

Taller de Python

Científico



Descargando sin API



Departamento de

Ciencias de la Atmósfera y los Océanos

Facultad de Ciencias Exactas y Naturales - Universidad de Buenos Aires

Descargando Datos o Imágenes

Necesito descargar la temperatura máx y mín del día cada 1 hora de Río Cuarto, pero el sitio no posee una API.

Legales Suscripción Institucionales Acerca de... Contacto OMM

Servicio Meteorológico Nacional
Creado el 4 de octubre de 1872
143 Años
Al servicio del País

SECRETARÍA DE CIENCIA, TECNOLOGÍA Y PRODUCCIÓN
Ministerio de Defensa
Presidencia de la Nación

Inicio Intramet SMN Links Preguntas frecuentes RSS Portal Móvil

>> Configurar el mapa como la página de inicio de este sitio.


Buscar en el sitio

Pronóstico para la Provincia de Córdoba

Para ver el Pronóstico, haga click sobre el Nombre de la Ciudad

Última información disponible

Alerta en el país



Río Cuarto

Día: 30-07-2016
Hora: 06:00 horas

Día	Mañana	Tar/Noc
Sáb		
	Min: 5°C Máx: 17°C	
Dom		
	Min: 10°C Máx: 17°C	
Lun		
	Min: 8°C Máx: 16°C	
Mar		
	Min: 7°C Máx: 16°C	

Temperatura actual en la ciudad de Buenos Aires
13.2°C
Fresco

Perspectiva Semanal
INFORME PERSPECTIVA
SEMANAL - Nuevo -

Servicios Climáticos
SERVICIOS CLIMÁTICOS
Pronóstico climático trimestral para Argentina

Observaciones
INDICES UV
« Estado del Tiempo
« Centro de Meteorología por Sensores Remotos
« Ozono

Cenizas Volcánicas
VAAC Buenos Aires

Productos Elaborados
« Pronósticos y alertas
« Servicio Meteorológico

Dos métodos básicos de HTTP

Los dos métodos básicos con los que podemos interactuar con un servidor son:

GET - Pide los datos al servidor.
Cuando cargamos una página web.

POST - Sube datos para ser procesados por el servidor.
Cuando se completa un formulario.

Por suerte, Python tiene una biblioteca que implementa estos métodos:

[Requests: HTTP for Humans](#)

Descargando la página web

```
>>> import requests  
  
>>> url = 'http://www.smn.gov.ar/?mod=pronografico&' +  
          'id=2&var=cordoba&imagen=ZN'  
  
>>> web = requests.get(url)  
>>> print(web.text)
```

```
<tr><td>  
  <!--p>  
    <div style="text-align:left">  
      &nbsp;<a href="htms/INFORME_TORMENTA_04_04_2012.pdf" target="_blank">IN  
FORME FINAL - TORMENTA SEVERA 4 DE ABRIL DE 2012</a><br />  
    </div>  
  </p!--><br />  
  <strong><font color="#FF0000">Para ver el Pron&ostico, haga click sobre el Nombre de la Ciudad </fon  
t></strong>  
  <br /><br /> <b>Última información disponible</b> </td>  
</tr>  
  
<tr>  
<td>  
<TABLE width='100%' CELLPADDING=0 CELLSPACING=0 BORDER=0 align='right'><tr></tr></TABLE> </td>  
</tr>
```

¿Cómo proceso eso?

Procesando texto (html) con Python

Scraping approach	Performance	Ease of use	Ease to install
Regular expressions	Fast	Hard	Easy (built-in module)
Beautiful Soup	Slow	Easy	Easy (pure Python)
Lxml	Fast	Easy	Moderately difficult

Entendiendo un poco de HTML

HTML es un lenguaje de etiquetado. Algunas etiquetas son:

`<h1>` a `<h6>`: títulos del documento con diferente relevancia.

`<table>`: define una tabla.

`<tr>`: fila de una tabla.

`<td>`: celda de una tabla.

`<a>`: hipervínculo o enlace, dentro o fuera del sitio web.

`<div>`: división de la página.

``: imagen.

``````: etiquetas para listas.

Atributos:

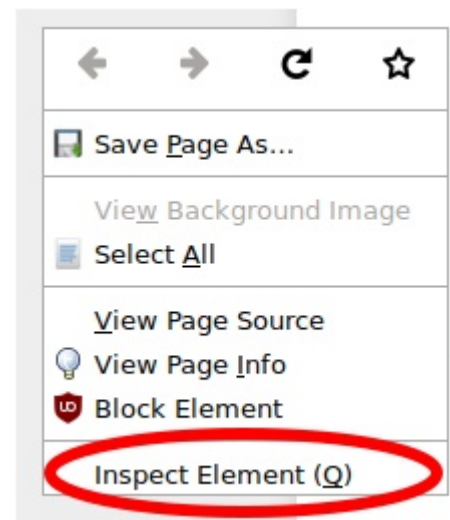
`class=" "` : Para darle un estilo particular

`id=" "` : identificador único

Beautiful Soup

Lo potente de [BeautifulSoup](#) es que nos permite buscar macheando etiquetas y/o atributos. Entonces, tenemos que buscar entre qué etiquetas se encuentra los datos que nos interesan.

En el navegador web, botón derecho, inspeccionar.



Buscando la etiqueta

The screenshot displays a weather application. On the left, a map of Argentina shows various cities with their respective weather icons and temperatures. On the right, a table provides a 3-day forecast. The browser's developer tools are open at the bottom, showing the DOM tree and the selected element's HTML code.

Día	Mañana	Tar/Noc
Sáb		
	Min: 5°C Máx: 17°C	
Dom		
	Min: 10°C Máx: 17°C	
Lun		
	Min: 8°C Máx: 16°C	
Mar		
	Min: 7°C Máx: 16°C	

```
<tr bgcolor="#E4E4E4"></tr>
<tr></tr>
<tr>
  <td colspan="2" align="center" bgcolor="#EAF4FF">
    <div align="center">
      <font size="1">Mín:</font>
      <font color="#0000FF" size="1">5°C</font>
      <font size="1">Máx:</font>
      <font color="#FF0000" size="1">17°C</font>
    </div>
  </td>
  <!--aca comienza el segundo dia-->
</tr>
<tr></tr>
```


Parseando

```
>>> soup = BeautifulSoup(html.text, 'html.parser')
>>> table = soup.find('td', attrs={'colspan':'2', 'align': 'center',
                                   'bgcolor':'#EAF4FF'})
>>> minima = table.find('font', attrs={'color':'#0000FF'})
>>> maxima = table.find('font', attrs={'color':'#FF0000'})
>>> print(minima, maximia)
5°C, 17°C
```

Procesos periódicos automatizados



Tecadmin.net

Crontab ejemplos

```
$ crontab -l #listar
```

```
$ crontab -e #editar
```

```
# cada 10 minutos corre un script
```

```
*/10 * * * * /path/to/script.py
```

```
# guarda la memoria que usa el sistema todos los lunes
```

```
# a las 3:30AM en el archivo /tmp/meminfo
```

```
30 3 * * mon cat /proc/meminfo >> /tmp/meminfo
```

```
# Corre un script el primer día del mes a las 4:10Am
```

```
10 4 1 * * /root/scripts/backup.sh
```

Más ejemplos