**BeautifulSoup – парсинг HTML в Python на примерах**



Данная инструкция по BeautifulSoup является вводным руководством по использованию библиотеки BeautifulSoup Python. В примерах показано использование тегов, модификация документа и перебор его элементов, а также парсинг веб-страниц.

**Содержание статьи**

* [BeautifulSoup на примерах](https://python-scripts.com/beautifulsoup-html-parsing#BeautifulSoup-examples)
* [Установка BeautifulSoup в Python](https://python-scripts.com/beautifulsoup-html-parsing#install-BeautifulSoup)
* [Пример HTML-кода страницы](https://python-scripts.com/beautifulsoup-html-parsing#html-example)
* [BeautifulSoup простой пример парсинга HTML](https://python-scripts.com/beautifulsoup-html-parsing#html-parsing-example)
* [BeautifulSoup теги, атрибуты name и text](https://python-scripts.com/beautifulsoup-html-parsing#atribute-name-text)
* [BeautifulSoap перебираем HTML теги](https://python-scripts.com/beautifulsoup-html-parsing#traverse-tags)
* [BeautifulSoup атрибут children](https://python-scripts.com/beautifulsoup-html-parsing#atribute-children)
* [BeautifulSoup атрибут descendants](https://python-scripts.com/beautifulsoup-html-parsing#atribute-descendants)
* [BeautifulSoup и веб-скрапинг HTML](https://python-scripts.com/beautifulsoup-html-parsing#web-scraping)
* [BeautifulSoup метод prettify()](https://python-scripts.com/beautifulsoup-html-parsing#method-prettify)
* [BeautifulSoup метод find(), поиск элементов по id](https://python-scripts.com/beautifulsoup-html-parsing#method-find)
* [BeautifulSoup метод find\_all() поиск всех тегов в HTML](https://python-scripts.com/beautifulsoup-html-parsing#method-find-all)
* [BeautifulSoup методы select() и select\_one() CSS селекторы](https://python-scripts.com/beautifulsoup-html-parsing#method-select)
* [BeautifulSoup метод append() добавление нового HTML-тега](https://python-scripts.com/beautifulsoup-html-parsing#method-append)
* [BeautifulSoup метод insert() вставка HTML-тега](https://python-scripts.com/beautifulsoup-html-parsing#method-insert)
* [BeautifulSoup метод replace\_with() замена текста в теге](https://python-scripts.com/beautifulsoup-html-parsing#method-replace)
* [BeautifulSoup метод decompose() удаление HTML-тега](https://python-scripts.com/beautifulsoup-html-parsing#method-decompose)

**BeautifulSoup на примерах**

BeautifulSoup является библиотекой Python для парсинга HTML и XML документов. Часто используется для **скрапинга веб-страниц**. BeautifulSoup позволяет трансформировать сложный HTML-документ в сложное древо различных объектов Python. Это могут быть теги, навигация или комментарии.

**Установка BeautifulSoup в Python**

Для установки необходимых модулей используется команда pip3.

Shell



|  |  |
| --- | --- |
| 1 | $ sudo pip3 install lxml |

**Есть вопросы по Python?**

На нашем форуме вы можете задать любой вопрос и получить ответ от всего нашего сообщества!

Для начала требуется установить lxml модуль, который используется в BeautifulSoup.

Shell



|  |  |
| --- | --- |
| 1 | $ sudo pip3 install bs4 |

BeautifulSoup устанавливается при помощи использования указанной выше команды.

**Пример HTML-кода страницы**

В последующих примерах будет использован данный HTML-файл:

XHTML



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | *<!DOCTYPE html>*  <html>      <head>          <title>Header</title>          <meta charset="utf-8">      </head>        <body>          <h2>Operating systems</h2>            <ul id="mylist" style="width:150px">              <li>Solaris</li>              <li>FreeBSD</li>              <li>Debian</li>              <li>NetBSD</li>              <li>Windows</li>          </ul>            <p>            FreeBSD is an advanced computer operating system used to            power modern servers, desktops, and embedded platforms.          </p>            <p>            Debian is a Unix-like computer operating system that is            composed entirely of free software.          </p>        </body>  </html> |

**BeautifulSoup простой пример парсинга HTML**

В первом примере будет использован BeautifulSoup модуль для получения трех тегов.

Python



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | #!/usr/bin/python3    from bs4 import BeautifulSoup    with open("index.html", "r") as f:        contents = f.read()        soup = BeautifulSoup(contents, 'lxml')        print(soup.h2)      print(soup.head)      print(soup.li) |

Код в данном примере позволяет вывести HTML-код трех тегов.

Python



|  |  |
| --- | --- |
| 1 | from bs4 import BeautifulSoup |

Здесь производится импорт класса BeautifulSoup из модуля bs4. Таким образом, BeautifulSoup является главным рабочим классом.

Python



|  |  |
| --- | --- |
| 1  2  3 | with open("index.html", "r") as f:        contents = f.read() |

Открывается файл index.html и производится чтение его содержимого при помощи метода read().

Python



|  |  |
| --- | --- |
| 1 | soup = BeautifulSoup(contents, 'lxml') |

Создается объект BeautifulSoup. Данные передаются конструктору. Вторая опция уточняет объект парсинга.

Python



|  |  |
| --- | --- |
| 1  2 | print(soup.h2)  print(soup.head) |

Далее выводится HTML-код следующих двух тегов: h2 и head.

Python



|  |  |
| --- | --- |
| 1 | print(soup.li) |

В примере много раз используются элементы li, однако выводится только первый из них.

Shell



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | $ ./simple.py  <h2>Operating systems</h2>  <head>  <title>Header</title>  <meta charset="utf-8"/>  </head>  <li>Solaris</li> |

Это результат вывода.

**BeautifulSoup теги, атрибуты name и text**

Атрибут name указывает на название тега, а атрибут text указывает на его содержимое.

Python



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | #!/usr/bin/python3    from bs4 import BeautifulSoup    with open("index.html", "r") as f:        contents = f.read()        soup = BeautifulSoup(contents, 'lxml')        print("HTML: {0}, name: {1}, text: {2}".format(soup.h2,          soup.h2.name, soup.h2.text)) |

Код в примере позволяет вывести HTML-код, название и текст h2 тега.

Shell



|  |  |
| --- | --- |
| 1  2 | $ ./tags\_names.py  HTML: <h2>Operating systems</h2>, name: h2, text: Operating systems |

Это результат вывода.

**BeautifulSoap перебираем HTML теги**

Метод recursiveChildGenerator() позволяет перебрать содержимое HTML-документа.

Python



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | #!/usr/bin/python3    from bs4 import BeautifulSoup    with open("index.html", "r") as f:        contents = f.read()        soup = BeautifulSoup(contents, 'lxml')        for child in soup.recursiveChildGenerator():            if child.name:                print(child.name) |

Данный пример позволяет перебрать содержимое HTML-документа и вывести названия всех его тегов.

Shell



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | $ ./traverse\_tree.py  html  head  title  meta  body  h2  ul  li  li  li  li  li  p  p |

Данные теги являются частью рассматриваемого HTML-документа.

**BeautifulSoup атрибут children**

При помощи атрибута children можно вывести все дочерние теги.

Python



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | #!/usr/bin/python3    from bs4 import BeautifulSoup    with open("index.html", "r") as f:        contents = f.read()        soup = BeautifulSoup(contents, 'lxml')        root = soup.html        root\_childs = [e.name for e in root.children if e.name is not None]      print(root\_childs) |

В данном примере извлекаются дочерние элементы html тега, после чего они помещаются в список Python и выводятся в консоль. Так как атрибут children также убирает пробелы между тегами, необходимо добавить условие, которое позволяет выбирать только названия тегов.

Shell



|  |  |
| --- | --- |
| 1  2 | $ ./get\_children.py  ['head', 'body'] |

Следовательно, у тегов html есть два дочерних элемента: head и body.

**BeautifulSoup атрибут descendants**

При помощи атрибута descendants можно получить список всех потомков (дочерних элементов всех уровней) рассматриваемого тега.

Python



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | #!/usr/bin/python3    from bs4 import BeautifulSoup    with open("index.html", "r") as f:        contents = f.read()        soup = BeautifulSoup(contents, 'lxml')        root = soup.body        root\_childs = [e.name for e in root.descendants if e.name is not None]      print(root\_childs) |

Данный пример позволяет найти всех потомков главного тега body.

Shell



|  |  |
| --- | --- |
| 1  2 | $ ./get\_descendants.py  ['h2', 'ul', 'li', 'li', 'li', 'li', 'li', 'p', 'p'] |

Перечисленные выше теги являются потомками главного тега body.

**BeautifulSoup и веб-скрапинг HTML**

[Requests](https://python-scripts.com/requests) является простой HTTP библиотекой в Python. Она позволяет использовать разнообразные методы для получения доступа к веб-ресурсам при помощи HTTP.

Python



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | #!/usr/bin/python3    from bs4 import BeautifulSoup  import requests as req    resp = req.get("[http://www.something.com](http://www.something.com/)")    soup = BeautifulSoup(resp.text, 'lxml')    print(soup.title)  print(soup.title.text)  print(soup.title.parent) |

Данный пример извлекает название рассматриваемой веб-страницы. Здесь также выводится имя ее родителя.

Python



|  |  |
| --- | --- |
| 1  2  3 | resp = req.get("[http://www.something.com](http://www.something.com/)")    soup = BeautifulSoup(resp.text, 'lxml') |

Здесь мы получаем информацию о веб-странице.

Python



|  |  |
| --- | --- |
| 1  2  3 | print(soup.title)  print(soup.title.text)  print(soup.title.parent) |

Код выше помогает вывести HTML-код заголовка, его текст, а также HTML-код его родителя.

Shell



|  |  |
| --- | --- |
| 1  2  3  4 | $ ./scraping.py  <title>Something.</title>  Something.  <head><title>Something.</title></head> |

Это результат вывода.

**BeautifulSoup метод prettify()**

При помощи метода prettify() можно добиться того, чтобы HTML-код выглядел аккуратнее.

Python



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | #!/usr/bin/python3    from bs4 import BeautifulSoup  import requests as req    resp = req.get("[http://www.something.com](http://www.something.com/)")    soup = BeautifulSoup(resp.text, 'lxml')    print(soup.prettify()) |

Таким образом, мы оптимизируем HTML-код простой веб-страницы.

Shell



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | $ ./prettify.py  <html>  <head>    <title>     Something.    </title>  </head>  <body>    Something.  </body>  </html> |

Это результат вывода.

**BeautifulSoup метод find(), поиск элементов по id**

При помощи метода find() можно найти элементы страницы, используя различные опорные параметры, id в том числе.

Python



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | #!/usr/bin/python3    from bs4 import BeautifulSoup    with open("index.html", "r") as f:        contents = f.read()        soup = BeautifulSoup(contents, 'lxml')        #print(soup.find("ul", attrs={ "id" : "mylist"}))      print(soup.find("ul", id="mylist")) |

Код в примере находит тег ul, у которого id mylist. Строка в комментарии является альтернативным способом выполнить то же самое задание.

**BeautifulSoup метод find\_all() поиск всех тегов в HTML**

При помощи метода find\_all() можно найти все элементы, которые соответствуют заданным критериям.

Python



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | #!/usr/bin/python3    from bs4 import BeautifulSoup    with open("index.html", "r") as f:        contents = f.read()        soup = BeautifulSoup(contents, 'lxml')        for tag in soup.find\_all("li"):          print("{0}: {1}".format(tag.name, tag.text)) |

Код в примере позволяет найти и вывести на экран все li теги.

Shell



|  |  |
| --- | --- |
| 1  2  3  4  5 | $ ./find\_all.py  li: Solaris  li: FreeBSD  li: Debian  li: NetBSD |

Это результат вывода.

Метод find\_all() также при поиске использует список из названий тегов.

Python



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | #!/usr/bin/python3    from bs4 import BeautifulSoup    with open("index.html", "r") as f:        contents = f.read()        soup = BeautifulSoup(contents, 'lxml')        tags = soup.find\_all(['h2', 'p'])        for tag in tags:          print(" ".join(tag.text.split())) |

В данном примере показано, как найти все h2 и p элементы, после чего вывести их содержимое на экран.

Метод find\_all() также может использовать функцию, которая определяет, какие элементы должны быть выведены.

Python



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | #!/usr/bin/python3    from bs4 import BeautifulSoup    def myfun(tag):        return tag.is\_empty\_element      with open("index.html", "r") as f:        contents = f.read()        soup = BeautifulSoup(contents, 'lxml')        tags = soup.find\_all(myfun)      print(tags) |

Данный пример выводит пустые элементы.

Shell



|  |  |
| --- | --- |
| 1  2 | $ ./find\_by\_fun.py  [<meta charset="utf-8"/>] |

Единственным пустым элементом в документе является meta.

Также можно найти запрашиваемые элементы, используя регулярные выражения.

Python



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | #!/usr/bin/python3    import re    from bs4 import BeautifulSoup    with open("index.html", "r") as f:        contents = f.read()        soup = BeautifulSoup(contents, 'lxml')        strings = soup.find\_all(string=re.compile('BSD'))        for txt in strings:            print(" ".join(txt.split())) |

В данном примере выводится содержимое элементов, в которых есть строка с символами ‘BSD’.

Shell



|  |  |
| --- | --- |
| 1  2  3  4 | $ ./regex.py  FreeBSD  NetBSD  FreeBSD is an advanced computer operating system used to power modern servers, desktops, and embedded platforms. |

Это результат вывода.

**BeautifulSoup методы select() и select\_one() CSS селекторы**

При помощи методов select() и select\_one() для нахождения запрашиваемых элементов можно использовать некоторые CSS селекторы.

Python



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | #!/usr/bin/python3    from bs4 import BeautifulSoup    with open("index.html", "r") as f:        contents = f.read()        soup = BeautifulSoup(contents, 'lxml')        print(soup.select("li:nth-of-type(3)")) |

В данном примере используется **CSS селектор**, который выводит на экран HTML-код третьего по счету элемента li.

Shell



|  |  |
| --- | --- |
| 1  2 | $ ./select\_nth\_tag.py  <li>Debian</li> |

Данный элемент li является третьим в списке.

В CSS символ # используется для выбора тегов по их id-атрибутам.

Python



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | #!/usr/bin/python3    from bs4 import BeautifulSoup    with open("index.html", "r") as f:        contents = f.read()        soup = BeautifulSoup(contents, 'lxml')        print(soup.select\_one("#mylist")) |

В данном примере выводятся элементы, которых есть id под названием mylist.

**BeautifulSoup метод append() добавление нового HTML-тега**

Метод append() добавляет в рассматриваемый HTML-документ новый тег.

Python



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | #!/usr/bin/python3    from bs4 import BeautifulSoup    with open("index.html", "r") as f:        contents = f.read()        soup = BeautifulSoup(contents, 'lxml')        newtag = soup.new\_tag('li')      newtag.string='OpenBSD'        ultag = soup.ul        ultag.append(newtag)        print(ultag.prettify()) |

В примере выше показано, как в HTML-документ добавить новый тег li.

Python



|  |  |
| --- | --- |
| 1  2 | newtag = soup.new\_tag('li')  newtag.string='OpenBSD' |

Для начала, требуется создать новый тег при помощи метода new\_tag().

Python



|  |  |
| --- | --- |
| 1 | ultag = soup.ul |

Далее создается сноска на тег ul.

Python



|  |  |
| --- | --- |
| 1 | ultag.append(newtag) |

Затем созданный ранее тег li добавляется к тегу ul.

Python



|  |  |
| --- | --- |
| 1 | print(ultag.prettify()) |

Таким образом, тег ul выводится аккуратно отформатированным.

**BeautifulSoup метод insert() вставка HTML-тега**

Метод insert() позволяет вставить тег в определенно выбранное место.

Python



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | #!/usr/bin/python3    from bs4 import BeautifulSoup    with open("index.html", "r") as f:        contents = f.read()        soup = BeautifulSoup(contents, 'lxml')        newtag = soup.new\_tag('li')      newtag.string='OpenBSD'        ultag = soup.ul        ultag.insert(2, newtag)        print(ultag.prettify()) |

В примере показано, как поставить тег li на третью позицию в выбранном ul теге.

**BeautifulSoup метод replace\_with() замена текста в теге**

Метод replace\_with() заменяет содержимое выбранного элемента.

Python



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | #!/usr/bin/python3    from bs4 import BeautifulSoup    with open("index.html", "r") as f:        contents = f.read()        soup = BeautifulSoup(contents, 'lxml')        tag = soup.find(text="Windows")      tag.replace\_with("OpenBSD")        print(soup.ul.prettify()) |

В примере показано, как при помощи метода find() найти определенный элемент, а затем, используя метод replace\_with(), заменить его содержимое.

**BeautifulSoup метод decompose() удаление HTML-тега**

Метод decompose() удаляет определенный тег из структуры документа и уничтожает его.

Python



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | #!/usr/bin/python3    from bs4 import BeautifulSoup    with open("index.html", "r") as f:        contents = f.read()        soup = BeautifulSoup(contents, 'lxml')        ptag2 = soup.select\_one("p:nth-of-type(2)")        ptag2.decompose()        print(soup.body.prettify()) |

В данном примере показано, как удалить второй элемент p в документе.

В данном руководстве было показано, как использовать библиотеку BeautifulSoup в Python.