

Basic

```
#
https://www.crummy.com/software/BeautifulSoup/bs4/doc/
from bs4 import BeautifulSoup
soup = BeautifulSoup(html_doc, 'html.parser')
soup.title # <title>The Dormouse's story</title>
soup.title.name # u'title'
soup.title.string # u'The Dormouse's story'
soup.title.parent.name # u'head'
# various finder
css_soup.select("p.strip.outbody") # css
finder
soup.p # <p class="title"><b>The Dormouse's
story</b></p>
soup.p['class'] # u'title'
soup.a # <a class="sister" href="http://example.com/else" id="link1">Else</a>
soup.find_all('a') # [<a .>, ..]
soup.find(id="link3") # <a class="sister"
href="http://example.com/tillie" id="link3">Tillie</a>
for link in soup.find_all('a'):
    print(link.get('href'))
# example.com/else, # http://example.com/lacie
```

Search (cont)

```
> soup.select('a[href=".com/el"]') # middle match
#-----
# basic
#-----
soup.find_all('b') # match by tag
soup.find_all(re.compile("^b")) # match by tag using regex
soup.find_all(['a', 'b']) # match by tag in list
# function (complex condition)
def has_class_but_no_id(tag):
    return tag.has_attr('class') and not tag.has_attr('id')
soup.find_all(has_class_but_no_id)
#-----
# find_all_api
#-----
find_all(name, attrs, recursive, string, limit, **kwargs)
soup.find_all("title") # tag condition
soup.find_all("p", "title") # tag and attr
# [<p class="title"><b>The Dormouse's story</b></p>]
soup.find_all("a")
# keyword arguments
soup.find_all(id="link2")
soup.find_all(href=re.compile("else"), id="link1")
soup.find(string=re.compile("sisters")) # text contain sisters
# css class (class is reserved keyword)
soup.find_all("a", class_="sister")
```

Search

```
search.pyhttps://www.crummy.com/software/BeautifulSoup/bs4/doc/
#-----
# css selector
#-----
css_soup.select("p.strip.outbody")
soup.select("p:nth-of-type(3)") # 3rd child
soup.select("head>title")
soup.select("p>a:nth-of-type(2)")
soup.select("p>#link1") # direct child
soup.select("#link1~.sister") # sibling
soup.select('a[href]') # existence of an attribute
soup.select_one(".sister")
# attribute value
soup.select('a[href="http://example.com/else"]')
# exact attribute
soup.select('a[href="http://example.com/"]') #
negative match
soup.select('a[href$="tille"]') # end match
```

Make soup

```
soup = BeautifulSoup(open("index.html"))
soup = BeautifulSoup("<html>d ata</html>")
```

Output

```
# HTML
soup.prettify() #pretty print
str(soup) # non-pretty print
# String
soup.get_text() #all text under the element
```

Encoding

```
#output
soup.prettify("l at n-1 ")
tag.encode("utf-8")
tag.encode("l at n-1 ")
tag.encode("ascii")
```

Navigation

```
#-----
# going up/down/side
#-----
# ----- going down -----
soup.head# <head> <title>The Dormouse's
story< /title> </head>
soup.title# <title>The Dormouse's story< /ti -
tle>
soup.body.b # <b>The Dormouse's story< /b>
soup.a # <a class= " sister " href="http:// -
example.com/else" id="link1"> Else< /a>
soup.find_all('a')
# [<a class= " sister " href="http:// example -
.com/else" id="link1"> Else< /a>,
# <a class= " sister " href="http:// example -
.com/lacie" id="link2"> Lacie< /a>,
# <a class= " sister " href="http:
# children = contents
head_tag.contents # [<title>The Dormouse's
story< /title>]
head_tag.children # [<title>The Dormouse's
story< /title>]
# descendants (all of a tag's children, recurs -
ively)
for child in head_tag.descendants:
    print(child)

# .string is tricky
head_tag.contents # [<title>The Dormouse's
story< /title>]
```

Navigation (cont)

```
> head_tag.string # u'The Dormouse's story' (because head tag has
only one child)
print(soup.html.string) # None (because html has many children)
# whitespace removed strings
for string in soup.stripped_strings:
    print(repr(string))

# ----- going up -----
title_tag.parent # <head><title>The Dormouse's story</title></head>
# going up recursively
link.parents # [ p, body, html, [document], None]
# ----- sideways -----
# sibling = include text node
sibling_soup.b.next_sibling
sibling_soup.c.previous_sibling
# multiple
sibling_soup.b.next_siblings
sibling_soup.c.previous_siblings
# element = not include text node
sibling_soup.b.next_element
sibling_soup.c.previous_element
sibling_soup.b.next_elements
sibling_soup.c.previous_elements
```

Navigation

```
#-----
# going up/down/side
#-----
# ----- going down -----
soup.head# <head> <title>The Dormouse's
story< /title> </head>
soup.title# <title>The Dormouse's story< /ti -
tle>
soup.body.b # <b>The Dormouse's story< /b>
soup.a # <a class= " sister " href="http:// -
example.com/else" id="link1"> Else< /a>
soup.find_all('a')
# [<a class= " sister " href="http:// example -
.com/else" id="link1"> Else< /a>,
# <a class= " sister " href="http:// example -
.com/lacie" id="link2"> Lacie< /a>,
```

Navigation (cont)

```
> # <a class="sister" href="http:
# children = contents
head_tag.contents # [<title>The Dormouse's story</title>]
head_tag.children # [<title>The Dormouse's story</title>]
# descendants (all of a tag's children, recursively)
for child in head_tag.descendants:
    print(child)

# .string is tricky
head_tag.contents # [<title>The Dormouse's story</title>]
head_tag.string # u'The Dormouse's story' (because head tag has
only one child)
print(soup.html.string) # None (because html has many children)
# whitespace removed strings
for string in soup.stripped_strings:
    print(repr(string))

# ---- going up ----
title_tag.parent # <head><title>The Dormouse's story</title></head>
# going up recursively
link.parents # [ p, body, html, [document], None]
# ---- sideways ----
# sibling = include text node
sibling_soup.b.next_sibling
sibling_soup.c.previous_sibling
# multiple
sibling_soup.b.next_siblings
sibling_soup.c.previous_siblings
# element = not include text node
sibling_soup.b.next_element
sibling_soup.c.previous_element
sibling_soup.b.next_elements
sibling_soup.c.previous_elements
```



By [wangmz](#)
cheatography.com/wangmz/

Not published yet.
Last updated 17th May, 2018.
Page 3 of 3.

Sponsored by [Readable.com](#)
Measure your website readability!
<https://readable.com>