Introdução à linguagem Python

Thiago Martins

Obtendo dados de arquivos CSV

- Comma-Separated Values – “valores separados por vírgulas”.
- Não é exatamente um “formato”.
- Historicamente, armazena-se números como dígitos decimais separados por vírgula desde FORTRAN.
- Planilhas populaziaram CSV como meio de intercâmbio de dados.

Exemplo:

https://dados.educacao.sp.gov.br/sites/default/files/VW_ALUNOS_ESTRANGEIROS_POR_NACIONALIDADE_20190517.csv

```
NOMEDEP;DE;DISTR;MUN;TIPOESC;COD_ESC;NOMESC;DS_PAIS;Nº alunos
ESTADUAL - OUTROS;CAMPINAS LESTE;CAMPINAS LESTE;CAMPINAS;27;0045949;CAMPINAS COLEGIO TECNICO DE - UNICAMP;França;1
ESTADUAL - OUTROS;CENTRO OESTE;RAPOSO TAVARES;SAO PAULO;11;0438820;RAPOSO TAVARES ETEC;Argentina;1
ESTADUAL - OUTROS;CENTRO SUL;BELA VISTA;SAO PAULO;11;0299364;MARIA AUGUSTA SARAIVA DRA ETEC;Bolívia;2
ESTADUAL - OUTROS;SOROCABA;SOROCABA;SOROCABA;11;0016354;RUBENS DE FARIA E SOUZA ETEC;Peru;1
ESTADUAL - OUTROS;TAUBATE;CACAPAVA;CACAPAVA;11;0014308;MACHADO DE ASSIS ETEC;Portugal;1
ESTADUAL - SE;ADAMANTINA;ADAMANTINA;ADAMANTINA;06;0985429;CEL JTO A EE FLEURIDES CAVALINI MENECHINO PROFA;Japão;3
ESTADUAL - SE;ADAMANTINA;ADAMANTINA;ADAMANTINA;08;0031112;FLEURIDES CAVALLINI MENECHINO PROFA;Venezuela;1
ESTADUAL - SE;ADAMANTINA;DRACENA;DRACENA;08;0031264;9 DE JULHO;Equador;2
ESTADUAL - SE;AMERICANA;AMERICANA;AMERICANA;08;0017176;MAGI MONSENHOR;Espanha;1
```

Obtendo dados de arquivos CSV

- Em Python há o pacote `csv`.
- Leitura de csv's:
  - `csv.reader(arquivo, dalect='excel', **`*fmtparams*`)`
    - Cria um objeto leitor de linhas para o arquivo (deve necessariamente ser aberto em modo texto).
    - As diversas combinações de possibilidades para a formatação do csv podem ser combinadas na forma de um *dialeto*. O dialeto padrão é "excel".
    - É possível especificar diretamente os parâmetros de formatação
- Objeto é *iterável* e representa uma sequência de *linhas*

```
with open("test.csv", "rt") as arq:
     cr = csv.reader(arquivo)
     for linha in cr:
        print(linha)
```

Obtendo dados de arquivos CSV - formatação

nome	descrição	Valor padrão
`delimiter`	Caractere que separa campos	,
`doublequote`	Como tratar caracteres especiais *dentro* de campos? True: aspas entre aspas. False: São precedidos do caractere de *escape*.	True
`escapechar`	Caractere que precede caracteres especiais quando doublequote é False	None
`lineterminator`	Caracteres usados para separar linhas (ignorado por reader, usado para escrita)	“\n\r”
`quotechar`	Caractere usado como aspas	“
`quoting`	Especifica o que o leitor deve esperar de campos entre aspas. Valores possíveis: `csv.QUOTE_ALL`: Aspas em todos os campos. `csv.QUOTE_MINIMAL`: Aspas apenas em campos contendo caracteres especiais. `csv.QUOTE_NONNUMERIC`: Aspas em todos os campos não numéricos. `csv.QUOTE_NONE`: Aspas jamais são esperadas.	`QUOTE_MINIMAL`
`skipinitialspace`	Ignora espaços imediatamente após o delimitador	False
`strict`	Se verdadeiro, lança uma exceção caso o arquivo esteja mal-formatado	False

Obtendo dados de arquivos CSV

- Mais sobre leitura de csv's:
  - `csv.DictReader(arquivo, fieldnames=None,...)`
    - Cria um objeto leitor de linhas que traduz cada linha em um *dicionário.*
    - Os nomes de cada entrada do dicionário são obtidos da *primeira linha* do csv ou da sequência passada em fieldnames.
    - Parâmetros restantes funcionam como em reader.
- Exemplo:

```
recurso = urllib.request.urlopen("https://dados.educacao.sp.gov.br/sites/default/files/
VW_ALUNOS_ESTRANGEIROS_POR_NACIONALIDADE_20190517.csv")

leitorcsv = csv.DictReader(io.TextIOWrapper(recurso,encoding='utf-8'), delimiter=";")

dicts = list(leitorcsv)

print(dicts[0]['DE')
```

XML

- Extensible Markup Language.
- Linguagem de Markup Genérica
- Bastante poderosa... E complexa...
- Até a década passada, principal padrão de intercâmbio de dados na web.
  - XLST, SOAP, Ajax...
- Hoje caindo em desuso como formato de intercâmbio de dados, mas aplicações legadas ainda são abundantes!
- Um documento xml segue uma estrutura *hierárquica* de nós.
  - Existe um único nó raiz da hierarquia.
- Cada nó contém uma *tag* (etiqueta).
- Opcionalmente, atributos e conteúdo.

Lendo XML em Python

- Processar um documento xml é uma tarefa complexa!
- XMLs podem incluir outros xmls, conter dados comprimidos, regras de validação específicas podem se aplicar
- Isso tudo é **muito** complexo.
- DIVERSAS bibliotecas em python processam xml, de modos distintos
    - ... Boa parte delas usa uma interface similar

Lendo XML em Python

- Módulo xml.etree.ElementTree
- Função `parse(caminho)`
  - Abre o arquivo especificado no caminho e tenta interpretá-lo como um documento xml.
  - Retorna um objeto do tipo ElementTree.
  - Use o método .getroot() para obter o elemento raiz do xml.
- A função `fromstring(string)` faz algo similar com um documento que esteja na memória em forma de string. Retorna diretamente o elemeto raiz.

Lendo XML em Python

- Elementos contém os campos:
  - tag: A etiqueta do nó (string).
  - attrib: Os atributos do nó (na forma de dicionário string->string)
  - text: O conteúdo do nó
- Adicionalmente, elementos são objetos armazenadores do tipo sequenciável. cada entrada corresponde a um nó filho.
  - Objeto é *iterável* e *indexável*
  - len(no) : Quantidade de nós filhos.
  - no[i]: Retorna o i-gésimo filho.

Lendo XML em Python

- Encontrando nós pela tag:
  - no.iter(tag): Iterador sobre filhos do nó com a tag especificada.
  - no.find(tag): Retorna o primeiro filho do nó com a tag especificada.
  - no.findall(tag): Retorna uma lista de *todos* os filhos do nó com a tag especificada.

Expressões Regulares em Python

- Módulo `re`
- Funções search(regexp, str) e match(regexp, str).
  - Buscam o primeiro casamento entre regexp e str.
  - Retornam um objeto do tipo match.
  - A diferença: search procura a partir de qualquer ponto da string, match, só a partir do início.
- Função finditer(regexp, str)
  - Encontra TODOS os casamentos *sem* sobreposição entre regexp e str na forma de um objeto iterador.

Expressões Regulares em Python

- Objetos do tipo match.
  - Representam um casamento entre uma expressão regular e uma string.
  - Métodos .start(), .end(). Representam o início e fim de um casamento.
  - Método .span(). Retorna uma tupla com o índice de início e fim do casamento
  - Método .str(): A string na qual se deu o casamento
  - Método .group(i). Retorna um sub-objeto match correspondente ao casamento da *i*-gésima sub-expressão (entre parênteses). O índice 0 retorna o objeto atual, o 1 a primeira sub-expressão, assim por diante.

Expressões Regulares em Python

- Expressões compiladas.
  - re.compile(expr): Gera um objeto do tipo pattern.
  - Mais eficiente quando uma mesma expressão deve ser testada múltiplas vezes.
- Objeto Pattern:
  - Métodos search(string) e match(string). Similares a re.search(expr, string) e match(expr,string)
  - Método finditer(string). Similar a re.finditer(re, string)
  - Método fullmatch(string). Retorna um objeto match somente se a string *inteira* casa com a expressão.