Web Scraping With Beautiful Soup in Python

Web Scraping is a technique employed to extract large amounts of data from websites whereby the data is retrieved and saved to a local file in your computer or to a database in a table (spreadsheet) format. This process of web scraping can be a time-consuming as it requires us to know the layout of the document and isolate the items. However, Python users have access to a robust library that makes the process of web scraping easy & quick. The name of this library is `BeautifulSoup` and comes preinstalled on the Anaconda platform.

In this tutorial, we will explore how to use this library to scrape a Wikipedia page for data.

Tip: If you do not have `BeautifulSoup` installed on Python simply run the follwing pip command `pip install beautifulsoup4`.

Section One: Getting the Web Page Content

The first thing we will do is import the libraries and modules. The most obvious one is the `BeautifulSoup` library as this is the library we will be using to scrape the page. However, to get the HTML content of the web page, we will need one more library; the `urllib.request` Library. This library is used to open the URL we want to scrape and then pass the HTML content of that page into the Beautiful Soup parser.

In [1]:
```
# beautiful soup libraries
from bs4 import BeautifulSoup
from urllib.request import Request, urlopen
import urllib.request
```

With our libraries & modules imported, let's get started by defining the webpage URL & then requesting it. Once we have our request object, we can pass it through our BeautifulSoup object, while making sure to specify the `html.parser` as the content of the response object is HTML. The loading process is the real beauty of the `BeautifulSoup` library, **as it handles all the content for use behind the scenes.**

In [84]:
```
# Define & request the url that we want to scrape
wiki_url = r"https://en.wikipedia.org/wiki/Eastern_Front_(World_War_II)"
html_content = urllib.request.urlopen(wiki_url)

# Pass the html_content(the webpage) through our beautiful soup object
soup = BeautifulSoup(html_content, 'html.parser')
```

Once the content is loaded, we have a new BeautifulSoup object that we can easily traverse. Here is a simple example where we use the `find` method to find the first a tag in our HTML code. Now, we can make our search a little more narrow by defining an attribute we want to identify. In this example, I want the `href` attribute to exist in my a tag.

In [85]:
```
# find the first `a` tag that has an `href` attribute.
link = soup.find('a', href=True)

display(link)
```

```
<a href="/wiki/Wikipedia:Protection_policy#semi" title="This article is semi-
protected until June 21, 2019 at 18:17 UTC."><img alt="Page semi-protected" d
ata-file-height="512" data-file-width="512" decoding="async" height="20" src
="//upload.wikimedia.org/wikipedia/en/thumb/1/1b/Semi-protection-shackle.svg/
20px-Semi-protection-shackle.svg.png" srcset="//upload.wikimedia.org/wikipedi
a/en/thumb/1/1b/Semi-protection-shackle.svg/30px-Semi-protection-shackle.svg.
png 1.5x, //upload.wikimedia.org/wikipedia/en/thumb/1/1b/Semi-protection-shac
kle.svg/40px-Semi-protection-shackle.svg.png 2x" width="20"/></a>
```

Section Two: Tag Objects

It's not apparent at first, but any object that we look for is returned to us as another object that has specific properties about it. To be more explicit, it's a beautiful soup tag object. This particular object has particular features that make retrieving different portions of it more manageable.

- **name:** This is the name of the tag.
- **text:** This is the readable text of the label if there is any.
- **attrs:** This returns a dictionary of each attribute behaving as the key and the corresponding info as the value.

To access any of the attributes, we call the object and then pass through the key which will return the value.

In [4]:
```
# what is the type.
display(type(link))

# define the name.
display(link.name)

# get the dictionary attribute.
display(link.attrs)

# lets get each value in that dictionary, luckily there is only 2.
display(link['href'])
display(link['title'])

# we can also get the text, if there is any. In this case there wasn't any tex
t to return so it just returns an empty string.
display(link.text)
```

```
bs4.element.Tag
```

```
'a'
```

```
{'href': '/wiki/Wikipedia:Protection_policy#semi',
 'title': 'This article is semi-protected until June 21, 2019 at 18:17 UTC.'}
```

```
'/wiki/Wikipedia:Protection_policy#semi'
```

```
'This article is semi-protected until June 21, 2019 at 18:17 UTC.'
```

```
''
```

Section Three: Finding Tags & Content

We've seen how to search for one tag, but what if want to search for multiple tags of the same type. Well, this is almost identical to searching for one tag we change the method we use to **`find_all`**. In this example, I use `find_all` to find all the a tags in the document, I still only want the ones that have a link, so I keep the `href` attribute set to the value `True`.

In [5]:
```
# find all the links in the document.
links = soup.find_all('a', href=True)

# what is the type of this links object, it should be a bs4.element.ResultSet.
display(type(links))

# we can iterate through this result set using a for loop. In this loop I grab
the href and limit the result sets to 10 items.
for link in links[0:10]:
    print(link['href'])
```

```
bs4.element.ResultSet
```

```
/wiki/Wikipedia:Protection_policy#semi
#mw-head
#p-search
/wiki/Great_Patriotic_War_(term)
/wiki/The_Great_Patriotic_War_(The_Americans)
/wiki/Patriotic_War_of_1812
/wiki/European_theatre_of_World_War_II
/wiki/World_War_II
/wiki/File:EasternFrontWWIIcolage.png
/wiki/Ilyushin_Il-2
```

We've seen how using `find_all` can make the process of getting the tags we want significantly easier. However, we just looked at some simple examples. This particular method is very flexible as it allows us to find some of the following:

- Multiple Tags
- Tags with specific attributes
- Tags with specific attributes & attribute values.
- Using functions to narrow our search.

Here are some more "advanced" examples of using `find_all`.

In [6]:
```
# return a list of all the <table> & and <a> tag
tables_and_links = soup.find_all(['a', 'table'])

# return a list of all the <table> tag with a 'content' attribute
tables_with_cont = soup.find_all('th', style="width: 10%;")

# return a list of all the <div> tags that have a class attribute of 'navbox'
div_with_nav = soup.find_all('div','navbox')

# define a function to find the item
def list_with_links(tag):
    return tag.name == 'li' and len(tag.find_all('a'))> 7

# use the function with find_all
list_items_with_links = soup.find_all(list_with_links)
```

Section Four: The Beautiful Soup Family Tree

Navigating the HTML tree is pretty simple, once you understand it's relationship to other elements in the document. For example, multiple items can fall under the same tag. Take, for example, a simple document provided by BeautifulSoup

```
<html>
  <body>
   <a>
    <b>
     text1
    </b>
    <c>
     text2
    </c>
   </a>
  </body>
 </html>
```

Let's explore the different types of relationships available to us in the BeautifulSoup library.

In [7]:
```
# define the simple tree we see above.
simple_tree = """<html><body><a><b>text1</b><c>text2</c></a></body></html>"""

# pass the simple tree into our Parser to create some simple soup.
simple_soup = BeautifulSoup(simple_tree, 'html.parser')

# we can always print it in a familiar structure.
print(simple_soup.prettify())
```

```
<html>
 <body>
  <a>
   <b>
    text1
   </b>
   <c>
    text2
   </c>
  </a>
 </body>
</html>
```

Going down the tree: Children

We notice that in our HTML code above that we have tags within tags, the tags contained in other tags are the **Children** and in Beautiful Soup we have different ways for moving through these children. If you look at the simple tree above, both tag `b` and tag `c` are children of the tag `a`. Let's see how we can access those children.

In [8]:
```
# if we want just a list of the children we could use the Contents attribute
a_content = simple_soup.a.contents

# display the list
display(a_content)

# display the first child
display(a_content[0])

# if we want to create a generator to iterate over them vice a list we can use
the `children` attribute
for child in simple_soup.a.children:
    display(child)
```

```
[<b>text1</b>, <c>text2</c>]
```

```
<b>text1</b>
```

```
<b>text1</b>
```

```
<c>text2</c>
```

Going down the tree: Descendants

When using the `contents` & `childrens` attribute, we are only considering tags direct children, the ones that fall underneath that tag. However, if wanted all the tags beneath the children we would need to use the `descendants` attribute. The `descendants` attribute returns not just the children, but the children within the children.

In [9]:
```
# let's first verify there is a difference
display(len(list(simple_soup.a.children)))
display(len(list(simple_soup.a.descendants)))

# greate so we know there is a difference, lets iterate through the descendant
s
for descendant in simple_soup.a.descendants:
    print(descendant)
```

```
2

4

<b>text1</b>
text1
<c>text2</c>
text2
```

Going up the tree: Parent & Parents

We now see that we can access the tags that are direct descendants of the tag in question. We refer to the accessing of these tags as going **down the tree**. If we can go down the tree, we should be able to go up the tree right? Correct, this is possible using the `parent` & `parents` attribute. The parent is **the tag that contains the tag we are referring to**. Here is how we access the parent/parents:

In [10]:
```
# lets see if tag `a` has a parent
display(simple_soup.a.parent)

# is there a parent of the parent?
display(simple_soup.a.parent.parent)

# looks like there is, but we can write this more concisely using a generator
 along with a for loop
for parent in simple_soup.a.parents:
    print(parent)
```

```
<body><a><b>text1</b><c>text2</c></a></body>

<html><body><a><b>text1</b><c>text2</c></a></body></html>

<body><a><b>text1</b><c>text2</c></a></body>
<html><body><a><b>text1</b><c>text2</c></a></body></html>
<html><body><a><b>text1</b><c>text2</c></a></body></html>
```

Going Sideways

In the simple tree up above, we would consider that element <b> and element <c> are **siblings** because they both fall under the same a tag element **the parent**. The a tag signifies that means we can navigate between the two using the `next_sibling` & `previous_sibling` properties. When we go from one sibling to the next, we are said to be going sideways because they both exist under the same tag.

In [11]:
```
# grab the b tag
display(simple_soup.b)

# grab the next sibling, this should be the c tag.
display(simple_soup.b.next_sibling)

# grab the previous sibling, this should be the b tag.
display(simple_soup.c.previous_sibling)

# if I call 'previous_sibling' on the b tag it will return none, that's becaus
e the 'b' tag has no siblings before it
# grab the next sibling, this should be the c tag.
display(simple_soup.b.previous_sibling)
```

```
<b>text1</b>

<c>text2</c>

<b>text1</b>

None
```

In [12]:
```
# grab the first paragraph, and then get it's sibling
display(soup.p.next_sibling)
display(soup.p.next_sibling.next_sibling)

# define the bottom
bottom = soup.p.next_sibling.next_sibling

# grab the first paragraph, and then get it's sibling
display(bottom.previous_sibling)
display(bottom.previous_sibling.previous_sibling)
```

```
'\n'

<div class="hatnote navigation-not-searchable" role="note">"Great Patriotic W
ar" redirects here. For a discussion of the term itself, see <a href="/wiki/G
reat_Patriotic_War_(term)" title="Great Patriotic War (term)">Great Patriotic
War (term)</a>. For the episode of The Americans, see <a href="/wiki/The_Grea
t_Patriotic_War_(The_Americans)" title="The Great Patriotic War (The American
s)">The Great Patriotic War (The Americans)</a>.</div>

'\n'

<p class="mw-empty-elt">
</p>
```

Going back & forth

One of the simplest ways to navigate our simple tree is to use the `next_element` and the `previous_element` properties. All these two properties do is proceed down from top to bottom. If I want to go forward down my tree, I will use `next_element` and if I wanted to go backward up my tree I would specify the `previous_element`. Keep in mind that you don't have started at the beginning of the code to start traversing it, you can call the tag you want to start at and then go down.

Tip: Now a lot of people ask this, "What's the difference between the next element versus the next sibling?". The answer is simple when we talk about the next element we mean the next item that Beautiful Soup parsed. Just because it was the next item parsed does not mean it's a sibling.

In [13]:
```
# grab the body
display(simple_soup.body)

# grab the next element after the body, which would be the 'a' tag.
display(simple_soup.body.next_element)

# keep going forward
display(simple_soup.body.next_element.next_element)
```

```
<body><a><b>text1</b><c>text2</c></a></body>

<a><b>text1</b><c>text2</c></a>

<b>text1</b>
```

In [14]:
```
# grab the b tag
display(simple_soup.b)

# grab the PREVIOUS element after the body, which would be the 'a' tag.
display(simple_soup.b.previous_element)

# keep going backwards
display(simple_soup.b.previous_element.previous_element)
```

```
<b>text1</b>

<a><b>text1</b><c>text2</c></a>

<body><a><b>text1</b><c>text2</c></a></body>
```

In [15]:
```
# let's do an example with one of our tables
my_table = soup.find_all('table')[5]

# using next element
display(my_table.next_element)
display(my_table.next_element.next_element)

# using previous element
display(my_table.previous_element)
display(my_table.previous_element.previous_element)
```

```
'\n'
```

```
<caption>Comparative strengths of combat forces, Eastern Front, 1941–1945<sup
class="reference" id="cite_ref-37"><a href="#cite_note-37">[37]</a></sup><sup
class="reference" id="cite_ref-FOOTNOTEGlantz1998107_38-0"><a href="#cite_not
e-FOOTNOTEGlantz1998107-38">[38]</a></sup><sup class="reference" id="cite_ref
-FOOTNOTEGlantzHouse199568_39-0"><a href="#cite_note-FOOTNOTEGlantzHouse19956
8-39">[39]</a></sup>
</caption>
```

```
'\n'
```

```
', leader of the Free French, who thought that it was important for French se
rvicemen to serve on all fronts.\n'
```

Sibling & Element Generators

What if I want all the siblings or all the next/previous elements that belong to a particular tag? Well up above, I made the process a little more complicated then it had to. What we could've done up above is instead of writing `next_sibling.next_sibling` we could've just called the `next_siblings` iterator to simple fetch all the siblings of that particular tag, and then we could iterate through each one of them. Here is how the above examples would look if we chose to follow that path:

In [16]:
```
# grab the body
display(simple_soup.body)

# create a next_elements generator
display(simple_soup.body.next_elements)

# loop through each element in the generator.
for element in simple_soup.body.next_elements:
    display(element)
```

```
<body><a><b>text1</b><c>text2</c></a></body>

<generator object next_elements at 0x0000020724935DB0>

<a><b>text1</b><c>text2</c></a>

<b>text1</b>

'text1'

<c>text2</c>

'text2'
```

In [17]:
```
# grab the b tag
display(simple_soup.b)

# create a previous_elements generator
display(simple_soup.b.previous_elements)

# loop through each element in the generator.
for element in simple_soup.b.previous_elements:
    display(element)
```

```
<b>text1</b>

<generator object previous_elements at 0x0000020726751DB0>

<a><b>text1</b><c>text2</c></a>

<body><a><b>text1</b><c>text2</c></a></body>

<html><body><a><b>text1</b><c>text2</c></a></body></html>
```

Find_Parent & Find_Parents

Okay, instead of navigating through the tags based on their relationship to other tags. What if we wanted to find a particular tags family member? Is this possible? It is! The first example we will look is using `find_parent` & `find_parents`, both of these will find a particular tags parent or parents.

In [29]:
```
simple_soup.b

display(simple_soup.b.find_parent())

display(simple_soup.b.find_parents())
```

```
<a><b>text1</b><c>text2</c></a>

[<a><b>text1</b><c>text2</c></a>,
 <body><a><b>text1</b><c>text2</c></a></body>,
 <html><body><a><b>text1</b><c>text2</c></a></body></html>,
 <html><body><a><b>text1</b><c>text2</c></a></body></html>]
```

Finding Siblings & elements

If we can search for parents, can we search for siblings and elements related to a tag? Is it possible to find the previous/next elements or tags for a given tag? The answer is yes, the examples below are identical to the `find_parent` & `find_parents` but now we are dealing with both `element` & `siblings`.

In [32]:
```
simple_soup.b

display(simple_soup.b.find_next_sibling())

display(simple_soup.b.find_next_siblings())
```

```
<c>text2</c>

[<c>text2</c>]
```

In [33]:
```
simple_soup.c

display(simple_soup.c.find_previous_sibling())

display(simple_soup.c.find_previous_siblings())
```

```
<b>text1</b>

[<b>text1</b>]
```

In [37]:
```
simple_soup.a

display(simple_soup.a.find_next())

display(simple_soup.a.find_all_next())
```

```
<b>text1</b>

[<b>text1</b>, <c>text2</c>]
```

In [38]:
```
simple_soup.c

display(simple_soup.c.find_previous())

display(simple_soup.c.find_all_previous())
```

```
<b>text1</b>
```

```
[<b>text1</b>,
 <a><b>text1</b><c>text2</c></a>,
 <body><a><b>text1</b><c>text2</c></a></body>,
 <html><body><a><b>text1</b><c>text2</c></a></body></html>]
```

Section Four Extracting & Manipulating Text

In the previous examples, we saw how to navigate and find tags that we are searching for. Assuming we found the tag we want, what are some other ways we can manipulate them? In the following examples, we will explore how to manage tags to get the information we want.

In [61]:
```
# get my table
table = soup.find_all('table')[5]
print(table.prettify())
```

```
<table class="wikitable">
 <caption>
  Comparative strengths of combat forces, Eastern Front, 1941–1945
  <sup class="reference" id="cite_ref-37">
   <a href="#cite_note-37">
    [37]
   </a>
  </sup>
  <sup class="reference" id="cite_ref-FOOTNOTEGlantz1998107_38-0">
   <a href="#cite_note-FOOTNOTEGlantz1998107-38">
    [38]
   </a>
  </sup>
  <sup class="reference" id="cite_ref-FOOTNOTEGlantzHouse199568_39-0">
   <a href="#cite_note-FOOTNOTEGlantzHouse199568-39">
    [39]
   </a>
  </sup>
 </caption>
 <tbody>
  <tr>
   <th style="my new width">
    Date
   </th>
   <th style="width: 45%;">
    Axis forces
   </th>
   <th style="width: 45%;">
    Soviet forces
   </th>
  </tr>
  <tr>
   <td>
    June 1941
   </td>
   <td>
    3,050,000 Germans, 67,000 (northern Norway); 500,000 Finns, 150,000 Roman
ians,
    <a href="/wiki/Italian_Expeditionary_Corps_in_Russia" title="Italian Expe
ditionary Corps in Russia">
     62,000 Italians
    </a>
    <br/>
    Total:
    <b>
     3,829,000
    </b>
    in the east (80% of the German Army)
   </td>
   <td>
    2,680,000 active in Western Military Districts out of
    <b>
     5,500,000
    </b>
    (overall); 14,000,000 mobilizable reserves (former conscripts who fulfill
ed their mandatory service prior)
   </td>
```

```
  </tr>
  <tr>
   <td>
    June 1942
   </td>
   <td>
    2,600,000 Germans, 90,000 (northern Norway); 430,000 Finns, 600,000 Roman
ians, Hungarians, and Italians
    <br/>
    Total:
    <b>
     3,720,000
    </b>
    in the east (80% of the German Army)
   </td>
   <td>
    5,313,000 (front); 383,000 (hospital)
    <br/>
    Total:
    <b>
     9,350,000
    </b>
   </td>
  </tr>
  <tr>
   <td>
    July 1943
   </td>
   <td>
    3,403,000 Germans, 80,000 (northern Norway); 400,000 Finns, 150,000 Roman
ians and Hungarians
    <br/>
    Total:
    <b>
     3,933,000
    </b>
    in the east (63% of the German Army)
   </td>
   <td>
    6,724,000 (front); 446,445 (hospital);
    <br/>
    Total:
    <b>
     10,300,000
    </b>
   </td>
  </tr>
  <tr>
   <td>
    June 1944
   </td>
   <td>
    2,460,000 Germans, 60,000 (northern Norway); 300,000 Finns, 550,000 Roman
ians and Hungarians
    <br/>
    Total:
    <b>
```

```
      3,370,000
     </b>
     in the east (62% of the German Army)
    </td>
    <td>
     <b>
      6,425,000
     </b>
    </td>
   </tr>
   <tr>
    <td>
     Jan. 1945
    </td>
    <td>
     2,230,000 Germans, 100,000 Hungarians
     <br/>
     Total:
     <b>
      2,330,000
     </b>
     in the east (60% of the German Army)
    </td>
    <td>
     <b>
      6,532,000
     </b>
     (360,000 Poles, Romanians, Bulgarians, and Czechs)
    </td>
   </tr>
   <tr>
    <td>
     Apr. 1945
    </td>
    <td>
     <b>
      1,960,000
     </b>
    </td>
    <td>
     <b>
      6,410,000
     </b>
     (450,000 Poles, Romanians, Bulgarians, and Czechs)
    </td>
   </tr>
  </tbody>
 </table>
```

Changing Attribute Values

Maybe you're not happy with the current value of an attribute in the HTML code, well lucky for you we can change the attribute value using beautiful soup. In the example, we call the attribute in question and then assign it a new value.

In [66]:
```
# grab a header tag from the table
header_tag = table.a

# display the style attribute content
display(header_tag['href'])

# change the style attribute content
header_tag['style'] = 'my new width'

# display the changes
header_tag
```

```
'#cite_note-37'
```

Out[66]:
```
<a href="#cite_note-37" style="my new width">[37]</a>
```

Same goes for the string, if we want to add a string to it we can.

In [67]:
```
# add a string to a tag
table.a.string = 'My New String'

# display the tag with a new string now.
table.a
```

Out[67]:
```
<a href="#cite_note-37" style="my new width">My New String</a>
```

Deleting Attribute Values

The alternative to adding or changing a value is deleting it. Well again, BeautifulSoup offers a mechanism where we can easily remove tags and content from our soup.

In [71]:
```
#removes the content of a tag
tag = table.a

display(tag)

tag.clear()

tag
```

```
<a href="#cite_note-37" style="my new width">My New String</a>
```

Out[71]:
```
<a href="#cite_note-37" style="my new width"></a>
```

Extracting Tags

Sometimes we want to work with a tag individually and not associated with any of its descendants or parents. If we use the `extract()` method, we yank that tag out of the HTML code and assign it to a new variable. Keep in mind once it's yanked, it's yanked. If you want the first soup to contain it again, you'll have to run the full code again.

In [ ]:
```
display(table.tbody)

th_tag = table.tbody.th.extract()

display(th_tag)
```

Extracting Strings

Many times I would say most, we are scraping a website for particular text. Make the process of getting the text accessible and quick by leveraging the built-in `string` & `strings` attributes which return the first or all strings in the document, respectively. Remember that if you use the `strings` attribute, it produces a generator which you can loop through. Also, if you want all the strings minus the `t\` & `n\` use the `stripped_strings` attribute instead.

In [81]:
```
# grab a single string that belongs to the table header
display(table.th.string)

# get all the strings from the table body using the strings attribute.
for string in list(table.tbody.strings)[0:10]:
    print(string)

print('-----------------------------------------------')

# get all the strings from the table body using the strings attribute.
for string in list(table.tbody.stripped_strings)[0:10]:
    print(string)
```

```
'Axis forces\n'



Axis forces



Soviet forces







June 1941


3,050,000 Germans, 67,000 (northern Norway); 500,000 Finns, 150,000 Romanian
s,
-----------------------------------------------
Axis forces
Soviet forces
June 1941
3,050,000 Germans, 67,000 (northern Norway); 500,000 Finns, 150,000 Romanian
s,
62,000 Italians
Total:
3,829,000
in the east (80% of the German Army)
2,680,000 active in Western Military Districts out of
5,500,000
```

More Wikipedia Examples

Okay, so now that we know, the whole structure of our tree let's put it to work. In the example below I will find all the tables in my soup, specify the sixth one because I like that one, and then create all of my necessary generators.

In [86]:
```
# grab all the tables
tables = soup.find_all('table')

# loop through each table
for table in tables:

    # get all the strings from the table body using the strings attribute.
    for string in list(table.tbody.stripped_strings)[0:10]:
        print(string)
```

Eastern Front
Part of the
European theatre
of
World War II
Clockwise from top left
: Soviet
Il-2
ground attack aircraft in Berlin sky; German
Tiger I
Date
22 June 1941
(
1941-06-22
)
– 25 May 1945
(
1945-05-25
)
[2]
v
t
e
Campaigns of
World War II
Europe
Poland
Phoney War
Winter War
Denmark & Norway
v
t
e
Eastern Front
Naval warfare
Baltic Sea
Black Sea
Arctic Ocean
1941
Barbarossa
Part of
a series
on the
History of the
Union of Soviet
Socialist Republics
1917–1927
Revolutionary beginnings
Revolution
Civil War
Date
Axis forces
Soviet forces
June 1941
3,050,000 Germans, 67,000 (northern Norway); 500,000 Finns, 150,000 Romanian
s,
62,000 Italians

```
Total:
3,829,000
in the east (80% of the German Army)
2,680,000 active in Western Military Districts out of
Year
Amount
(tons)
%
1941
360,778
2.1
1942
2,453,097
14
This section
does not
cite
any
sources
.
Please help
improve this section
by
adding citations to reliable sources
This section
does not
cite
any
sources
.
Please help
improve this section
by
adding citations to reliable sources
This section
needs additional citations for
verification
.
Please help
improve this article
by
adding citations to reliable sources
. Unsourced material may be challenged and removed.
Find sources:
This section
does not
cite
any
sources
.
Please help
improve this section
by
adding citations to reliable sources
This section
needs additional citations for
verification
```

.
Please help
improve this article
by
adding citations to reliable sources
. Unsourced material may be challenged and removed.
Find sources:
Year
Coal
(million tonnes, Germany includes lignite and bituminous types)
Steel
(million tonnes)
Aluminium
(thousand tonnes)
Oil
(million tonnes)
German
Year
Tanks and self-
propelled guns
Soviet
German
Italian
Hungarian
Romanian
Japanese
1941
Year
Aircraft
Soviet
German
Italian
Hungarian
Romanian
Japanese
1941
15,735
Year
Industrial Labour
Foreign Labour
Total Labour
Soviet
German
Soviet
German
Total Soviet
Total German
Forces fighting with the Axis
Total Dead
KIA
/
DOW
/
MIA
Prisoners taken by the Soviets
Prisoners who died in Captivity
WIA

Forces fighting with the Soviet Union
Total Dead
KIA/DOW/MIA
Prisoners taken by the Axis
Prisoners who died in captivity
WIA (not including DOW)
Soviet
8,668,400–10,000,000
6,829,600
4,059,000 (military personnel only)–5,700,000
Wikisource
has original text related to this article:
Adolf Hitler's Letter to Benito Mussolini Explaining the Invasion of the Sovi
et Union
Wikisource
has original text related to this article:
The Führer to the German People: 22 June 1941
Wikisource
has original text related to this article:
Adolf Hitler's Order of the Day to the German Troops on the Eastern Front (2
October 1941)
Wikisource
has original text related to this article:
Adolf Hitler Explains His Reasons for Invading the Soviet Union
Wikisource
has original text related to this article:
Adolf Hitler's Proclamation to the German Army on His Assumption of Direct Co
mmand
Wikisource
has original text related to this article:
Adolf Hitler's Speech at the Berlin Sportpalast (30 January 1942)
Portals
Access related topics
Hungary portal
Military of Germany portal
Poland portal
Romania portal
Soviet Union portal
World War II portal
Find out more on Wikipedia's
Sister projects
v
t
e
World War II
Outline
Military engagements
Battles
Operations
Commanders
Casualties
Allies
.mw-parser-output .nobold{font-weight:normal}
(
leaders
)
Australia

Belgium
Brazil
Canada
China
Prelude
Africa
Asia
Europe
1939
Poland
Phoney War
Winter War
Atlantic
Changsha
General
Famines
Bengal famine of 1943
Chinese famine of 1942–43
Greek Famine of 1941-1944
Dutch famine of 1944–45
Vietnamese Famine of 1945
Air warfare of World War II
In Europe
Blitzkrieg
Famines
Bengal famine of 1943
Chinese famine of 1942–43
Greek Famine of 1941-1944
Dutch famine of 1944–45
Vietnamese Famine of 1945
Air warfare of World War II
In Europe
Blitzkrieg
Comparative military ranks
v
t
e
Wehrmacht
Army Group Rear Areas
during the
German–Soviet War
, 1941–45
Army Group Rear Area
North
v
t
e
Annual
Moscow military parade on Victory Day in the Great Patriotic War 1941-1945
By year
1945
1965
1985
1990
v
t
e

```
Armed conflicts involving
Russia
(incl.
Imperial
and
Soviet
times)
```

In [18]:
```
# get my table
table = soup.find_all('tr')[5]

# define my generators
the_parents = table.parents
the_children = table.children
the_descendants = table.descendants

next_sibs = table.next_siblings
prev_sibs = table.previous_siblings
next_elem = table.next_elements
prev_elem = table.previous_elements
```

In [19]:
```
for child in list(the_children):
    print('-----------------------------')
    print(child)
```

```
-----------------------------
<th style="padding-right:1em">Location</th>
-----------------------------
<td><div class="location">Europe east of Germany: <a href="/wiki/Central_and_
Eastern_Europe" title="Central and Eastern Europe">Central and Eastern Europe
</a>, in later stages Germany and Austria</div></td>
```

In [20]:
```
for descendant in the_descendants:
    print('----------------------------')
    print(descendant)
```

```
----------------------------
<th style="padding-right:1em">Location</th>
----------------------------
Location
----------------------------
<td><div class="location">Europe east of Germany: <a href="/wiki/Central_and_
Eastern_Europe" title="Central and Eastern Europe">Central and Eastern Europe
</a>, in later stages Germany and Austria</div></td>
----------------------------
<div class="location">Europe east of Germany: <a href="/wiki/Central_and_East
ern_Europe" title="Central and Eastern Europe">Central and Eastern Europe</a
>, in later stages Germany and Austria</div>
----------------------------
Europe east of Germany:
----------------------------
<a href="/wiki/Central_and_Eastern_Europe" title="Central and Eastern Europ
e">Central and Eastern Europe</a>
----------------------------
Central and Eastern Europe
----------------------------
, in later stages Germany and Austria
```

In [21]:
```
for sib in next_sibs:
    print('----------------------------')
    print(sib)
```

```
----------------------------
<tr><th style="padding-right:1em">Result</th><td>
<p>Soviet victory
</p>
<ul><li>Fall of the <a class="mw-redirect" href="/wiki/Third_Reich" title="Th
ird Reich">Third Reich</a></li>
<li><a class="mw-redirect" href="/wiki/Allied_occupation_of_Germany" title="A
llied occupation of Germany">Allied occupation of Germany</a></li>
<li>Beginning of the <a href="/wiki/Cold_War" title="Cold War">Cold War</a> a
nd creation of the <a href="/wiki/Eastern_Bloc" title="Eastern Bloc">Eastern
Bloc</a> and the <a href="/wiki/Iron_Curtain" title="Iron Curtain">Iron Curta
in</a></li>
<li>Beginning of the <a href="/wiki/Greek_Civil_War" title="Greek Civil War">
Greek Civil War</a></li></ul></td></tr>
----------------------------
<tr><th style="padding-right:1em">Territorial<br/>changes</th><td>
<li>The Soviet Union occupies <a href="/wiki/Central_Europe" title="Central E
urope">Central</a>, <a href="/wiki/Eastern_Europe" title="Eastern Europe">Eas
tern</a>, <a class="mw-redirect" href="/wiki/Northeastern_Europe" title="Nort
heastern Europe">Northeastern</a> and <a class="mw-redirect" href="/wiki/Sout
heastern_Europe" title="Southeastern Europe">Southeastern Europe</a> and esta
blishes pro-Soviet <a href="/wiki/Communist_state" title="Communist state">co
mmunist</a> <a href="/wiki/Puppet_state" title="Puppet state">puppet governme
nts</a> in countries including <a href="/wiki/People%27s_Republic_of_Bulgari
a" title="People's Republic of Bulgaria">Bulgaria</a>, <a href="/wiki/Czechos
lovak_Socialist_Republic" title="Czechoslovak Socialist Republic">Czechoslova
kia</a>, <a href="/wiki/Hungarian_People%27s_Republic" title="Hungarian Peopl
e's Republic">Hungary</a>, <a href="/wiki/Polish_People%27s_Republic" title
="Polish People's Republic">Poland</a>, <a class="mw-redirect" href="/wiki/Ro
manian_People%27s_Republic" title="Romanian People's Republic">Romania</a>, a
nd <a href="/wiki/East_Germany" title="East Germany">Eastern Germany</a>.</li
>
<li>Establishment of the <a class="mw-redirect" href="/wiki/Federal_People%27
s_Republic_of_Yugoslavia" title="Federal People's Republic of Yugoslavia">Fed
eral People's Republic of Yugoslavia</a></li>
<li><a href="/wiki/History_of_Germany_(1945%E2%80%931990)" title="History of
Germany (1945–1990)">Partition of Germany</a></li>
<li><a href="/wiki/Territorial_changes_of_Poland_immediately_after_World_War_
II" title="Territorial changes of Poland immediately after World War II">Bord
ers of Poland changed</a>.</li></td></tr>
```

In [22]:
```
for sib in prev_sibs:
    print('----------------------------')
    print(sib)
```

```
----------------------------
<tr><th style="padding-right:1em">Date</th><td>22 June 1941<span style="displ
ay:none"> (<span class="bday dtstart published updated">1941-06-22</span>)</s
pan> – 25 May 1945<span style="display:none"> (<span class="bday dtstart publ
ished updated">1945-05-25</span>)</span><sup class="reference" id="cite_ref-
2"><a href="#cite_note-2">[2]</a></sup><br/>(3 years, 11 months and 3 days)</
td></tr>
```

In [23]:
```
# just for speed purposes, let's only do a few next elements
next_list = list(next_elem)

for elem in next_list[0:2]:
    print('----------------------------')
    print(elem)
```

```
----------------------------
<th style="padding-right:1em">Location</th>
----------------------------
Location
```

In [24]:
```
# just for speed purposes, let's only do a few previous elements
prev_list = list(prev_elem)

for elem in prev_list[0:2]:
    print('----------------------------')
    print(elem)
```

```
----------------------------
(3 years, 11 months and 3 days)
----------------------------
<br/>
```

In [25]:
```
# just for speed purposes, let's only do a few parents
par_list = list(the_parents)

for parent in par_list[0:2]:
    print('----------------------------')
    print(parent)
```

```
----------------------------
<tbody><tr><th style="padding-right:1em">Date</th><td>22 June 1941<span style
="display:none"> (<span class="bday dtstart published updated">1941-06-22</sp
an>)</span> – 25 May 1945<span style="display:none"> (<span class="bday dtsta
rt published updated">1945-05-25</span>)</span><sup class="reference" id="cit
e_ref-2"><a href="#cite_note-2">[2]</a></sup><br/>(3 years, 11 months and 3 d
ays)</td></tr><tr><th style="padding-right:1em">Location</th><td><div class
="location">Europe east of Germany: <a href="/wiki/Central_and_Eastern_Europ
e" title="Central and Eastern Europe">Central and Eastern Europe</a>, in late
r stages Germany and Austria</div></td></tr><tr><th style="padding-right:1e
m">Result</th><td>
<p>Soviet victory
</p>
<ul><li>Fall of the <a class="mw-redirect" href="/wiki/Third_Reich" title="Th
ird Reich">Third Reich</a></li>
<li><a class="mw-redirect" href="/wiki/Allied_occupation_of_Germany" title="A
llied occupation of Germany">Allied occupation of Germany</a></li>
<li>Beginning of the <a href="/wiki/Cold_War" title="Cold War">Cold War</a> a
nd creation of the <a href="/wiki/Eastern_Bloc" title="Eastern Bloc">Eastern
Bloc</a> and the <a href="/wiki/Iron_Curtain" title="Iron Curtain">Iron Curta
in</a></li>
<li>Beginning of the <a href="/wiki/Greek_Civil_War" title="Greek Civil War">
Greek Civil War</a></li></ul></td></tr><tr><th style="padding-right:1em">Terr
itorial<br/>changes</th><td>
<li>The Soviet Union occupies <a href="/wiki/Central_Europe" title="Central E
urope">Central</a>, <a href="/wiki/Eastern_Europe" title="Eastern Europe">Eas
tern</a>, <a class="mw-redirect" href="/wiki/Northeastern_Europe" title="Nort
heastern Europe">Northeastern</a> and <a class="mw-redirect" href="/wiki/Sout
heastern_Europe" title="Southeastern Europe">Southeastern Europe</a> and esta
blishes pro-Soviet <a href="/wiki/Communist_state" title="Communist state">co
mmunist</a> <a href="/wiki/Puppet_state" title="Puppet state">puppet governme
nts</a> in countries including <a href="/wiki/People%27s_Republic_of_Bulgari
a" title="People's Republic of Bulgaria">Bulgaria</a>, <a href="/wiki/Czechos
lovak_Socialist_Republic" title="Czechoslovak Socialist Republic">Czechoslova
kia</a>, <a href="/wiki/Hungarian_People%27s_Republic" title="Hungarian Peopl
e's Republic">Hungary</a>, <a href="/wiki/Polish_People%27s_Republic" title
="Polish People's Republic">Poland</a>, <a class="mw-redirect" href="/wiki/Ro
manian_People%27s_Republic" title="Romanian People's Republic">Romania</a>, a
nd <a href="/wiki/East_Germany" title="East Germany">Eastern Germany</a>.</li
>
<li>Establishment of the <a class="mw-redirect" href="/wiki/Federal_People%27
s_Republic_of_Yugoslavia" title="Federal People's Republic of Yugoslavia">Fed
eral People's Republic of Yugoslavia</a></li>
<li><a href="/wiki/History_of_Germany_(1945%E2%80%931990)" title="History of
Germany (1945–1990)">Partition of Germany</a></li>
<li><a href="/wiki/Territorial_changes_of_Poland_immediately_after_World_War_
II" title="Territorial changes of Poland immediately after World War II">Bord
ers of Poland changed</a>.</li></td></tr></tbody>
----------------------------
<table style="width:100%;margin:0;padding:0;border:0"><tbody><tr><th style="p
adding-right:1em">Date</th><td>22 June 1941<span style="display:none"> (<span
class="bday dtstart published updated">1941-06-22</span>)</span> – 25 May 194
5<span style="display:none"> (<span class="bday dtstart published updated">19
45-05-25</span>)</span><sup class="reference" id="cite_ref-2"><a href="#cite_
note-2">[2]</a></sup><br/>(3 years, 11 months and 3 days)</td></tr><tr><th st
yle="padding-right:1em">Location</th><td><div class="location">Europe east of
Germany: <a href="/wiki/Central_and_Eastern_Europe" title="Central and Easter
```

n Europe">Central and Eastern Europe, in later stages Germany and Austria
</td></tr><tr><th style="padding-right:1em">Result</th><td>
Soviet victory

Fall of the <a class="mw-redirect" href="/wiki/Third_Reich" title="Th
ird Reich">Third Reich
<a class="mw-redirect" href="/wiki/Allied_occupation_of_Germany" title="A
llied occupation of Germany">Allied occupation of Germany
Beginning of the Cold War a
nd creation of the Eastern
Bloc and the Iron Curta
in
Beginning of the
Greek Civil War</td></tr><tr><th style="padding-right:1em">Terr
itorial
changes</th><td>
The Soviet Union occupies <a href="/wiki/Central_Europe" title="Central E
urope">Central, Eas
tern, <a class="mw-redirect" href="/wiki/Northeastern_Europe" title="Nort
heastern Europe">Northeastern and <a class="mw-redirect" href="/wiki/Sout
heastern_Europe" title="Southeastern Europe">Southeastern Europe and esta
blishes pro-Soviet co
mmunist puppet governme
nts in countries including <a href="/wiki/People%27s_Republic_of_Bulgari
a" title="People's Republic of Bulgaria">Bulgaria, <a href="/wiki/Czechos
lovak_Socialist_Republic" title="Czechoslovak Socialist Republic">Czechoslova
kia, <a href="/wiki/Hungarian_People%27s_Republic" title="Hungarian Peopl
e's Republic">Hungary, <a href="/wiki/Polish_People%27s_Republic" title
="Polish People's Republic">Poland, <a class="mw-redirect" href="/wiki/Ro
manian_People%27s_Republic" title="Romanian People's Republic">Romania, a
nd Eastern Germany.
Establishment of the <a class="mw-redirect" href="/wiki/Federal_People%27
s_Republic_of_Yugoslavia" title="Federal People's Republic of Yugoslavia">Fed
eral People's Republic of Yugoslavia
<a href="/wiki/History_of_Germany_(1945%E2%80%931990)" title="History of
Germany (1945–1990)">Partition of Germany
<a href="/wiki/Territorial_changes_of_Poland_immediately_after_World_War_
II" title="Territorial changes of Poland immediately after World War II">Bord
ers of Poland changed.</td></tr></tbody></table>