# Tratando erros ao acessar os dados na web

Introdução

# Tratando erros ao acessar os dados na web

Ao retornarmos o conteúdo de uma *url* com *urlopen* podemos nos deparar com situações que gerem erros, seja o site fora do ar ou até mesmo por um erro no servidor.

Além disso, ao tentar acessar as informações com o *BeautifulSoup* também podemos nos deparar com erros, por exemplo, dados mal formatados.

Temos que tratar estes erros para que nosso sistema não aborte um *scraping* inesperadamente.

# Tratando erros ao acessar os dados na web

Ao usar o *urlopen*, caso uma página não seja encontrada ou ocorra algum erro na página, o servidor web retorna uma mensagem de erro *HTTP*. Veja abaixo dois exemplos:

404 – Página não encontrada.

500 – Erro interno no servidor.

Nós temos que adequar nosso sistema para tratar os erros *HTTP* retornados.

# Tratando erros ao acessar os dados na web

Neste exemplo, tentamos acessar uma página inexistente no site da *Udemy*.

```
from urllib.request import urlopen
html = urlopen("http://www.wingsec.com")
print(f"Html 1: {html}")

html = urlopen("http://www.wingsec.com/erro")
print(f"Html 2: {html}")

html = urlopen("http://www.wingsec.com")
print(f"Html 3: {html}")
```

Resultado:
"...\python.exe" .../tratando_erros1.py
**Html 1: <http.client.HTTPResponse object at 0x000001D6AB3F3908>**
Traceback (most recent call last):
  File ".../tratando_erros1.py", line 5, in <module>
    html = urlopen("http://www.wingsec.com/erro")
...
**urllib.error.HTTPError: HTTP Error 404: NOT FOUND**

*O sistema parou a execução na linha 5.*

# Tratando erros ao acessar os dados na web

## Usando *HTTPError* para tratar os erros *HTTP*.

```
from urllib.request import urlopen
from urllib.error import HTTPError

html = urlopen("http://www.udemy.com")
print(f"Html 1: {html}")

try:
    html = urlopen("http://www.udemy.com/erro")
    print(f"Html 2: {html}")
except HTTPError as erro:
    print(erro)

html = urlopen("http://www.udemy.com")
print(f"Html 3: {html}")
```

Tratamos o erro *HTTP*, imprimimos o mesmo e o sistema prosseguiu sem ser interrompido.

Resultado:

```
"C:\Program Files\Python36\python.exe" .../tratando_erros1.py
Html 1: <http.client.HTTPResponse object at 0x000001DFF7F339E8>
HTTP Error 404: NOT FOUND
Html 3: <http.client.HTTPResponse object at 0x000001DFF7F334A8>

Process finished with exit code 0
```

# Tratando erros ao acessar os dados na web

E se o servidor não for encontrado ou der erro na URL? Então não temos um *HTTPError* para tratar.

Neste caso podemos tratar o *URLError*.

Para isso temos que importar a Classe *URLError* da biblioteca *urllib*.

# Tratando erros ao acessar os dados na web

## *URLError*:

```
from urllib.request import urlopen
from urllib.error import HTTPError, URLError

try:
    html = urlopen("http://www.udemy.com/erro")
except HTTPError as erro:
    print(f"Erro HTTP: {erro}")

try:
    html = urlopen("http://www.xptoxyzabracadabra.com/")
except URLError as erro:
    print(f"Erro URL: {erro}")
```

Resultado:
"C:\Program Files\Python36\python.exe"
.../tratando_erros2.py
Erro HTTP: HTTP Error 404: NOT FOUND
Erro URL: <urlopen error [Errno 11001] getaddrinfo failed>

Process finished with exit code 0

# Tratando erros ao acessar os dados na web

Agora que você sabe tratar erros relacionados ao servidor e à página, resta uma pergunta.

E se a página for recuperada com sucesso mas o conteúdo não for o que esperamos?

Ao acessar uma *tag* em um objeto *BeautifulSoup*, temos que verificar se a mesma existe, caso contrário, o *BeautifulSoup* retorna *None* (*null* em outras linguagens de programação).

Ao tentar acessar uma *tag* em um objeto *None* ocorrerá o *AttributeError* (um erro de atributo).

# Tratando erros ao acessar os dados na web

## Veja o exemplo:

```
from urllib.request import urlopen
from bs4 import BeautifulSoup

html = urlopen("http://www.udemy.com")
bsObj = BeautifulSoup(html.read(), "html.parser")
print(bsObj.html.tag_nao_existente)
```

Resultado:
...\python.exe ".../tratando_erros3.py"
**None**

Process finished with exit code 0

Ao acessar a tag "tag_nao_existente" obtemos ***None***.

```
from urllib.request import urlopen
from bs4 import BeautifulSoup

html = urlopen("http://www.udemy.com")
bsObj = BeautifulSoup(html.read(), "html.parser")
print(bsObj.html.tag_nao_existente.outra_tag)
```

Resultado:
...\python.exe ".../tratando_erros3.py"
Traceback (most recent call last):
File ".../tratando_erros3.py", line 6, in <module>
print(bsObj.html.tag_nao_existente.outra_tag)
**AttributeError: 'NoneType' object has no attribute 'outra_tag'**

Ao tentar acessar uma tag dentro da tag "tag_nao_existente" é gerado um ***AttributeError***.

# Tratando erros ao acessar os dados na web

## Tratando o *AttributeError*:

```
from urllib.request import urlopen
from bs4 import BeautifulSoup

html = urlopen("http://www.udemy.com")
bsObj = BeautifulSoup(html.read(), "html.parser")

try:
    resultado = bsObj.html.tag_nao_existente.outra_tag
except AttributeError as erro:
    print("A tag não foi encontrada")
```

```
...\python.exe ".../tratando_erros5.py"
A tag não foi encontrada

Process finished with exit code 0
```

# Tratando erros ao acessar os dados na web

```
from urllib.request import urlopen
from bs4 import BeautifulSoup

html = urlopen("http://www.udemy.com")
bsObj = BeautifulSoup(html.read(), "html.parser")

if bsObj.html.tag_nao_existente is not None:
    print(bsObj.html.tag_nao_existente.outra_tag)
else:
    print("bsObj.html.tag_nao_existente é None")

if bsObj.html is not None:
    resultado = bsObj.html.body
    print("bsObj.html.body ok. bsObj.html não é None.")
else:
    print("bsObj.html é None")

if bsObj.html is not None:
    resultado = bsObj.html.bodyTeste
    print(f"Resultado: {resultado}") #html não é None, mas bodyTeste é None.
else:
    print("bsObj.html é None")
```

Resultado:
"...\python.exe" .../tratando_erros6.py
bsObj.html.tag_nao_existente é None
bsObj.html.body ok. bsObj.html não é None.
**Resultado: None**

Process finished with exit code 0

# Tratando erros ao acessar os dados na web

```python
from urllib.request import urlopen
from urllib.error import HTTPError, URLError
from bs4 import BeautifulSoup

def getTitulo(url):
    try:
        html = urlopen(url)
    except HTTPError as erro:
        print(f"Ocorreu um erro HTTP: {erro}")
        return None
    except URLError as erro:
        print(f"Ocorreu um erro de URL: {erro}")
        return None
    except:
        print(f"Ocorreu um erro ao acessar a página.")
        return None

    try:
        bsObj = BeautifulSoup(html.read(), "html.parser")
        titulo = bsObj.body.h1
    except AttributeError as erro:
        print(f"Ocorreu um erro ao acessar o atributo h1: {erro}")
        return None
    except:
        print(f"Ocorreu um erro no conteúdo da página.")
        return None

    return titulo

titulo = getTitulo(input("Informe a URL: "))

if titulo is not None:
    print(titulo)
else:
    print("Título não encontrado.")
```

Resultado: Foi impresso o conteúdo do primeiro “<h1>” encontrado no site “https://www.linuxmint.com”.

“...\python.exe" .../exemplo_completo.py
Informe a URL: https://www.linuxmint.com
<h1>How to upgrade to Linux Mint 18.2</h1>

Resultado: Foi gerada uma exceção que não era um HTTPError ou URLError:

“...\python.exe" .../exemplo_completo.py
Informe a URL: http://xyznaoexisteestesite.com
Ocorreu um erro de URL: <urlopen error [Errno 11001] getaddrinfo failed>
Título não encontrado.

# Tratando erros ao acessar os dados na web

Como um sistema de web *scraping* normalmente é executado durante horas ou até mesmo dias, é muito importante que sejam tratados todos os possíveis erros para que o sistema não aborte a execução e você perca tempo entre o ajuste do erro e uma nova execução.

# FIM