



# PYTHON – AULA 1

## WEB SCRAPING



emerson.abraham@fiap.com.br

**Emerson R. Abraham**

Professor

Doutor e Mestre em Engenharia de Produção; Especialização em Tecnologia da Informação; Bacharel em Administração de Empresas; e Certificado Profissional Scrum Master (PSM)

Professor e Pesquisador em cursos superiores como ADS, Sistemas de Informação, Automação Industrial e Gestão de Tecnologia da Informação; como pesquisador tenho trabalhado com estudos de modelagem e simulação relacionados a Lógica Fuzzy, Redes Neurais Artificiais e Algoritmos Genéticos.

# AGENDA

## Aula 1

Introdução – conceitos, arquitetura cliente – servidor, Introdução a HTML, CSS e JavaScript, URLLIB e Requests.

## Aula 2

Introdução ao BeautifulSoup, Expressões Lambdas, Expressões regulares (REGEX)

## Aula 3

BeautifulSoup, List Comprehension, Revisitando NUMPY e PANDAS, Introdução ao Selenium

## Aula 4

Selenium 4, Scraping de Imagens, Trabalhando com inputs em pesquisas, Introdução ao Docker.

## Aula 5

Selenium 4, Banco de dados SQLite, projeto final, considerações finais

# WEB SCRAPING

**INTRODUÇÃO – CONCEITOS**  
**ARQUITETURA CLIENTE – SERVIDOR**  
**INTRODUÇÃO A HTML, CSS E JAVASCRIPT**  
**URLLIB E REQUESTS.**



## Introdução - conceitos

- Em tradução literal **Web Scraping** significa “raspagem da web”; sendo um outro termo também muito empregado **Web Crawling**, que significa “rastreamento da web”.
- É uma prática de extração automatizada de dados da web.
- Não deve considerar um programa interagindo com uma **Application Programming Interface (API)**.
- Engloba uma variedade de técnicas de programação e tecnologias, tais como análise de dados, **parsing** e segurança da informação.

**Nota:** **Web Crawling** é utilizado por motores de busca para obtenção de informações mais genéricas; **Web Scraping**, faz a obtenção de informações específicas.

## Introdução – benefícios

- Existe uma enorme possibilidade de explorar a internet que não seja pelo navegador.
- Apesar dos navegadores serem muito bons para lidar com recursos visuais, mais intuitivos as pessoas, os web scrapers são ótimos para extrair um grande volume de dados.
- Além disso, os web scrapers podem acessar lugares que as ferramentas tradicionais, tais como buscadores não estão aptos, pois são limitados.
- Mesmo que hajam API's para extração, os limites para volume, taxa de requisição, tipo e formato dos dados podem ser insuficientes em relação ao seus objetivos.

## Funcionamento Web - HTTP

- Hypertext Transfer Protocol (HTTP) é o protocolo padrão para transferência de dados na web.
- Um protocolo é um conjunto de regras que duas ou mais entidades usam para se comunicarem.
- Este protocolo é baseado em texto, sendo assim, de fácil entendimento pelas pessoas.
- Outra propriedade interessante é ser extensível, ou seja, novas funcionalidades podem ser facilmente implementadas por meio de cabeçalhos (Header).



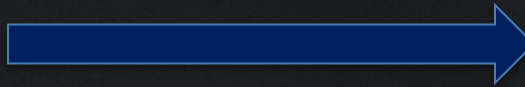
## Arquitetura Cliente - Servidor

- O protocolo HTTP funciona por meio de requisições e respostas.
- Ao buscar por uma página web, o cliente informa o endereço no navegador e o servidor devolve uma resposta.
- Esta arquitetura é definida por: **Cliente – Servidor**



Cliente

Request



Response



Servidor

## Status Code

Toda requisição feita por um cliente a um servidor resulta em uma resposta representado por um código HTTP (resultado da requisição), tenha ela sido processada com sucesso ou não.

Classe	Semântica
2xx	Indica que a requisição foi processada com sucesso.
3xx	Indica ao cliente uma ação a ser tomada para que a requisição possa ser concluída.
4xx	Indica erro(s) na requisição causado(s) pelo cliente.
5xx	Indica que a requisição não foi concluída devido a erro(s) ocorrido(s) no servidor.

## Status Code

Código	Descrição	Quando utilizar
200	OK	Em requisições GET, PUT e DELETE executadas com sucesso.
201	Created	Em requisições POST, quando um novo recurso é criado com sucesso.
206	Partial Content	Em requisições GET que devolvem apenas uma parte do conteúdo de um recurso.
302	Found	Em requisições feitas à URI's antigas, que foram alteradas.

## Status Code

400	Bad Request	Em requisições cujas informações enviadas pelo cliente sejam invalidas.
401	Unauthorized	Em requisições que exigem autenticação, mas seus dados não foram fornecidos.
403	Forbidden	Em requisições que o cliente não tem permissão de acesso ao recurso solicitado.
404	Not Found	Em requisições cuja URI de um determinado recurso seja inválida.
405	Method Not Allowed	Em requisições cujo método HTTP indicado pelo cliente não seja suportado.

## Status Code

406	Not Acceptable	Em requisições cujo formato da representação do recurso requisitado pelo cliente não seja suportado.
415	Unsupported Media Type	Em requisições cujo formato da representação do recurso enviado pelo cliente não seja suportado.
429	Too Many Requests	No caso do serviço ter um limite de requisições que pode ser feita por um cliente, e ele já tiver sido atingido.
500	Internal Server Error	Em requisições onde um erro tenha ocorrido no servidor.
503	Service Unavailable	Em requisições feitas a um serviço que esta fora do ar, para manutenção ou sobrecarga.



## Verbs HTTP

- Os verbos HTTP indicam qual ação está sendo requisitada pelo consumidor do serviço; os principais verbos são:

GET

HEAD

POST

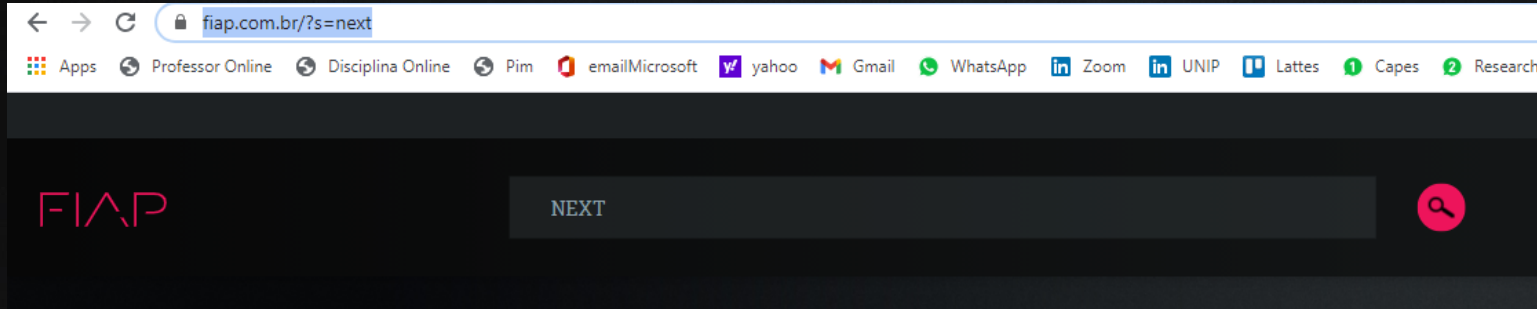
PUT

PATCH

DELETE

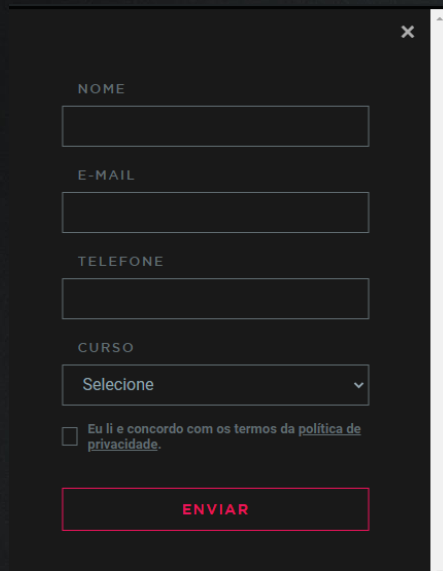
## Verbos HTTP

- + **GET** - os parâmetros são passados no cabeçalho da requisição e podem ser vistos pela URI. Utiliza-se mais em campos de busca ou listagem de itens.



## Verbos HTTP

**POST** - os parâmetros são passados no corpo da requisição HTTP, escondendo eles da URI. Usado em formulário de cadastro.



A screenshot of a registration form. The form is white with a black border and a close button (X) in the top right corner. It contains the following fields:

- NOME: A text input field.
- E-MAIL: A text input field.
- TELEFONE: A text input field.
- CURSO: A dropdown menu with the text "Selecione" and a downward arrow.
- Eu li e concordo com os termos da [política de privacidade](#): A checkbox.
- ENVIAR: A red button with white text.

**Nota:** para proteger a aplicação precisamos utilizar a "versão segura" do HTTP, o HTTPS. Com esse protocolo é possível criptografar os dados enviados.

## Verbos HTTP

### HEAD

Idêntico ao GET, exceto que o servidor NÃO retorna um corpo de mensagem na resposta; geralmente usado para testar links de hipertexto.

### PUT

Usado para editar um recurso

### PATCH

Usado para editar parcialmente um recurso, sem a necessidade de enviar todos os atributos

### DELETE

Usado para excluir um recurso

## Verbos HTTP

Verbo HTTP	Ação
GET	Ler
POST	Criar
HEAD	Ler metadados
PUT	Atualizar
PATCH	Modificar parcialmente
DELETE	Excluir

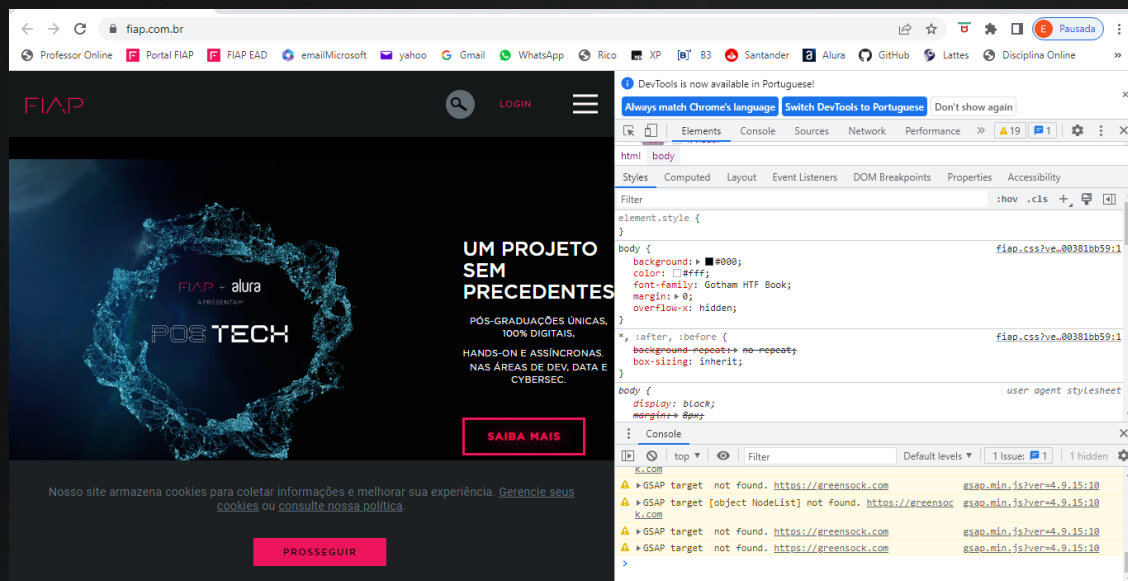


## Header

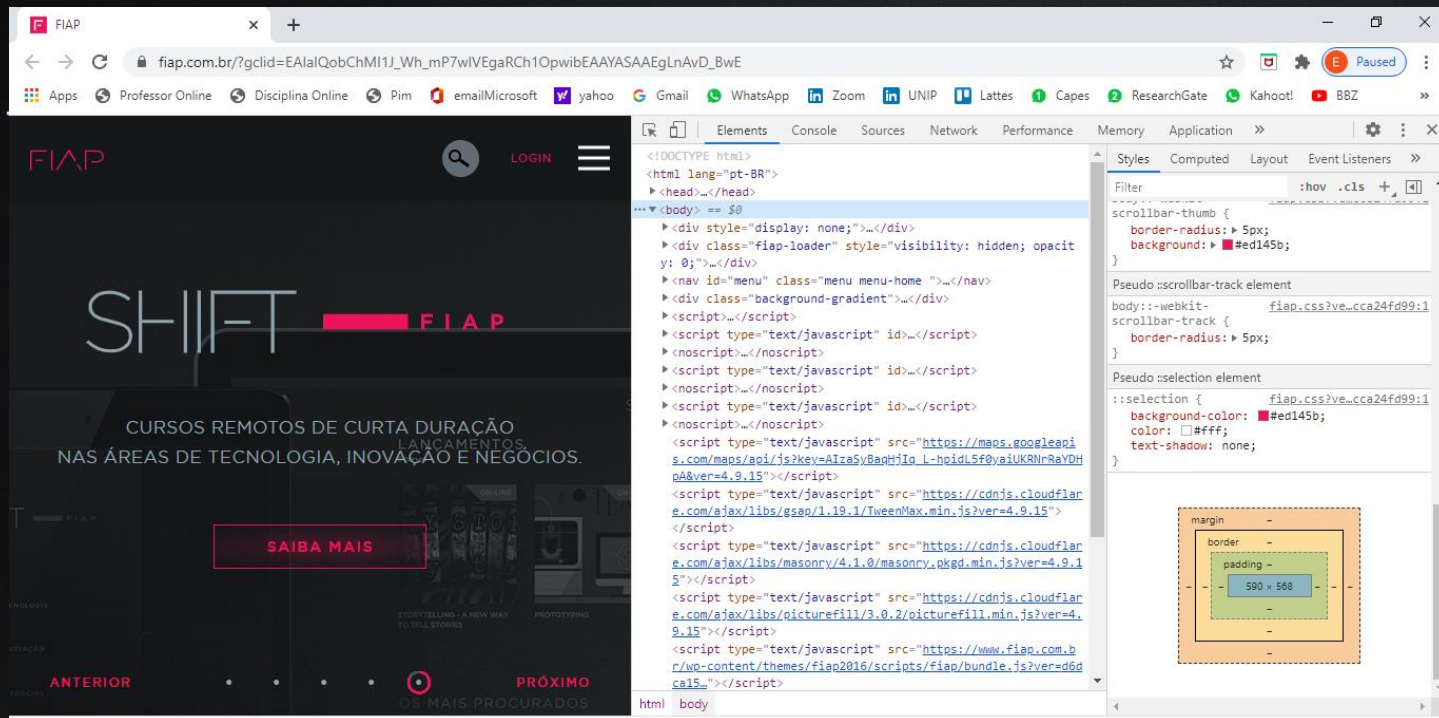
- O que significa dizer que o protocolo HTTP é extensível?
- Significa que podemos agregar informações durante as requisições e respostas.
- O Header nos confere essa possibilidade.
- Headers são pares de chave-valor que são enviados junto com cada solicitação e mensagem de resposta.

# Muito mais que textos e imagens

- Quando o servidor provê uma resposta para o cliente, os dados são renderizados pelo navegador, entretanto, existe muito mais do que textos e imagens.
- Vamos acessar a página da FIAP. **CTRL + SHIFT + i** para acessar ferramentas de desenvolvedor



# Muito mais que textos e imagens



# Muito mais que textos e imagens

The screenshot displays a web browser window with the FIAP website loaded. The website has a dark theme with a large, stylized 'N' logo in the center. The text on the page reads: "CONECTE-SE ÀS TECNOLOGIAS EXPONENCIAIS E À EVOLUÇÃO DIGITAL PARA REDEFINIR". Below the logo, there are navigation buttons labeled "ANTERIOR" and "PRÓXIMO".

The browser's developer tools are open, showing the Network tab. The list of requests includes several GIF files and a CSS file. The first request is highlighted, showing its details in the right-hand pane.

Name	Status	Type	Initiator	Size	Time	Waterfall
?id=101677741508579...	200	gif	fbevents.js:23	87 B	16 ms	
?id=101677741508579...	200	gif	fbevents.js:23	87 B	13 ms	
?id=101677741508579...	200	gif	fbevents.js:23	87 B	9 ms	
?id=101677741508579...	200	gif	fbevents.js:23	87 B	13 ms	
?id=101677741508579...	200	gif	fbevents.js:23	87 B	13 ms	
?id=101677741508579...	200	gif	fbevents.js:23	87 B	15 ms	
?id=101677741508579...	200	gif	fbevents.js:23	87 B	17 ms	
?id=101677741508579...	200	gif	fbevents.js:23	87 B	12 ms	
?id=101677741508579...	200	gif	fbevents.js:23	87 B	11 ms	
?id=101677741508579...	200	gif	fbevents.js:23	87 B	10 ms	
?id=101677741508579...	200	gif	fbevents.js:23	87 B	12 ms	
?id=101677741508579...	200	gif	fbevents.js:23	87 B	10 ms	
?id=101677741508579...	200	gif	fbevents.js:23	87 B	15 ms	
effects.png	200	png	fiap.css?ver=f...	8.3 kB	155 ms	

Summary: 14 requests | 9.4 kB transferred | 8.5 kB resources

# Muito mais que textos e imagens

The screenshot shows a web browser window with the FIAP website loaded. The browser's address bar displays the URL `fiap.com.br/?gclid=EAlaQobChMI1J_Wh_mP7wIVeGaRCh1OpwibEAAAYASAAEgLnAvD_BwE`. The browser's developer tools are open, showing the Network tab. The network tab displays a list of requests, with the selected request being a GET request to `https://fiap.com.br/.../effects.png`. The request details show a status code of 200, a remote address of `[2a03:2880:f1ff:83:face:b00c:0:25de]:443`, and a referrer policy of `strict-origin-when-cross-origin`. The response headers show `alt-svc: h3-29=":443"; ma=3600, h3-27=":443"; ma=3600`, `cache-control: no-cache, must-revalidate, max-age=0`, `content-length: 44`, `content-type: image/gif`, `cross-origin-resource-policy: cross-origin`, `date: Mon, 01 Mar 2021 20:36:17 GMT`, `expires: Mon, 01 Mar 2021 20:36:17 GMT`, `last-modified: Fri, 21 Dec 2012 00:00:01 GMT`, `server: proxygen-bolt`, and `set-cookie`. The browser's address bar also shows a search bar and a login button. The browser's tabs show the FIAP website and several other tabs, including Apps, Professor Online, Disciplina Online, Pim, emailMicrosoft, yahoo, Gmail, WhatsApp, Zoom, UNIP, Lattes, Capes, ResearchGate, Kahoot!, and BBZ.

FIAP

LOGIN

ANTERIOR

PRÓXIMO

19 requests | 9.8 kB transferred

Network

Filter

Blocked Requests

Name

Headers

Preview

Response

Initiator

Timing

Cookies

Request Method: GET

Status Code: 200

Remote Address: [2a03:2880:f1ff:83:face:b00c:0:25de]:443

Referrer Policy: strict-origin-when-cross-origin

Response Headers

alt-svc: h3-29=":443"; ma=3600, h3-27=":443"; ma=3600

cache-control: no-cache, must-revalidate, max-age=0

content-length: 44

content-type: image/gif

cross-origin-resource-policy: cross-origin

date: Mon, 01 Mar 2021 20:36:17 GMT

expires: Mon, 01 Mar 2021 20:36:17 GMT

last-modified: Fri, 21 Dec 2012 00:00:01 GMT

server: proxygen-bolt

set-cookie



## User-agent

- O *user-agent* é um pedaço de software que é usado para acessar um recurso web.
- O *user-agent* pode ser um browser, terminal, web crawler, bot, script, etc.
- É o cliente por trás do cliente em uma requisição.
- Responsável por interpretar a resposta e renderizar para o usuário

## User-agent

- Vamos testar o acesso de um site pelo terminal.
- Digite CMD na barra de pesquisa do Windows
- Utilize o comando curl

- Exemplo:

curl <https://www.freeimages.com/pt>

- Outra forma de acessar uma página web é por meio de API de testes. Vamos ver como funciona.

Acesse: <https://httpie.io/app>

httpie.io

GET nytimes.com/2023/0...

Environments

GET https://www.nytimes.com/2023/03/12/us/politics/mike-pence-donald-trump-gridiron.html

Send

Params

Headers 1

Auth

Body

</>

Request GET

Response 200


☒ Referrer https://www.google.com

☐ name value

▶ HTTP/1.1 200 OK (35 headers)

## ***‘History Will Hold Donald Trump Accountable’ for Jan. 6, Pence Says***

At a Washington dinner event, Mike Pence criticized the president he served under as well as Republicans who are minimizing the Capitol riot.

 Give this article



HTML  Preview

245KB, 1.1s, 6m ago

# Cookies

São fragmentos de dados que um servidor envia para o cliente. O cliente pode armazenar estes dados e enviá-los de volta na próxima requisição.

Normalmente, são usados para três propósitos:

- Tipificação: preferências de usuário, temas etc.
- Sessão do usuário: logins, carrinhos de compra etc.
- Rastreo: registro e análise do comportamento do usuário.



# HTML, CSS e JavaScript

Quando o cliente envia requisições para o servidor, este devolve códigos que utilizam as tecnologias HTML, CSS e Javascript.

- HTML - é uma linguagem de marcação (*markup language*) utilizada para estruturar o conteúdo da página web, tais como definir seções, divisões, formulários, cabeçalhos, rodapés, links, conteúdo semântico etc.
- CSS – linguagem de estilo; define como os elementos HTML serão exibidos, tais como posições, cores, efeitos visuais etc.
- Javascript – linguagem de programação que complementa o trio, provendo funcionalidade as páginas web, tais como validações, buscas, mapas, infográficos, formulários, animações etc



# HTML

- A linguagem de marcação HTML (*HyperText Markup Language*) utiliza tags

Exemplo:

```
<h1>FIAP</h1>
```

```
<p>Let's Rock the Future</p>
```

- Motores de busca utilizam certas marcações para entender melhor o conteúdo da página.
- Ao usar as *tags* corretas, auxiliamos os motores de busca entender melhor nosso conteúdo e assim ranquear a página de forma apropriada.

# CSS

A linguagem de estilos CSS (*Cascading Style Sheets*) pode ser utilizada junto com o código HTML por meio da *tag style* ou em um arquivo separado.

Exemplo:

```
<style>
```

```
h1 {
```

```
    color: white;
```

```
    text-align: center;
```

```
}
```

```
p {
```

```
    font-family: verdana;
```

```
    font-size: 20px;
```

```
}
```

```
</style>
```

## JavaScript

A linguagem de programação mais usada na web para definir o comportamento das páginas.

Exemplo:

```
<button type="button"
  onclick="document.getElementById('demo').innerHTML = Date()">Data e hora
</button>
```

**Hands ON**

**Hora de praticar!**

**Vamos exercitar um pouco de HTML, CSS e Javascript**

**Acesse:**

**<https://www.w3schools.com/>**

# URLLIB

- Chegou o momento do primeiro scraping com Python.
- Crie um novo projeto denominado `scraping_urllib`.
- Vamos utilizar a biblioteca `urllib` para acessar a página Quotes to Scrape

```
from urllib.request import urlopen  
  
url = "https://quotes.toscrape.com/"  
  
response = urlopen(url)  
  
print(response.status)
```



## URLLIB

Para exibir o conteúdo da página utilizamos as funções *read* e *decode*

```
content = response.read()

print(content.decode("utf-8"))
```

Experimente exibir o conteúdo sem aplicar a função *decode*; perceba a diferença.

## URLLIB

Vamos refatorar o conteúdo para uma sintaxe mais enxuta e elegante.

```
with urlopen(url) as response:  
    content = response.read()  
    print(content.decode("utf-8"))
```

- Este modo de escrita evita *exceptions* durante as requisições e fecha automaticamente objetos de resposta.

## Requests: HTTP for Humans™

- Requests é uma biblioteca de alto nível que proporciona um modo mais conveniente de utilizar HTTP em Python.
- Não há necessidade de adicionar strings de consulta manualmente as URLs ou codificar dados POST.
- O Pool de conexão HTTP é 100% automático.



Vamos acessar a página da biblioteca em: [Pythonhttps://requests.readthedocs.io/en/latest/](https://requests.readthedocs.io/en/latest/)

## Requests: HTTP for Humans™

- Para utilizar a biblioteca, faça o importe do pacote **Requests**
- Crie um novo arquivo denominado `scraping_requests` e codifique conforme mostrado abaixo

```
import requests

url = "https://quotes.toscrape.com/"

response = requests.get(url)
print(response.status_code)
print(response.text)
```

## Requests: Headers

- Percebeu que não precisamos utilizar a função decode? A biblioteca trata a codificação de forma elegante.
- Vamos a mais uma funcionalidade. Implemente a função headers; o resultado é um dicionário de dados de cabeçalho.

```
print(response.headers)
```

```
{'Date': 'Tue, 14 Mar 2023 21:01:46 GMT', 'Content-Type': 'text/html; charset=utf-8',  
'Content-Length': '11053', 'Connection': 'keep-alive', 'Strict-Transport-Security': 'max-  
age=0; includeSubDomains; preload'}
```



## Requests: JSON

- Outro recurso muito importante de se obter através das requisições é o arquivo JSON com os dados.

```
print(response.json( ))
```

Nota: essa solicitação pode retornar erros:

1. Citação não compatível com JSON
2. Arquivo JSON vazio
3. Saída XML (forma antiga de se trafegar dados)

## Requests: JSON

- Podemos acessar diversas API's que irão nos retornar dados no formato JSON.
- Neste caso, é interessante separarmos alguns parâmetros para as requisições, conforme mostrado no código abaixo.

```
url = "https://api.coinbase.com/v2/exchange-rates"
parameter = {"currency": "BTC"}
response = requests.get(url, parameter)
print(response.json()["data"]["rates"])
```

## Requests: Autenticação e Autorização

- Em alguns casos vamos precisar interagir com API's que não são públicas. Assim sendo, vamos precisar de autenticação e autorização.
- Autenticação: quem nós somos
- Autorização: o que nós podemos fazer

```
API_KEY = "Sua chave para a API"
url="https://api.exchange.coinbase.com/fills"
headers = {"Authorization": f"Bearer{API_KEY}",
           "Content-Type": "application/json"}

response = requests.get(url, headers=headers)
print(response)
```

- Bearer Token é um string criptografada, geralmente gerada pelo servidor em resposta a uma solicitação de login.

## Requests: POST, PUT, PATCH e DELETE

- Vamos acessar a API HTTPBin ([httpbin.org](http://httpbin.org)) e testar os demais verbos HTTP.

```
response = requests.delete("https://www.httpbin.org/delete").json()
print(response)
```

```
response = requests.post("https://www.httpbin.org/post").json()
print(response)
```

```
response = requests.patch("https://www.httpbin.org/patch").json()
print(response)
```

```
response = requests.put("https://www.httpbin.org/put").json()
print(response)
```

- **Requests: POST**

- Podemos enviar os dados no corpo da requisição por meio de um dicionário (chave-valor), compatível com JSON.

```
response = requests.post("https://www.httpbin.org/post",
    data={"M": "Jose",
        "F": "Maria"}).json()

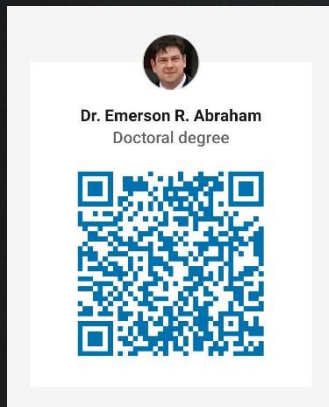
print(response)
```



## BIBLIOGRAFIA BÁSICA

- BEAZLEY, David. *Python Essential Reference*, 2009.
- BEK, ANDY. *The Ultimate Web Scraping With Python Bootcamp 2023*.
- BHARGAVA, ADITYA Y. *Entendendo Algoritmos. Um guia ilustrado para programadores e outros curiosos*. São Paulo: Ed. Novatec, 2017
- DOWNEY, ALLEN B. *Pense em Python. Pense como um cientista da computação*. São Paulo: Ed. Novatec, 2016
- GRANATYR, Jones; PACHOLOK, Edson. *IA Expert Academy*. Disponível em: <https://iaexpert.academy/>
- KOPEC, DAVID. *Problemas clássicos de ciência da computação com Python*. São Paulo: Ed. Novatec, 2019
- MCKINNEY, WILLIAM WESLEY. *Python para análise de dados. Tratamento de dados com Pandas, Numpy e Ipython*. São Paulo: Ed. Novatec, 2018
- MITCHELL, RYAN. *Web Scraping com Python. Coletando mais dados na web moderna*. São Paulo: Ed. Novatec, 2019

# OBRIGADO



## FIAP

Copyright © 2023 | Professor Dr. Emerson R. Abraham

Todos os direitos reservados. A reprodução ou divulgação total ou parcial deste documento é expressamente proibida sem o consentimento formal, por escrito, do professor/autor.

