

# Programação Python

## Aula 10: SQL, JSON e XPath

Prof. Eduardo Corrêa Gonçalves

30/03/2021

# Sumário

## Introdução

Por que SQL, JSON e XPath são importantes?

## SQL

## JSON

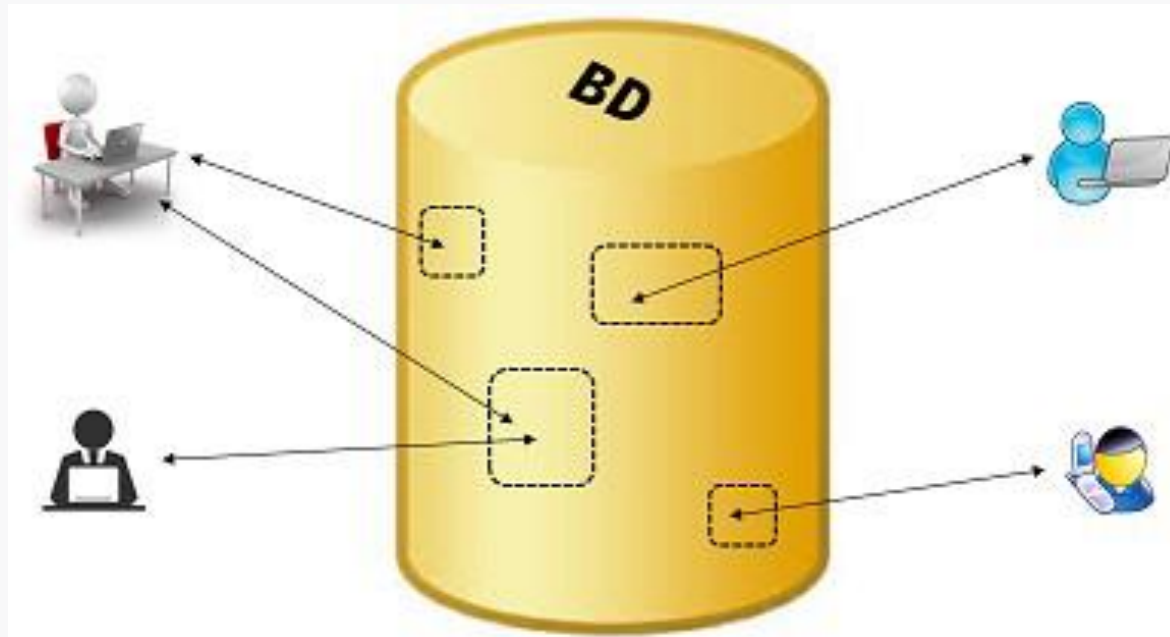
## XPath

# Introdução

- **Por que SQL, JSON e XPath?**
  - Essas tecnologias são utilizadas em diferentes problemas:
  - **SQL:**
    - Linguagem padrão para manipular bancos de dados relacionais (modelo dominante no mercado).
  - **JSON:**
    - Linguagem mais popular para a representação de informações no formato texto.
    - Adotada pelos bancos de dados orientados a documentos (padrão ascendente no mercado).
  - **XPath:**
    - Linguagem importante em processos de Web Scraping (recuperação automática do conteúdo de páginas da Internet)

# SQL (1/5)

- Banco de Dados (BD)
  - **Repositório central** de informações que podem ser consultadas e/ou atualizadas por **diversos usuários** simultaneamente.
  - São estruturados “dentro” de um software especial chamado Sistema Gerenciador de Bancos de Dados (SGBDs).
    - Ex.: Oracle, MySQL, MongoDB, PostgreSQL, Neo4J, MS SQL Server, sqlite, ...



# SQL (2/5)

- Banco de Dados (BD)
  - Os **SGBDs relacionais** são o padrão dominante do mercado.
    - Ou seja, é o modelo de SGBD mais encontrado nas empresas.
    - Exemplos: Oracle, Microsoft SQL Server, MySQL e sqlite.
    - Ele baseia-se no uso de tabelas para representar os dados.
  - SGBDs relacionais (independente do fabricante) são manipulados através de uma linguagem desenvolvida especialmente para este ambiente, denominada **SQL** (*Structured Query Language*).
    - O modelo relacional e a linguagem SQL são apresentados em detalhes na disciplina “Bases de Dados”.
    - Nesta aula, apresentaremos apenas **um exemplo** de programa Python que acessa um banco de dados usando SQL.

# SQL (3/5)

- **Base de Dados Exemplo: “Rh.db”**
  - É uma base de dados **sqlite** (software que “emula” um SGBD relacional)
  - Possui duas tabelas: *Profissao* e *Funcionario*.
  - Veja que ambas estão vinculadas pelo “id da profissão”

*Profissao*

<u>id</u>	<i>continente</i>
1	Engenheiro
2	Desenvolvedor
3	Cientista de Dados
4	Minerador de Dados
5	Matemático

*Funcionario*

<u>mat</u>	<i>nome</i>	<i>idade</i>	<i>sexo</i>	<i>id_prof</i>
M01	George	58	M	5
M02	Jane	32	F	3
M03	Aldous	40	M	3
M04	Thomas	28	M	1
M05	Mary	43	F	NULL

# SQL (4/5)

- Exemplo – recupera todos os dados da tabela *Funcionario*
  - Usando **import sqlite**, podemos acessar BDs do sqlite no Python

```
#P1- Modelo básico para interação Python x SQLite via SQL
import sqlite3

nomeBD = 'RH.db';
minha_conn = sqlite3.connect(nomeBD) #(1)-Conecta com o BD

c = minha_conn.cursor(); #(2)- Executa o comando SQL
c.execute('SELECT * FROM Funcionario')

for linha in c: print(linha) #(3)-Exibe os resultados

minha_conn.close() #(4)-Fecha a conexão
```

```
('M01', 'George', 58, 'M', 5)
('M02', 'Jane', 32, 'F', 3)
('M03', 'Aldous', 40, 'M', 3)
('M04', 'Thomas', 28, 'M', 1)
('M05', 'Mary', 43, 'F', None)
```

# SQL (5/5)

- **Observações**

- A biblioteca **'sqlite'** já faz parte do Python padrão. Ela é específica para o SGBD relacional sqlite.
- As distribuições do Python voltadas para ciência de dados costumam vir equipadas com bibliotecas para os SGBDs mais populares.
  - Exemplos:
    - **MongoDB** – biblioteca **'pymongo'**
    - **mySQL**: biblioteca **'mysql.connector'**
    - **redis**: biblioteca **'redis'**



# JSON (1/5)

- **O que é JSON** (*JavaScript Object Notation*)?
  - É um formato leve para a representação de dados.
    - É o formato mais adotado para a troca de informações entre aplicativos, APIs e camadas de software;
    - É fácil para pessoas lerem e entenderem (autodescritivo);
    - É fácil para softwares interpretarem (realizarem o parsing).
    - Não se limita a dados tabulares, como o CSV.
  - **Ex.:** Representação de um filme.

```
{  
  "titulo": "JSON versus CSV",  
  "resumo": "o duelo de dois modelos de representação de informações",  
  "ano": 2012,  
  "generos": ["aventura", "ação", "thriller"],  
  "elenco": ["Json Statham", "Array Seagal"]  
}
```

# JSON (2/5)

- **Sintaxe**

- Para cada valor representado atribui-se um **campo** (*field*)
  - "nome": "George Best"
- Apenas **4 tipos básicos**: numérico, booleano, string e null
  - "idade": 21
  - "altura": 1.75
  - "craque": true
  - "clube": "Manchester United"
  - "obediencia\_tatica": null

# JSON (3/5)

- **Sintaxe**

- A partir dos tipos básicos, é possível construir o tipo complexo **objeto** { }

```
{  
  "nome": "George Best",  
  "idade": 21,  
  "altura": 1.75,  
  "craque": true,  
  "clube": "Manchester United",  
  "obediencia_tatica": null  
}
```

# JSON (4/5)

- **Sintaxe**

- Outro tipo complexo é o **array** [ ]

- Exemplo 1 - array de strings: ["Ação", "Aventura", "Suspense"]

- Exemplo 2 - matriz de inteiros: 

```
[  
    [1, 5],  
    [-1, 9],  
    [1000, 0]  
]
```

# JSON (5/5)

- **Sintaxe**

- Objetos podem armazenar outros objetos e arrays.
- Arrays podem armazenar outros arrays e objetos.
- Assim, o JSON consegue representar praticamente todo tipo de base de dados (*ao contrário do formato CSV!!!*)

```
[
{
  "titulo": "JSON versus XML",
  "resumo": "o duelo de dois modelos de representação de informações",
  "ano": 2012,
  "generos": ["aventura", "ação", "thriller"],
  "elenco": ["Json Statham", "Array Seagal"]
},
{
  "titulo": "JSON James",
  "resumo": "A história de uma lenda do velho oeste",
  "ano": 2019,
  "generos": ["western"],
  "elenco": ["Json Wayne"]
}
]
```

# JSON no Python (1/7)

- JSON e as EDs Python
  - Os objetos e arrays JSON podem ser **mapeados diretamente** para EDs nativas do Python.
  - objeto JSON  $\Leftrightarrow$  dicionário
  - array JSON  $\Leftrightarrow$  lista
  - Isso é ótimo!
  - Assim podemos manipular qualquer JSON em memória usando o Python padrão!
    - Veja que o arquivo JSON ao lado é igual a uma lista de dicionários...

“empregados.json”

```
[  
  {  
    "cod": "E01",  
    "dept": "D01",  
    "nome": "João",  
    "inicial-meio": "S.",  
    "sobrenome": "Santos"  
  },  
  {  
    "cod": "E02",  
    "dept": "D01",  
    "nome": "Ana",  
    "sobrenome": "Ferraz"  
  }  
]
```

# JSON no Python (2/7)

- Módulo 'json'
  - No Python, o módulo built-in 'json' oferece métodos para importar e manipular arquivos JSON:
  - Abaixo a receita de bolo para importar um JSON para a memória:

*# importa o módulo*

**import json**

*# parsing do documento:*

*# carrega o arquivo "empregados.json" para uma ED.*

*# no caso de "empregados.json", carrega para uma lista de dicionários*

**with open("empregados.json") as f:**

**emps = json.load(f)**

# JSON no Python (3/7)

- Imprimindo o documento

```
print (type(emps))  # neste caso, é uma lista  
                    # (de dicionários)
```

```
print(emps)
```

```
print("total de empregados: ", len(emps))
```

```
>>>
```

```
<class 'list'>
```

```
[{'cod': 'E01', 'dept': 'D01', 'nome': 'João', 'inicial-  
meio': 'S.', 'sobrenome': 'Santos'}, {'cod': 'E02',  
'dept': 'D01', 'nome': 'Ana', 'sobrenome': 'Ferraz'}]
```

```
total de empregados: 2
```

```
[  
  {  
    "cod": "E01",  
    "dept": "D01",  
    "nome": "João",  
    "inicial-meio": "S.",  
    "sobrenome": "Santos"  
  },  
  {  
    "cod": "E02",  
    "dept": "D01",  
    "nome": "Ana",  
    "sobrenome": "Ferraz"  
  }  
]
```



# JSON no Python (4/7)

- Iterando sobre os elementos
  - Normal, como fazemos com qualquer lista...

**for** empregado **in** emps:

**print**(empregado)

>>>

{'cod': 'E01', 'dept': 'D01', 'nome': 'João', 'inicial-meio': 'S.', 'sobrenome': 'Santos'}

{'cod': 'E02', 'dept': 'D01', 'nome': 'Ana', 'sobrenome': 'Ferraz'}

```
[  
  {  
    "cod": "E01",  
    "dept": "D01",  
    "nome": "João",  
    "inicial-meio": "S.",  
    "sobrenome": "Santos"  
  },  
  {  
    "cod": "E02",  
    "dept": "D01",  
    "nome": "Ana",  
    "sobrenome": "Ferraz"  
  }  
]
```

# JSON no Python (5/7)

- Recuperando um elemento específico
  - Basta indexar...
  - Também é possível fatiar ou fazer qualquer operação de lista.

*# imprime o nome e sobrenome do segundo empregado*

```
print(emps[1]['nome'], emps[1]['sobrenome'])
```

```
>>>
```

```
Ana Ferraz
```

```
[  
  {  
    "cod": "E01",  
    "dept": "D01",  
    "nome": "João",  
    "inicial-meio": "S.",  
    "sobrenome": "Santos"  
  },  
  {  
    "cod": "E02",  
    "dept": "D01",  
    "nome": "Ana",  
    "sobrenome": "Ferraz"  
  }  
]
```

# JSON no Python (6/7)

- **Buscando um campo específico**
  - **Exemplo:** recuperar todos os nomes
  - Mais uma vez, é só usar o Python padrão...

```
nomes = [pessoa['nome'] for pessoa in emps]  
print(nomes)
```

```
>>>  
['João', 'Ana']
```

```
[  
  {  
    "cod": "E01",  
    "dept": "D01",  
    "nome": "João",  
    "inicial-meio": "S.",  
    "sobrenome": "Santos"  
  },  
  {  
    "cod": "E02",  
    "dept": "D01",  
    "nome": "Ana",  
    "sobrenome": "Ferraz"  
  }  
]
```

# JSON no Python (7/7)

- **Fazendo buscas**

- **Exemplo:** recuperar todos os funcionários com o sobrenome 'Santos'
- E, de novo, basta usar o Python padrão...

```
os_santos = [pessoa for pessoa in emps if pessoa["sobrenome"]=="Santos"]  
print(os_santos)
```

```
>>>
```

```
[{'cod': 'E01', 'dept': 'D01', 'nome': 'João', 'inicial-meio': 'S.', 'sobrenome': 'Santos'}]
```

# JSON parser

- Na Internet existem muitos sites que fazem o parsing de documentos JSON. Basicamente, eles testam se o seu documento está correto.
  - Ex.: <http://json.parser.online.fr/>

The screenshot shows a web browser window titled "Json Parser Online - Mozilla Firefox". The address bar displays "json.parser.online.fr". The website header includes the title "Json Parser Online", a link to "You like it? Support it! Donate", and buttons for "Try out Beta!", "Samples", and "Options".

The main content area is divided into three panels:

- Input JSON:** A text area containing a valid JSON array of two objects:

```
[
{
  "cod": "E01",
  "dept": "D01",
  "nome": "João",
  "inicial-meio": "S.",
  "sobrenome": "Santos"
},
{
  "cod": "E02",
  "dept": "D01",
  "nome": "Ana",
  "sobrenome": "Ferraz"
}
]
```
- String parse:** A text area showing the JSON input with syntax highlighting and collapsible nodes (indicated by square icons).
- JS eval:** A text area showing the JSON input converted into a JavaScript array of objects, also with syntax highlighting and collapsible nodes.

At the bottom of the page, the footer contains the text "Author: Olivier Cuenot" on the left and a list of links: "FAQ", "Privacy Policy", "Changelog", "Donation via PayPal", and "Donation in Bitcoins" on the right.

# XPath (1/8)

- O que é XPath?
  - Trata-se de uma **linguagem de consulta** que permite extrair partes de um documento da Internet (XML ou HTML).
  - É uma linguagem importante em processos de **Web Scraping**.

## Web Scraping

- Processo em que utilizamos um programa que “**finge**” ser um **navegador Web** com o intuito de recuperar automaticamente páginas da Internet (esses programas são chamados de **robôs**)
- Após as páginas serem recuperadas, seus dados são examinados para que sejam extraídas informação de interesse.
- Instituições como **IBGE** e **FGV** utilizam Web Scraping para coletar informações relevantes para o cálculo de índices.
- <https://agenciadenoticias.ibge.gov.br/agencia-noticias/2012-agencia-de-noticias/noticias/26796-ibge-comeca-a-divulgar-inflacao-com-novo-calculo-e-precos-coletados-por-robos>

## XPath (2/8)

- É possível implementar “robôs” simples em Python com o uso de pacotes da *standard library*:
  - **urllib**: recupera páginas da Internet.
  - **lxml**: pacote que oferece a XPath (entre outras coisas)
  - **re**: pacote de expressões regulares (não será coberto neste curso).
- Entretanto, para elaborar Web scrapers mais sofisticados, é preciso utilizar outros pacotes, que não fazem parte do Python padrão, tais como:
  - BeautifulSoup,
  - Scrapy
  - ...

# XPath (3/8)

- **EXEMPLO** – extraindo informações de uma página Web com XPath.
- Considere a página HTML localizada no endereço abaixo:

[https://web.ics.purdue.edu/~gchopra/class/public/pages/webdesign/05\\_simple.html](https://web.ics.purdue.edu/~gchopra/class/public/pages/webdesign/05_simple.html)

**A very simple webpage. This is an "h1" level header.**

**This is a level h2 header.**

This is a level h6 header. Pretty small!

This is a standard paragraph.

Now I've aligned it in the center of the screen.

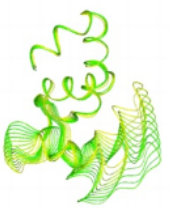
Now aligned to the right

**Bold text**

**Strongly emphasized text** Can you tell the difference vs. bold?


*Italics*

*Emphasized text* Just like Italics!



Here is a pretty picture:

Same thing, aligned differently to the paragraph:




---

**How about a nice ordered list!**

1. This little piggy went to market
2. This little piggy went to SB228 class
3. This little piggy went to an expensive restaurant in Downtown Palo Alto
4. This little piggy ate too much at Indian Buffet.
5. This little piggy got lost



# XPath (4/8)

- **EXEMPLO – extraindo informações de uma página Web com XPath.**
  - Abaixo apresentamos um trecho de seu código HTML:

```
<html>
<head>
<title>A very simple webpage</title>
<basefont size=4>
</head>

<body bgcolor=FFFFFF>

<h1>A very simple webpage. This is an "h1" level
header.</h1>

<h2>This is a level h2 header.</h2>

<h6>This is a level h6 header.  Pretty small!</h6>

<p>This is a standard paragraph.</p>

...
```

# XPath (5/8)

- **EXEMPLO – extraindo informações de uma página Web com XPath.**
- Suponha que desejamos extrair todos títulos de seções
  - Nesta página esses títulos são definidos com a tag <h2>

A very simple webpage. This is an "h1" level header.

This is a level h2 header.

This is a level h4 header. Pretty small!

This is a standard paragraph.

Now I've aligned it in the center of the screen.

Now aligned to the right

**Bold text**

**Strongly emphasized text** Can you tell the difference vs. bold?

*Italics*

*Emphasized text* Just like Italics!



Here is a pretty picture:

Same thing, aligned differently to the paragraph:



How about a nice ordered list!

1. This little piggy went to market
2. This little piggy went to SB228 class
3. This little piggy went to an expensive restaurant in Downtown Palo Alto
4. This little piggy ate too much at Indian Buffet.
5. This little piggy got lost

# XPath (6/8)

- **PASSO 1:** capturar a página e armazenar o seu código HTML em uma variável string.
  - Esse processo é feito com o pacote **urllib**:

```
import urllib.request

nomPagina = 'https://web.ics.purdue.edu/~gchopra/class/public/pages/webdesign/05_simple.html'

#"abre" a página
conteudo = urllib.request.urlopen(nomPagina)

#converte o seu conteúdo para uma string padrão
conteudo = conteudo.read().decode('utf-8')

print(conteudo) #imprime só para checar!
```

# XPath (7/8)

- **PASSO 2**: usa XPath para obter os dados de interesse.
  - Esse processo é feito com o pacote **lxml**:

```
from lxml import etree

#estrutura o documento HTML em uma árvore em memória
arvore = etree.HTML(conteudo)

#esse é o comando XPath que pega todos os textos associados à tag <h2>
r = arvore.xpath('//h2/text()')

print(r) #imprime os textos encontrados
```

```
['This is a level h2 header.', 'How about a nice ordered list!', 'Unordered list', 'Nested Lists!']
```

# XPath (8/8)

- Explicação do comando **"//h2/text()"**
  - **//** : é a instrução que ordena com que a busca seja feita em todo o documento HTML, a partir da raiz, independente da posição.
  - **h2** : esta é a tag HTML cujos textos queremos obter.
  - **text()** : para pegar os textos.
- Para aprender XPath (que tem muitos outros comandos!) é interessante ter o conhecimento prévio de outros conceitos, como **HTML** e **árvore DOM**.
  - Não temos tempo de fazer isso neste curso!!! Só em "Programação Avançada".
- Nesta aula apresentamos um pequeno exemplo, apenas para introduzir os alunos ao tema.

# Tarefa

- DOJO: processando um BD JSON

# Referências

- Corrêa, E. (2020). “Meu Primeiro Livro de Python”. V 2.0.0, edubd, 2020. (*capítulos 4 e 5*).
  - Disponível em: [https://github.com/edubd/meu\\_primeiro\\_livro\\_de\\_python](https://github.com/edubd/meu_primeiro_livro_de_python)
- Byron, D. (2020). “How to Use Python and XPath to Scrape Websites”.
  - Disponível em: <https://towardsdatascience.com/how-to-use-python-and-xpath-to-scrape-websites-99eaed73f1dd>