual a extensão média dos países da Europa?

```
1 # extensao media dos países da Europa
2 f = open('países.csv', encoding="utf-8")
3 f.readline() # usado para pular a linha de cabeçalho
4 soma_ext_eu = tot_eu = 0
5
6 for linha in f: # para cada linha...
7     linha = linha.rstrip() # tira o \n do final
8     colunas = linha.split(",") # separa as colunas em uma lista
9     if colunas[2] == 'Europa':
10         tot_eu += 1
11         soma_ext_eu += float(colunas[3])
12
13 f.close()
14 print(f'Extensão média dos países da Europa: {soma_ext_eu / tot_eu: .2f}')
```

Shell ×

```
>>> %Run extensao_media_eu.py
```

```
Extensão média dos países da Europa: 396.33
```

Exercícios Resolvidos (5/5)

- Exercício Resolvido 2: qual a extensão média dos países da Europa?

```
1 # extensao media dos países da Europa
2 f = open('países.csv', encoding="utf-8")
3 f.readline() # usado para pular a linha de cabeçalho
4 soma_ext_eu = tot_eu = 0
5
6 for linha in f: # para cada linha...
7     linha = linha.rstrip() # tira o \n do final
8     colunas = linha.split(",") # separa as colunas em uma lista
9     if colunas[2] == 'Europa':
10         tot_eu += 1
11         soma_ext_eu += float(colunas[3])
12
13 f.close()
14 print(f'Extensão média dos países da Europa: {soma_ext_eu / tot_eu: .2f}')
```

Shell ×

```
>>> %Run extensao_media_eu.py
Extensão média dos países da Europa: 396.33
```

- **IMPORTANTE !!!**

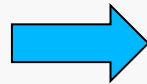
- “**colunas**” é uma lista de strings (o **split()** sempre gera lista de strings)
- Por isso, sempre que você precisar fazer uma conta com algo que estiver nessa lista, será preciso **converter** o valor para **int** ou **float** !!!

Arquivos Separados por Colunas

- Ao contrário do CSV, não possui caractere separador. Veja o exemplo abaixo:
 - Em “arq_colunas.txt” temos duas variáveis:
 - Uma delas numérica, da coluna 0 a 3.
 - A outra é categórica, da coluna 4 a 8.
 - Solução: Usar o **fatiamento** para separar as variáveis.

arq_colunas.txt

```
1001aaaaa  
1002bbbbb  
1003cccc  
1004ddddd  
1005eeee
```



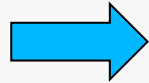
```
nomeArq = 'ARQ_COLUNAS.txt'  
f = open(nomeArq)  
for linha in f:  
    linha = linha.rstrip()  
    v1 = int(linha[:4]) #usei int p/ converter v1 p/ inteiro  
    v2 = linha[4:]  
    print(v1, v2)  
  
>>>  
1001 aaaaa  
1002 bbbbb  
1003 cccc  
1004 dddd  
1005 eeee
```

Método read()

- Lendo um arquivo inteiro para uma string – read()
 - Com o método **read()**, todos os caracteres do arquivo – do primeiro ao último, incluindo os “\n” – são armazenados em um único “stringão”

produtos.txt

```
1001Leite  
1002Biscoito  
1003Café  
1004Torradas  
1005Chá
```



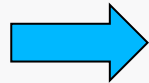
```
nomeArq = 'PRODUTOS.txt'  
f = open(nomeArq)  
conteudo = f.read()  
print(conteudo)  
  
>>>  
1001Leite  
1002Biscoito  
1003Café  
1004Torradas  
1005Chá
```


Método readlines()

- Lendo um arquivo inteiro para uma lista de strings – `readlines()`
 - Com o método **`readlines()`**, produz-se uma lista de strings.
 - Cada elemento da lista consistirá em uma linha do arquivo
 - * * **CUIDADO**: os caracteres “\n” de cada linha também são levados.

produtos.txt

```
1001Leite  
1002Biscoito  
1003Café  
1004Torradas  
1005Chá
```



```
nomeArq = 'PRODUTOS.txt'  
f = open(nomeArq)  
conteudo = f.readlines()  
print(conteudo)  
  
>>>  
['1001Leite\n', '1002Biscoito\n', '1003Café\n',  
'1004Torradas\n', '1005Chá\n']
```

Gravando Arquivos (1/5)

• I. Gravando um Arquivo (1/2)

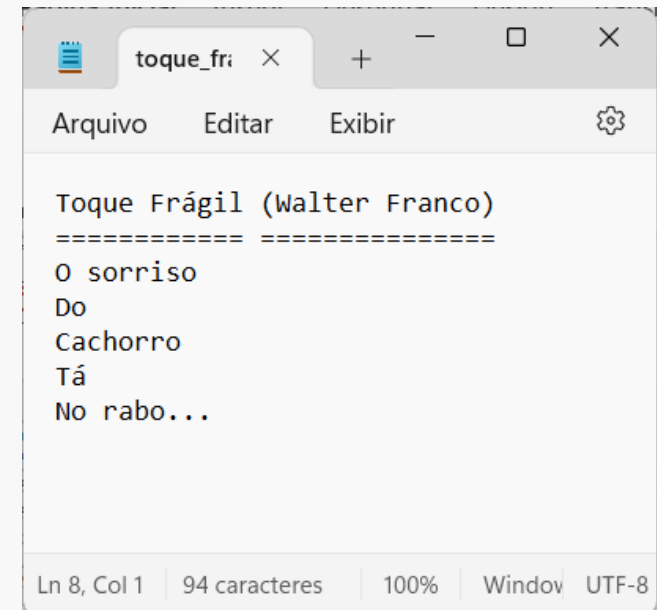
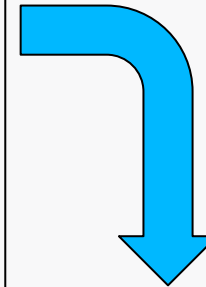
- Você deve abrir o arquivo utilizando o modo “w”.
 - **CUIDADO 1**: caso um arquivo com mesmo nome exista, ele será destruído!
 - **CUIDADO 2**: se quiser gravar com o encoding UTF-8, é preciso usar o `open()` especificando `encoding = 'utf-8'`
- Quando a gravação terminar, é necessário utilizar a função `close()` para fechar o arquivo.
- A função `write()` é utilizada para gravar uma linha.
 - **IMPORTANTE**: ela só consegue gravar strings. Se tiver algum dado numérico você terá que **converter para string**.
 - **Obs.:** sempre que você quiser uma quebra de linha, deverá especificar explicitamente o “\n”

Gravando Arquivos (2/5)

• I. Gravando um Arquivo (2/2)

```
fout = open('toque_fragil.txt', 'w', encoding = "utf-8")
msg1 = "O sorriso\n"
msg2 = "Do\n"
msg3 = "Cachorro\n"
msg4 = "Tá\n"
msg5 = "No rabo...\n"

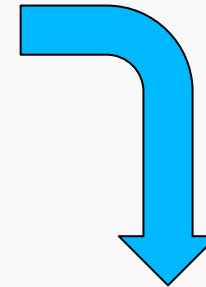
fout.write("Toque Frágil (Walter Franco)\n")
fout.write("===== ")
fout.write("\n")
fout.write(msg1 + msg2 + msg3 + msg4 + msg5)
fout.close()
```



Gravando Arquivos (3/5)

- II. Inserindo Linhas no Final de um Arquivo Existente
 - Você deve abrir o arquivo utilizando o modo “a” (append).

```
fout = open('C:/CursoPython/toque_fragil.txt', 'a')  
fout.write("#####\n")  
fout.write("*****\n")  
fout.close()
```

A screenshot of a text editor window titled 'toque_fragil.txt - Bloco de N...'. The window has a menu bar with 'Arquivo', 'Editar', 'Formatar', 'Exibir', and 'Ajuda'. The text content is as follows:
Toque Frágil (Walter Franco)
=====
O sorriso
Do
Cachorro
Tá
No rabo...

The status bar at the bottom shows 'Ln 10, 100%', 'Windows (CRLF)', and 'ANSI'.

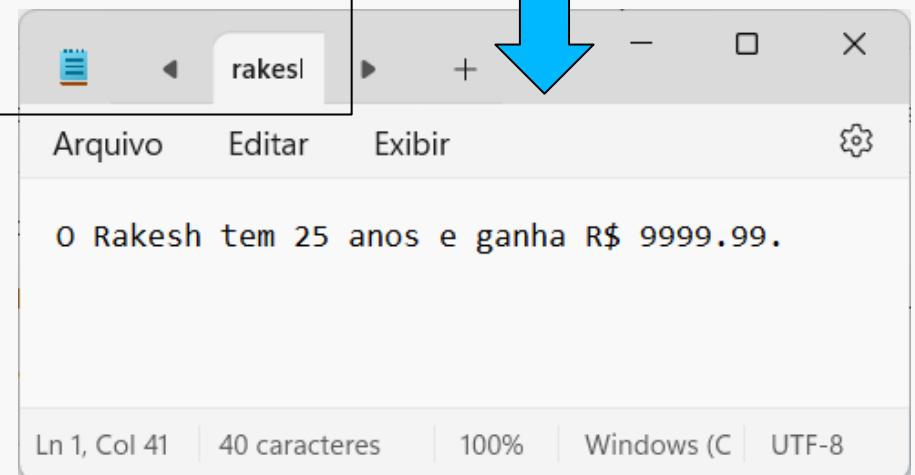
Gravando Arquivos (4/5)

- Como gravar dados numéricos?
 - Como `write()` aceita apenas strings, você precisa:
 - converter qualquer dado numérico para string se quiser gravá-lo.
 - Por exemplo, você pode colocar numa f-string

```
fout = open('rakesh.txt', 'w', encoding = "utf-8")
nome = "Rakesh"
idade = 25
salario = 9999.99

msg = f"O {nome} tem {idade} anos e ganha R$ {salario}."
fout.write(msg)

fout.close()
```



Gravando Arquivos (5/5)

- Como gravar dados numéricos?
 - Outra forma, dessa vez com concatenação e str()

```
fout = open('rakesh.txt', 'w', encoding = "utf-8")  
nome = "Rakesh"  
idade = 25  
salario = 9999.99  
  
fout.write("O " + nome + " tem " + str(idade) + " anos e ganha R$ " + str(salario))  
  
fout.close()
```

