Beauti t

Tip: Use search to find it faster

Find it faster.

This is an interactive **Beatuiful Soup** cheat sheet. Beautiful Soup is a Python library for parsing web pages and HTML. Use the search to instantly get answers.

Basic

- Hello World
- Installing
- Parsing
- Online Docs

Page Properties

- Page
- Page Properties
- Page Functions
- Element
- Element Properties
- Element Functions

Related SpeedSheets

- Python
- aiohttp
- requests

Online Documents

Beautiful Soup Documents

https://www.crummy.com/software/BeautifulSoup/

https://beautiful-soup-4.readthedocs.io/en/latest/

Beautiful Soup Library on PyPi

https://pypi.org/project/beautifulsoup4/

Library

Install Library - pip

```
pip install beautifulsoup4
```

Install Library - pip requirements.txt

requirements.txt:

```
beautifulsoup4
```

Command:

```
pip install -r requiremets.txt
```

Program - Import

```
from bs4 import BeautifulSoup
```

Hello World

Read Google

Gets the page title for Google's landing page.
Use Python's internal urllib library to retrieve the page.

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

url = 'https://google.com'

with urlopen(url) as response:
    content = response.read()

soup = BeautifulSoup(content, 'html.parser')

print(soup.title)
```

Read Google / Using Requests

Get the page title for Google's landing page using the requests http library:

```
from bs4 import BeautifulSoup
import requests

url = 'https://google.com'

content = requests.get(url).text
soup = BeautifulSoup(content, 'html.parser')

print(soup.title)
```

Parse

Parse HTML - Using Python's Stock Parser

```
= BeautifulSoup(content, 'html.parser')
```

Usage:

```
from bs4 import BeautifulSoup

soup = BeautifulSoup(content, 'html.parser')
```

Parse HTML - Using lxml

```
= BeautifulSoup(content, 'lxml')
```

Usage:

```
from bs4 import BeautifulSoup

soup = BeautifulSoup(content, 'lxml')
```

Requires lxml:

pip / requirements.txt

```
lxml
```

Parse XML - Using lxml

```
= BeautifulSoup(content, 'xml')
```

Usage:

```
from bs4 import BeautifulSoup
soup = BeautifulSoup(content, 'xml')
```

Requires lxml:

pip / requirements.txt

```
lxml
```

Page

Page - Properties

Page - Body

```
= soup.body
```

Returns the body element of the page.

Page - Head

```
= soup.head
```

Returns the head element of the page.

Page - HTML

```
= str(soup)
```

Returns the page HTML.

Page - Links

```
= [ element['href']
        for element in soup.find_all('a') if
'href' in element.attrs]
```

Returns all page links.

Page - Title

```
= soup.title.string
```

Page - Text - Entire Text

```
= soup.get_text()
```

Page - Functions

Page - Find By Attribute

```
= soup.find(attrs = {'attribute':'value'}
```

```
= soup.find(attrs = {
        'attribute_1':'value',
        'attribute_2':'value',
        ...
}
```

Page - Find By Class Name

```
= soup.find(class_ = 'class_name')
```

Page - Find By Element Type

```
= soup.find('element_type')
```

Page - Find By ID

```
= soup.find(id = 'id')
```

Page - Find By Selector

```
= soup.select_one('selector')
```

Page - Find All

```
= soup.find_all()
```

Returns all sub-elements.

Page - Find All By Element Type

```
= soup.find_all('element_type')
```

Page - Find All By Class Name

```
= soup.find_all(class_ = 'class_name')
```

Page - Find All By Selector

```
= soup.select('selector')
```

Element

Element - Properties

Element - Get Name

```
= element.name
```

Returns the name (element type) of the element.

For a div element, returns 'div'.

Element - Get Text

```
= element.string
```

Returns the inner text of the element.

Element - Get HTML

```
= str(element)
```

Element - Get Attribute

```
= element['attribute']
```

or

```
= element.get('attribute')
```

Element - Get Attributes

```
= element.attrs
```

Returns: `dict`

Returns a dict of all attribute names.

Element - Get Classes

```
= element['class']
```

Returns: `list[str]`

Returns an array of all classes for the element.

If only one class, still returns an array.
If no class, the class attribute is not present.

Get Classes If Any:

```
def get_classe(element):
    ''' Return classes or empty list if
none.'''

    if 'class' not in element.attrs:
        return []

    return element['class']

classes = get_classes(element_1)
```

Element - Get ID

```
= element['id']
```

Returns: `str`

Returns the element id.

Only Tags (type `bs4.element.Tag`) have ids.
Tags with no ids will return `None`.
Non tags will have no id attribute.

Element - Has Attribute?

```
if 'attribute' in element.attrs:
```

Element - Has Class Attribute?

```
= type(item) == Tag and 'class' in element.attr
```

Usage:

```
from bs4.element import Tag

= type(item) == Tag and 'class' in element.attr
```

Returns true when the item has a classes attribute.

The `type(...) = Tag` prevent errors when checking on an item

that is not an element.

Element - Has Class Name?

```
= type(item) == Tag and 'class' in element.attr
and class_name in element['class']
```

Usage:

```
from bs4.element import Tag

= type(item) == Tag and 'class'  inelement.attr
```

and *class_name* in *element*['class']<<>

Returns true when the item has a class of the given name.

The `type(...) = Tag` and `'class' in element.attr` prevent errors when checking on an item that is not an element or has no class attribute.

Element - Has Id?

```
= type(item) == Tag and element.id is not None
```

Usage:

```
from bs4.element import Tag

= type(item) == Tag and 'class' element.id is
not None
```

Returns true when the item has an id attribute.

Element - Type

```
bs4.element.Tag
```

```
from bs4.element import Tag

= type(element) == Tag
```

Elements are returned as instances of the `Tag` class.

Element - Properties - Relative Elements

Element - Get Children

```
= element.children
```

Returns: `iterator`

Returns all element's direct children.

Includes elements (tags) and non element children.

Element - Get Child Elements Only

```
= [child for child in element.children if
type(child) == Tag]
```

Usage:

```
from bs4.element import Tag

= [child for child in element.children if
type(child) == Tag]
```

Returns: `list[Tag]`

Returns all direct child elements only.

Non elements are filtered out.

Element - Get Parent

```
= element.parent
```

Returns the parent element of the element.

Element - Get Sibling - Previous

```
= element.previous_sibling
```

Returns the next sibling element.

Element - Get Sibling - Next

```
= element.next_sibling
```

Returns the next sibling element.

Element - Functions

Element - Find By Element Type

```
= element.find('element_type')
```

Element - Find By Class Name

```
= element.find(class_ = 'class_name')
```

Element - Find By ID

```
= element.find(id = 'id')
```

Element - Find All

```
= element.find_all()
```

Returns all sub-elements.

Element - Find All By Element Type

```
= element.find_all(element_type)
```

Element - Find All By Class Name

```
= element.find_all(class_ = class_name)
```

How To

Is Element / Tag

```
= type(item) == Tag
```

Usage:

```
from bs4.element import Tag
= type(item) == Tag
```

Format HTML

```
= soup.prettify()
```

```
= element_1.prettify()
```

Returns formatted HTML with each element on it's own line.

Example:

```
from bs4 import BeautifulSoup

html = '''<html><body><table><tr><td>Item
1</td></tr></table></body></html>'''

soup = BeautifulSoup(html, 'html.parser')
formatted = soup.prettify()

print(formatted)

# Prints:
#
#   <html>
#    <body>
#     <table>
#      <tr>
#       <td>
#        Item 1
#       </td>
#      </tr>
#     </table>
#    </body>
#   </html>
```

Table - Parse

```
data = []

table = soup.find('table', ...)

for row in table.find_all('tr'):
    values = [column.text.strip() for column in
row.find_all('td')]
    data.append(values)
```

With 'tbody':

```
data = []

table = soup.find('table', ...)
body = table.find('tbody')

for row in body.find_all('tr'):
    values = [column.text.strip() for column in
row.find_all('td')]
    data.append(values)
```

Extracts the column values in the table into a list of lists.

Example:

```
from bs4 import BeautifulSoup

html = '''<html><body>
    <table id="table-1">
        <tr><td>1</td><td>Item 1</td></tr>
        <tr><td>2</td><td>Item 2</td></tr>
        <tr><td>3</td><td>Item 3</td></tr>
    </table>
</body></html>'''

soup = BeautifulSoup(html, 'html.parser')

data = []
table = soup.find('table', id='table-1')
for row in table.find_all('tr'):
    values = [column.text.strip() for column in
row.find_all('td')]
    data.append(values)

print(data)

# Prints: [['1', 'Item 1'], ['2', 'Item 2'],
['3', 'Item 3']]
```

Terms

lxml

A fast library for parsing XML and HTML.

Wraps the C librarie libxml2 and libxslt.

Home Page:
https://lxml.de/

PyPi:
https://pypi.org/project/lxml/

XPath

Not supported by BeautifulSoap