FLASK FRAMEWORK

$POST e Tratamento de Dados com Python

Introdução

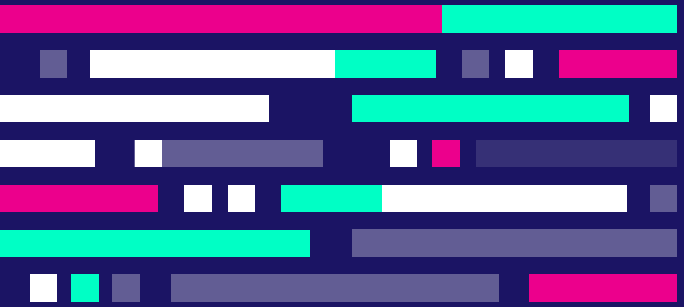
Utilizando o framework flask podemos “postar” variaveis, recebe-las e trata-las normalmente com python.

O Flask renderiza um ou mais arquivos html, retornando para o(s) mesmo(s) as variaveis.

Não utilizaremos Django por sua maior complexidade e necessidade de entender melhor todo o seu “ecossistema”, com Flask adicionamos o que realmente precisamos.

Flask

web development,
one drop at a time



Observações:

+ Pasta `templates` para os HTMLs no flask

+ pip install flask # Instalção de bibliotecas

+ pip install requests # Instalção de bibliotecas

+ $env:FLASK_APP = "main“ # Informa ao flask que o arquivo principal se chama main, no lugar de main, coloque o nome do arquivo .py.

+ flask run # Para rodar o servidor local, o qual irá prover um link para acessar a página. Lembre-se, a cada alteração no código desligue o server com Ctrl + c no terminal e rode-o novamente.

Iniciando

```
from flask import Flask, render_template, request
app = Flask(__name__)
@app.route('/', methods=['GET', 'POST’])
def index():
        if request.method == 'POST’:
                num1 = int(request.form[“numero1”])
                num2 = int(request.form.get(“numero2”))
                num3 = int(request.form[“numero3”])
                delta = (num2 * num2) + (-4 * (num1 * num3))
                return render_template('resultado.html', deltahtml=delta)
        return render_template('index.html’)

if __name__ == 'main’:
        app.run()
```

```
from flask import Flask, render_template, request
```

Importamos as bibliotecas Flask, render_template e request do módulo flask;

O Flask é a classe principal, usada para criar um objeto (instância) do aplicativo;
O render_template é usado para renderizar templates HTML;
O request é usado para acessar os dados enviados por meio de requisições HTTP;

```
app = Flask(__name__)
```

Cria uma instância do Flask, passamos __name__ com argumento e atribuímos a app, para o Flask localizar os nossos módulos/arquivos.

@app.route('/', methods=['GET', 'POST’])

@app.route associa a URL a função e nesse caso a nossa rota padrão é ‘/’;

Quando o cliente/usuário fizer uma requisição para a rota padrão (‘/’) a nossa função será executada;

methods=['GET', 'POST’], define que a rota aceitará ser acessada por POST e GET, sendo POST mais seguro e lento que GET;

GET é usado para solicitor recursos do servidor;
POST é usado para enviar os dados.

```
def index():
        if request.method == 'POST':
                num1 = int(request.form[“numero1”])
                num2 = int(request.form.get(“numero2”))  # Se vazio = none
                num3 = int(request.form[“numero3”])
                delta = (num2 * num2) + (-4 * (num1 * num3))
                return render_template('resultado.html', deltahtml=delta)
        return render_template('index.html’)
```

Definimos a função index() que será executada quando a rota for acessada;

if request.method == ‘POST’ confere se o método de requisição for POST (será definido no form html);

int(request.form[“numero1”]) Faz uma requisição para o formulário html procurando pelo name = ‘numero1’, o converte para inteiro e atribui a num1;
int(request.form.get(“numero2”)) faz o mesmo que o anterior, mas evita erros no servidor caso o usuário não preencha os dados, pois .get tenta requisitar os dados.

```
def index():
        if request.method == 'POST':
                num1 = int(request.form["numero1"])
                num2 = int(request.form.get("numero2"))
                num3 = int(request.form["numero3"])
                delta = (num2 * num2) + (-4 * (num1 * num3))
                return render_template('resultado.html', deltahtml=delta
        return render_template('index.html')
```

return render_template('resultado.html', deltahtml=delta) Renderiza a página 'resultado.html' retornando para o html o valor deltahtml que contém o valor da variável delta;

return render_template('index.html') Perceba que está fora do if, então, o método será GET, pois nenhum dado está sendo enviado, com isso, será renderizada a página 'index.html'. Resumindo quando o servidor local for executado e o link for acessado, será retornada para a requisição, através de GET a página principal.

```
if __name__ == '__main__':
    app.run()
```

Verifica se o módulo está sendo sendo executado diretamente, no programa principal, e não importado em outro programa, fazendo com que o mesmo seja executado automaticamente.

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <title>Entrada Usuário - Home</title>
    <style>
    </style>
</head>

<body>

    <h1>Calculadora de Equações do 2º Grau</h1>

    <div>
        <form action="/" method="POST">  # Perceba a Action. Enviando para a rota raiz atravez de POST!

            <h3>Digite os Coeficiente</h3>
            <input required type="number" name="numero1" placeholder="Digite Coeficiente A">
            <input required type="number" name="numero2" placeholder="Digite Coeficiente B">
            <input required type="number" name="numero3" placeholder="Digite Coeficiente C">
            <input type="submit" value="Calcular">
        </form>
    </div>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <title>Resultado</title>
    <style>
    </style>
</head>

<body>
    <div>
        <h1>Resultados:</h1>
        <h3> Delta: {{deltahtml}}</h3> # Para inserir as variáveis retornadas:
                                                                {{variavel}}
    </div>
</body>
</html>
```

PyScrap

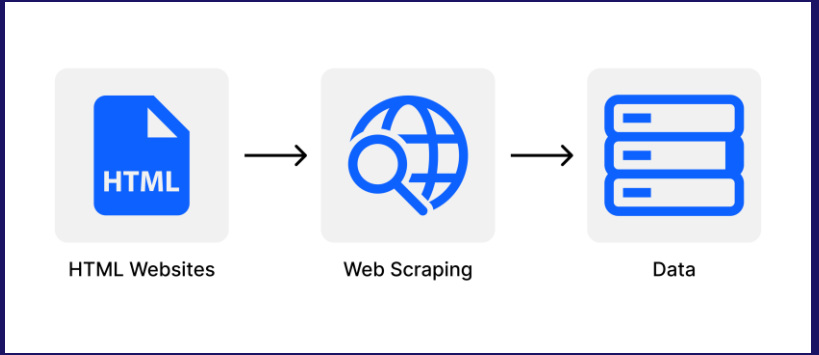
Raspagem de Dados da Web com Python
Web Scraping com BeatifulSoup

Introdução

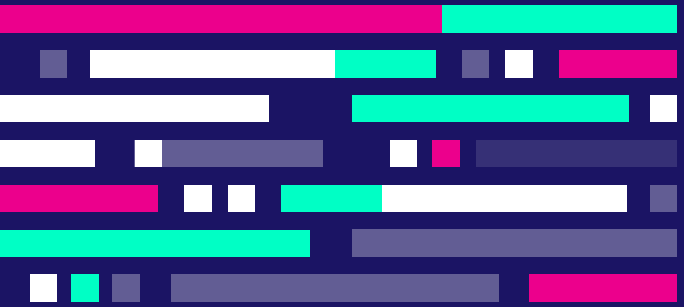
Utilizamos PyScrap para “raspar” dados da web com python e trata-los normalmente.

Utilizam-se seletores CSS para identificar os conteúdos necessarios, como class(.) e id(#).

Usaremos a biblioteca BeautifulSoup para isso.



Observações:



+ pip install beautifulsoup4 # Instalção de #bibliotecas

+ from bs4 import BeautifulSoup # Importar libs

Também utilizaremos a lib requests:

+ pip install requests # Instalar lib

+ import requests # Importar lib

```
def scrpreco(url, seletor):
    headers = {
        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/71.0.3578.98 Safari/537.36',
        'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8',
        'Accept-Language': 'en-US,en;q=0.5',
        'DNT': '1',
        'Connection': 'close'
    }
    response = requests.get(url, headers=headers)
    soup = BeautifulSoup(response.text, 'html.parser')

    preco = soup.select_one(seletor).text.strip()

    precofloat0 = ""

    for x in preco:
            if x == ",":
                precofloat0 += "."
            elif x.isdigit():
                precofloat0 += x

    return float(precofloat0)

url0 = 'https://www.neosolar.com.br/loja/painel-solar-fotovoltaico-155w-resun-rs6e-155p.html'
seletor0 = '#product-price-31084’

url1 = 'https://www.neosolar.com.br/loja/painel-solar-fotovoltaico-330w-osda-oda330-36-p.html'
seletor1 = '#product-price-30168'

precoscrap0 = scrpreco(url0, seletor0)
precoscrap1 = scrpreco(url1, seletor1)
```

```
def scrpreco(url):
```

Definimos uma função scrpreco que recebe o parâmetro url;

```
 headers = {
        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/71.0.3578.98 Safari/537.36',
        'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8',
        'Accept-Language': 'en-US,en;q=0.5',
        'DNT': '1',
        'Connection': 'close'
    }
```

Definimos um dicionário contendo alguns recursos HTTP para tentar burlar as verificações de bots, que alguns sites podem possuir. Isso fará com que o código simule um navegador.

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

Envia-se uma solicitação GET para a URL que foi recebida como parâmetro, passa os headers definidos e armazena a resposta da solicitação em response.

soup = BeautifulSoup(response.text, 'html.parser’)

Criamos um objeto BeatifulSoup, que utiliza os dados guardados em response e com o .text recolhe os dados da página como uma string, ou seja, como um texto.

'html.parser’ especifica que o analisador html a ser utilizado será o do BeautifulSoup, mas existem outros que podem ser utilizados.

```
preco = soup.select_one(seletor).text.strip()
```

Utilizamos o html obtido em string na variável soup, usamos a função select_one() com um selector CSS (que foi enviado para a função), no caso acima um id. Usamos .text para obter o texto e .strip para remover os espaçoes em branco.

OBS.: select() retorna a lista de todos elementos correspondents, enquanto select_one() retorna apenas o primeiro elemento.

precofloat0 = “”

Criamos uma variável vázia do tipo string

```
 for x in preco:
            if x == ",":
                        precofloat0 += "."
            elif x.isdigit():
                        precofloat0 += x
```

for itera sobre a variável preco e, se for o caso, concatena com a variável precofloat0

```
return float(precofloat0)
```

Retorna a variável com os dados tratados, a converte para float e atribui o resultado para a variável precoscrapX

```
url0 = 'https://www.neosolar.com.br/loja/painel-solar-fotovoltaico-155w-resun-rs6e-155p.html'
seletor0 = '#product-price-31084'

url1 = 'https://www.neosolar.com.br/loja/painel-solar-fotovoltaico-330w-osda-oda330-36-p.html'
seletor1 = '#product-price-30168’
```

Definimos váriaveis, do tipo string, para enviarmos a função.

```
precoscrap0 = scrpreco(url0, seletor0)
precoscrap1 = scrpreco(url1, seletor1)
```

Em python executamos uma função com `nome()`, logo, em scrpreco(url0, seletor0) estamos executando uma função e enviando alguns parâmetros, por fim, atribuímos o resultado retornado da função para a variável precoscrap0.
precoscrapX seria o valor a ser retornado/renderizado para o html de resultado, pois é a variável global.

EXERCICIO

Teste o código e tente entender melhor como funciona;

Modifique-o e/ou reescreva em um projeto próprio.

LEMBRE-SE! Não se preocupe em decorar código, e sim em entende-lo e saber o que você pode fazer.