ENGR 102 PROGRAMMING PRACTICE

WEEK 10

Searching & Ranking

Search Engine

1. **Crawl to collect documents.**
2. Index to improve search.
3. Query for a select set of documents.
4. Rank the documents

Search Engine

- Create a Python module (`searchengine.py`).
- The module will have two classes:
  - one for crawling and creating the database, and
  - the other for doing full-text searches by querying the database, as well as ranking.

Crawler Code

- `urllib2:` download web pages
- `BeautifulSoup:` build a structured representation of web pages.
- Using `urllib2` and `BeautifulSoup`, you can build a crawler that will take a list of URLs to index and crawl their links to find other pages to index.

Using urllib2

- Makes it easy to download web pages
- Input: a URL

```
import urllib2
c = urllib2.urlopen('http://cs.sehir.edu.tr')
contents = c.read()
print contents[0:50]
```

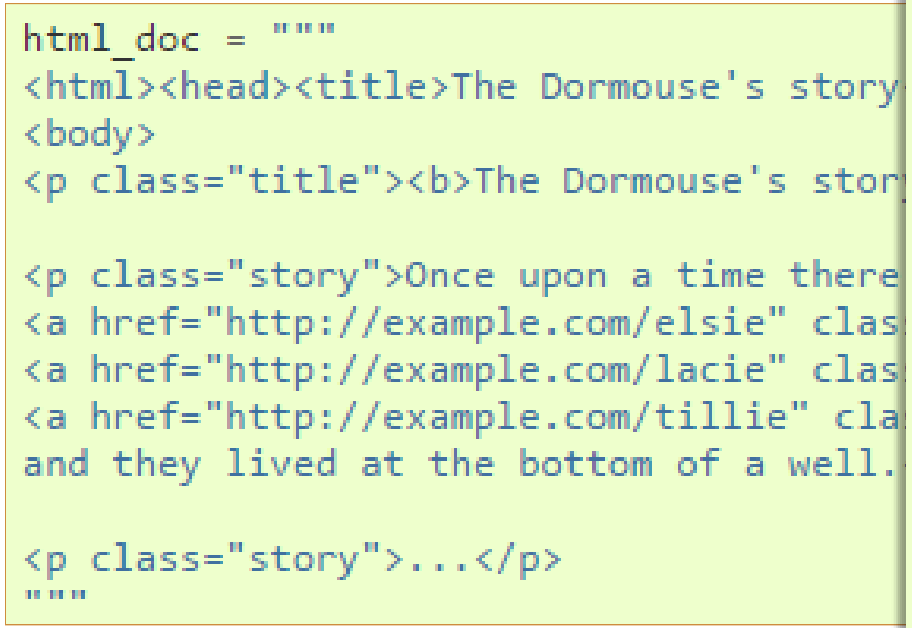
Beautiful Soup

- To parse a web page and build a structured representation.
- To access any element of the page by type, ID, or any of its properties, and to get a string representation of its contents.
- Install BeautifulSoup4 on PyCharm (make sure that it is version 4 not 3.x)
- Usually used with `urllib2`

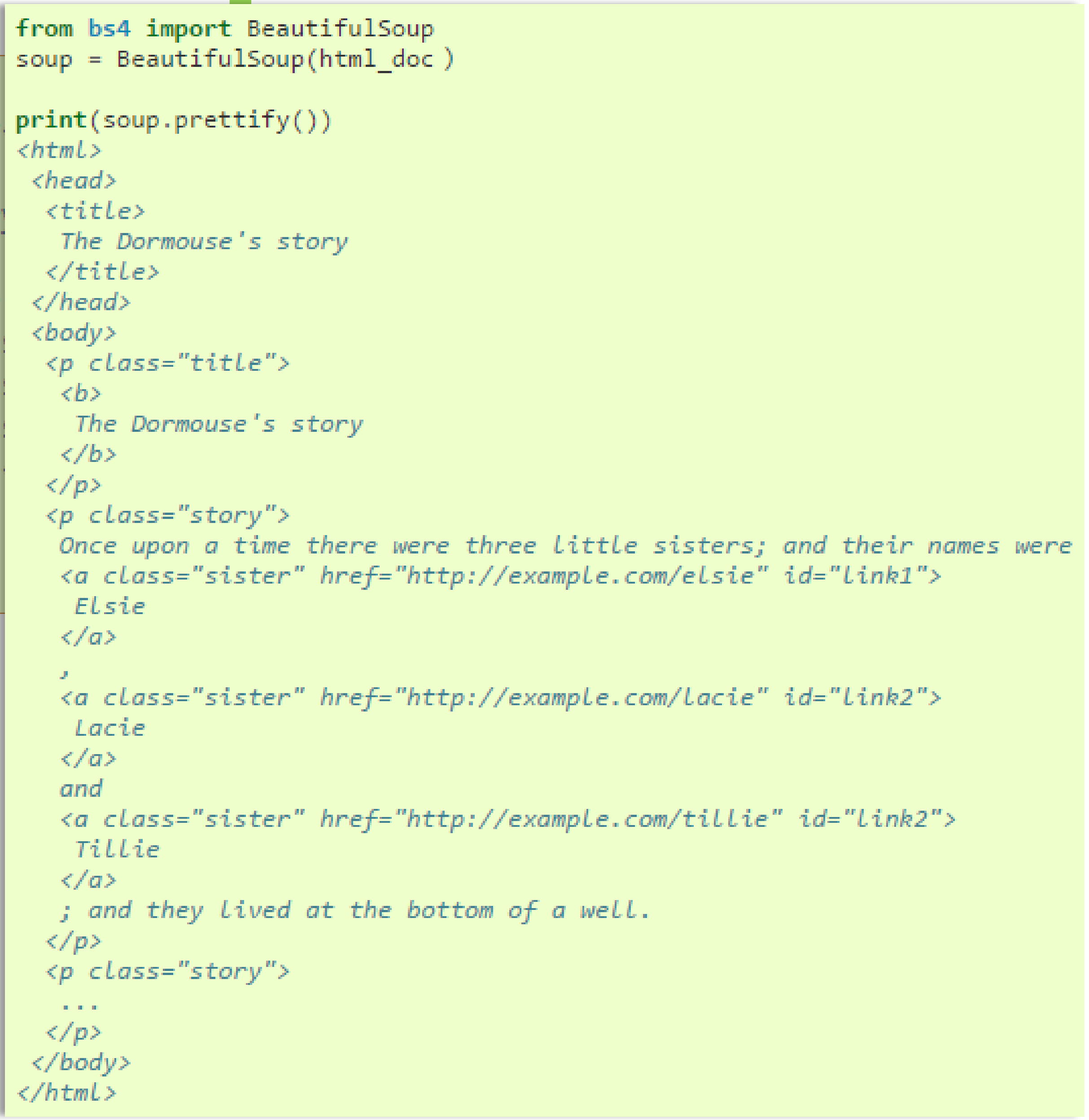
Example

- Write a function when given a tag, returns all html links under the tag
- print the first <p> tag located in html.body
- print the class of first <p> tag located in html.body
- print all html links under the first <p> tag located in html.body
- print all links under tags with class name “story”

Beautiful Soup



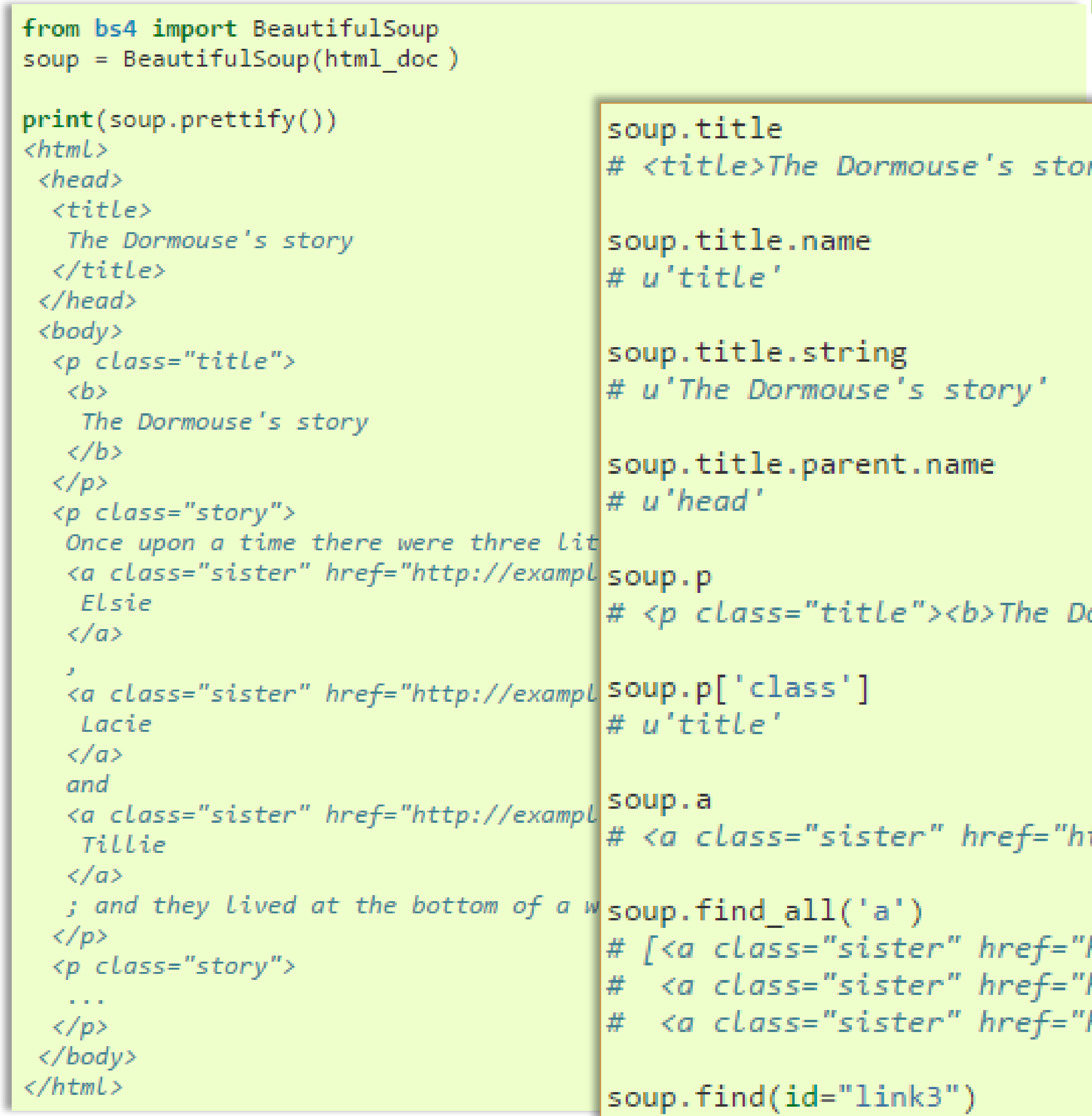
```
html_doc = """
<html><head><title>The Dormouse's story
<body>
<p class="title"><b>The Dormouse's story

<p class="story">Once upon a time there
<a href="http://example.com/elsie" clas
<a href="http://example.com/lacie" clas
<a href="http://example.com/tillie" cla
and they lived at the bottom of a well.

<p class="story">...</p>
"""
```



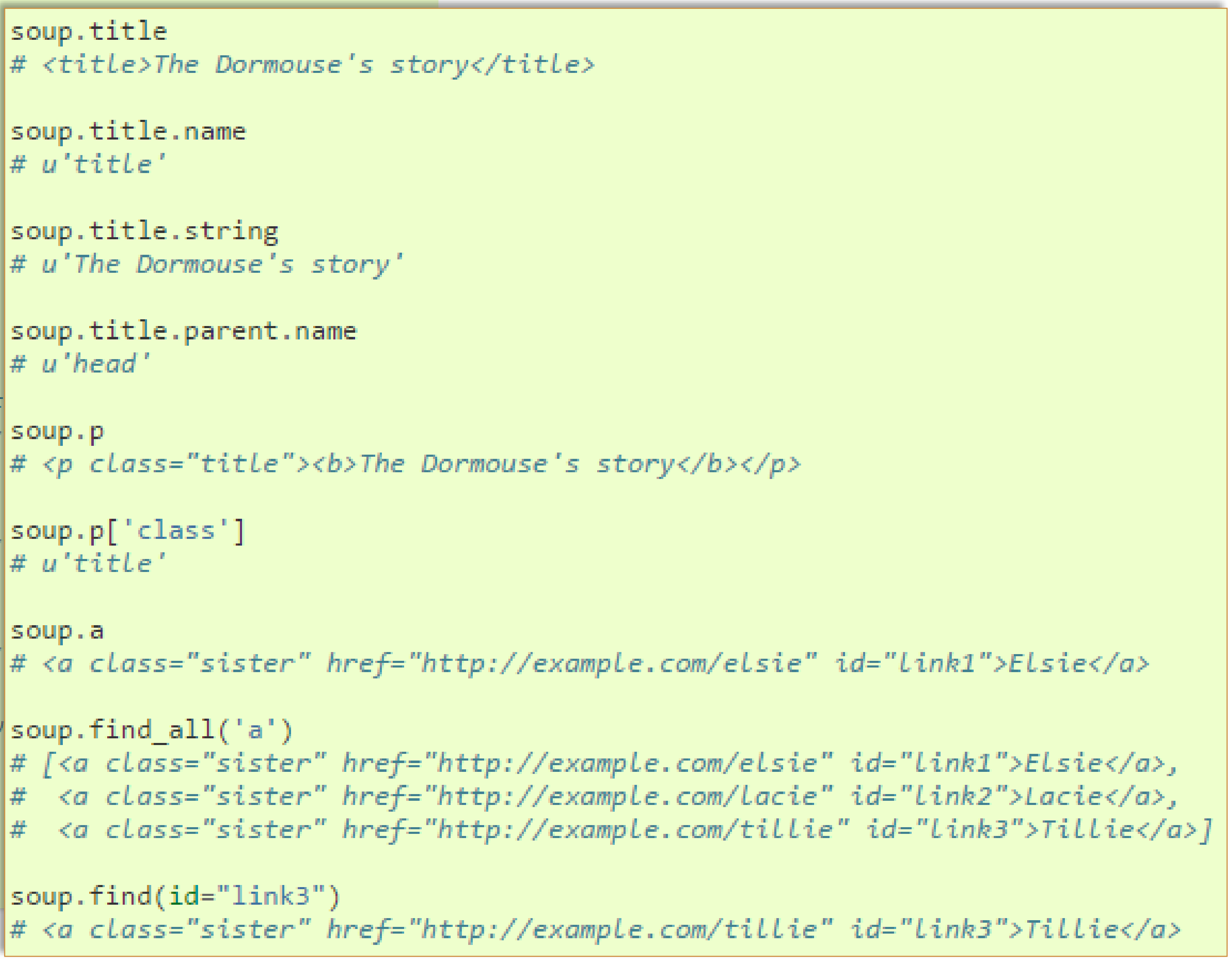
```
from bs4 import BeautifulSoup
soup = BeautifulSoup(html_doc )

print(soup.prettify())
<html>
 <head>
  <title>
   The Dormouse's story
  </title>
 </head>
 <body>
  <p class="title">
   <b>
    The Dormouse's story
   </b>
  </p>
  <p class="story">
   Once upon a time there were three little sisters; and their names were
   <a class="sister" href="http://example.com/elsie" id="link1">
    Elsie
   </a>
   ,
   <a class="sister" href="http://example.com/lacie" id="link2">
    Lacie
   </a>
   and
   <a class="sister" href="http://example.com/tillie" id="link2">
    Tillie
   </a>
   ; and they lived at the bottom of a well.
  </p>
  <p class="story">
   ...
  </p>
 </body>
</html>
```

https://www.crummy.com/software/BeautifulSoup/bs4/doc/

Beautiful Soup



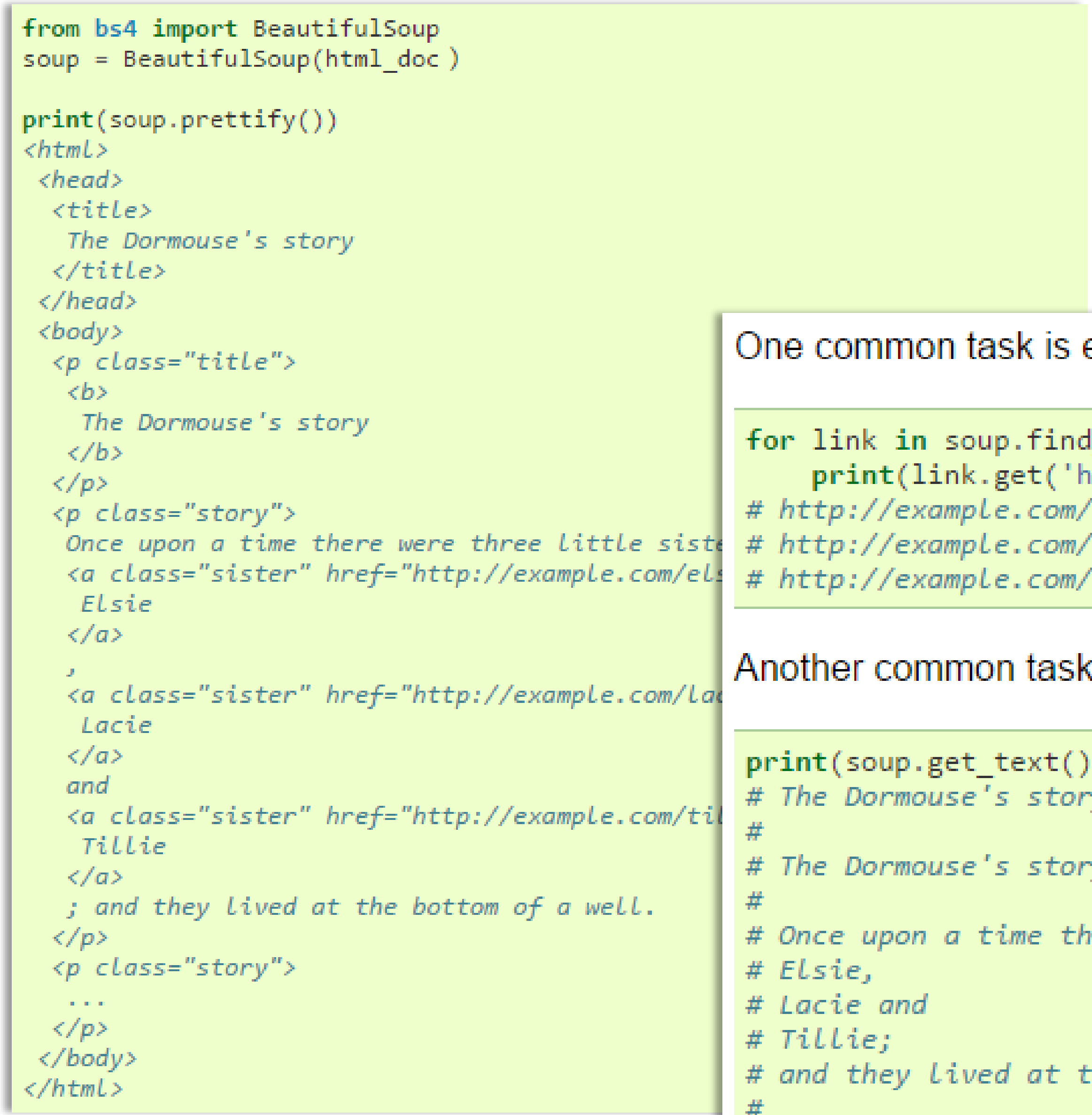
```
from bs4 import BeautifulSoup
soup = BeautifulSoup(html_doc )

print(soup.prettify())
<html>
 <head>
  <title>
   The Dormouse's story
  </title>
 </head>
 <body>
  <p class="title">
   <b>
    The Dormouse's story
   </b>
  </p>
  <p class="story">
   Once upon a time there were three lit
   <a class="sister" href="http://exampl
    Elsie
   </a>
   ,
   <a class="sister" href="http://exampl
    Lacie
   </a>
   and
   <a class="sister" href="http://exampl
    Tillie
   </a>
   ; and they lived at the bottom of a w
  </p>
  <p class="story">
   ...
  </p>
 </body>
</html>
```



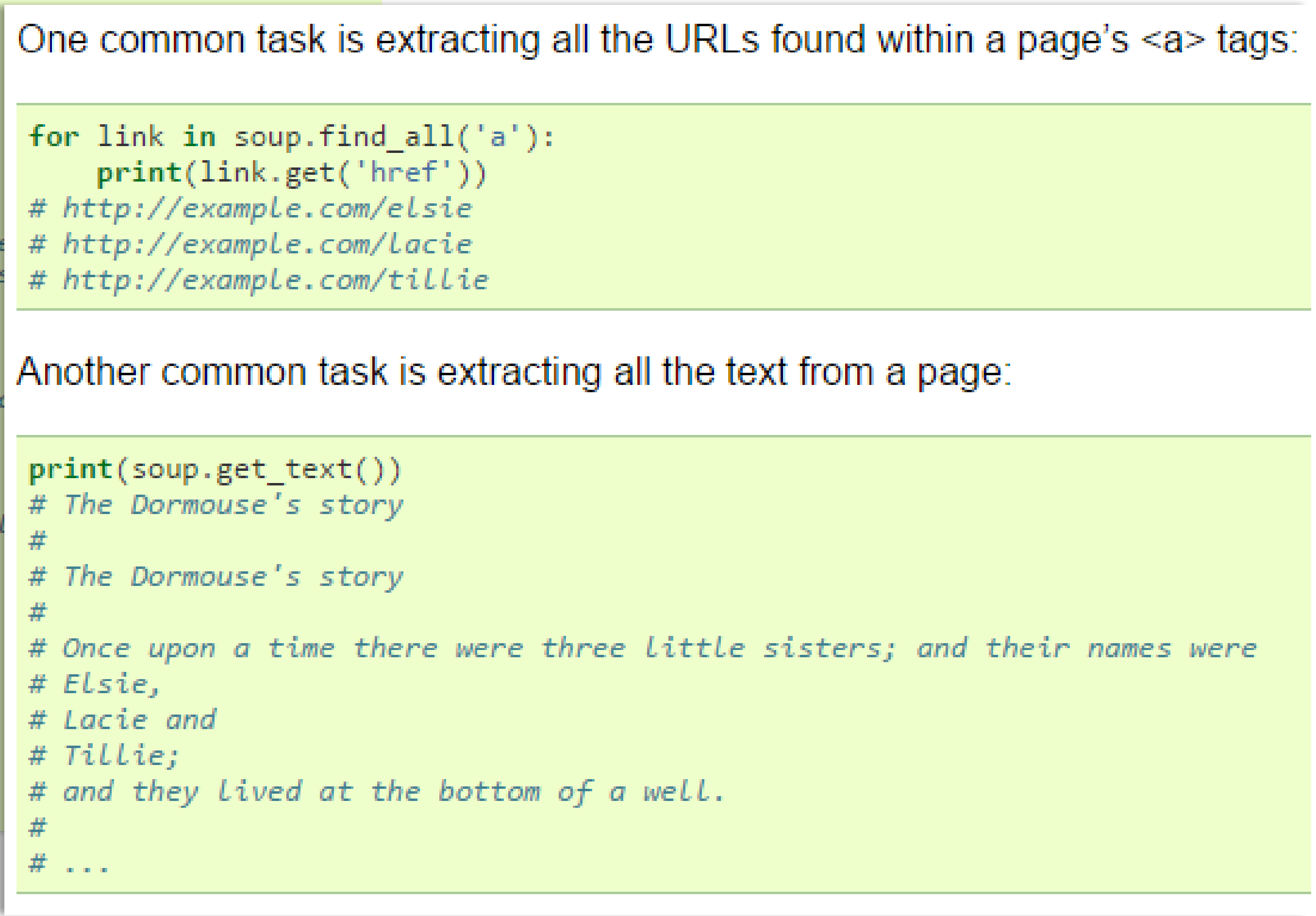
```
soup.title
# <title>The Dormouse's story</title>

soup.title.name
# u'title'

soup.title.string
# u'The Dormouse's story'

soup.title.parent.name
# u'head'

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/elsie" id="link1">Elsie</a>

soup.find_all('a')
# [<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,
#  <a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>,
#  <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>]

soup.find(id="link3")
# <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>
```

https://www.crummy.com/software/BeautifulSoup/bs4/doc/

Beautiful Soup



```
from bs4 import BeautifulSoup
soup = BeautifulSoup(html_doc )

print(soup.prettify())
<html>
 <head>
  <title>
   The Dormouse's story
  </title>
 </head>
 <body>
  <p class="title">
   <b>
    The Dormouse's story
   </b>
  </p>
  <p class="story">
   Once upon a time there were three little siste
   <a class="sister" href="http://example.com/els
    Elsie
   </a>
   ,
   <a class="sister" href="http://example.com/lac
    Lacie
   </a>
   and
   <a class="sister" href="http://example.com/til
    Tillie
   </a>
   ; and they lived at the bottom of a well.
  </p>
  <p class="story">
   ...
  </p>
 </body>
</html>
```



One common task is extracting all the URLs found within a page's <a> tags:

```
for link in soup.find_all('a'):
    print(link.get('href'))
# http://example.com/elsie
# http://example.com/lacie
# http://example.com/tillie
```

Another common task is extracting all the text from a page:

```
print(soup.get_text())
# The Dormouse's story
#
# The Dormouse's story
#
# Once upon a time there were three little sisters; and their names were
# Elsie,
# Lacie and
# Tillie;
# and they lived at the bottom of a well.
#
# ...
```

https://www.crummy.com/software/BeautifulSoup/bs4/doc/

Beautiful Soup

```
from bs4 import BeautifulSoup
soup = BeautifulSoup(html_doc )

print(soup.prettify())
<html>
 <head>
  <title>
   The Dormouse's story
  </title>
 </head>
 <body>
  <p class="title">
   <b>
    The Dormouse's story
   </b>
  </p>
  <p class="story">
   Once upon a time there were three little sisters; and their na
   <a class="sister" href="http://example.com/elsie" id="link1">
    Elsie
   </a>
   ,
   <a class="sister" href="http://example.com/lacie" id="link2">
    Lacie
   </a>
   and
   <a class="sister" href="http://example.com/tillie" id="link2">
    Tillie
   </a>
   ; and they lived at the bottom of a well.
  </p>
  <p class="story">
   ...
  </p>
 </body>
</html>
```

```
soup.find_all('b')
# [<b>The Dormouse's story</b>]
```

```
import re
for tag in soup.find_all(re.compile("^b")):
    print(tag.name)
# body
# b
```

```
for tag in soup.find_all(re.compile("t")):
    print(tag.name)
# html
# title
```

```
soup.find_all(["a", "b"])
# [<b>The Dormouse's story</b>,
#  <a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,
#  <a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>,
#  <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>]
```

```
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)
# [<p class="title"><b>The Dormouse's story</b></p>,
#  <p class="story">Once upon a time there were...</p>,
#  <p class="story">...</p>]
```

https://www.crummy.com/software/BeautifulSoup/bs4/doc/

Beautiful Soup

```
from bs4 import BeautifulSoup
soup = BeautifulSoup(html_doc )

print(soup.prettify())
<html>
 <head>
  <title>
   The Dormouse's story
  </title>
 </head>
 <body>
  <p class="title">
   <b>
    The Dormouse's story
   </b>
  </p>
  <p class="story">
   Once upon a time there were three little sisters; and their names were
   <a class="sister" href="http://example.com/elsie" id="link1">
    Elsie
   </a>
   ,
   <a class="sister" href="http://example.com/lacie" id="link2">
    Lacie
   </a>
   and
   <a class="sister" href="http://example.com/tillie" id="link2">
    Tillie
   </a>
   ; and they lived at the bottom of a well.
  </p>
  <p class="story">
   ...
  </p>
 </body>
</html>
```

```
soup.find_all("title")
# [<title>The Dormouse's story</title>]

soup.find_all("a")
# [<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,
#  <a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>,
#  <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>]

soup.find_all(id="link2")
# [<a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>]

import re
soup.find(string=re.compile("sisters"))
# u'Once upon a time there were three little sisters; and their names were\n'
```

```
soup.find_all(class_=re.compile("itl"))
# [<p class="title"><b>The Dormouse's story</b></p>]

def has_six_characters(css_class):
    return css_class is not None and len(css_class) == 6

soup.find_all(class_=has_six_characters)
# [<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,
#  <a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>,
#  <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>]
```

https://www.crummy.com/software/BeautifulSoup/bs4/doc/

Crawler Class

```
class crawler:
  # Initialize the crawler with the names of database tables
  def __init__(self, dbtables):
    pass

  # Starting with a list of pages, do a breadth-first search
  # to the given depth, indexing pages as we go
  def crawl(self, pages, depth=2):
    pass

  # Index an individual page
  def addtoindex(self, url, soup):
    pass

  # Extract the text from an HTML page (no tags)
  def gettextonly(self, soup):
    pass

  # Separate the words by any non-whitespace character
  def separatewords(self, text):
    pass
```

Crawling pages - crawl( )

```
def crawl(self, pages, depth=2):
    for i in range(depth):
      newpages = set()
      for page in pages:
        c = urllib2.urlopen(page)
        soup = BeautifulSoup(c.read())
        if not self.addtoindex(page, soup):
             continue

        links = soup.find_all('a')
        for link in links:
          if ('href' in dict(link.attrs)):
            url = urljoin(page, link['href'])

            if not self.isindexed(url):
              newpages.add(url)

            linkText = self.gettextonly(link)
            self.addlinkref(page, url, linkText)


      pages=newpages
```

Search Engine

1. Crawl to collect documents.
2. **Index to improve search.**
3. Query for a select set of documents.

Setting Up Database

Four dictionaries:

- **urllist** is the list of URLs that have been indexed.
  {url: outgoing_link_count}

- **wordlocation** is a list of the locations of words in the documents.
  {word: {url: [loc1, loc2, ..., locN]}}

- **link** stores two URL IDs, indicating a link from one page to another.
  {tourl: {fromUrl: None}}

- **linkwords** store words that are included in a link.
  {word: [(urlFrom1, urlTo1), ..., (urlFromN, urlToN)]}

Building the Database

- The database will be stored using `shelve` module
- Provides persistent object storage on disk
- Similar to anydbm, but more practical
- Use with `import shelve`

shelve – Persistent storage of arbitrary Python objects

- Key-value structure (like a dictionary)
- Persists data on disk (like anydbm)
- Keys may only be strings (like anydbm)
- Values may be any object (unlike anydbm, like a dictionary)
  - No need to pickle objects
- Handles updates automagically

shelve – open and insert data

```
import shelve


s = shelve.open('test_shelf.db')
s['key1'] = {'int': 10, 'float':9.5, 'string':'data'}

s.close()

# this will create test_shelf.db file on disk
```

shelve – read existing content

```
import shelve

s = shelve.open('test_shelf.db')
existing = s['key1']
print existing
s.close()
# prints: {'int': 10, 'float': 9.5, 'string': ‘data’}
```

shelve – auto update with writeback = True

```
import shelve

s = shelve.open('test_shelf.db')
print s['key1']
s['key1']['new_value'] = 'this was not here before'
s.close()

s = shelve.open('test_shelf.db')
print s['key1']
s.close()
# prints: {'int': 10, 'float': 9.5, 'string': 'data'}
          {'int': 10, 'float': 9.5, 'string': 'data'}
```

shelve – auto update with writeback = True

```
import shelve

s = shelve.open('test_shelf.db', writeback = True)
print s['key1']
s['key1']['new_value'] = 'this was not here before'
s.close()

# prints: {'int': 10, 'float': 9.5, 'string': 'data'}

s = shelve.open('test_shelf.db', writeback = True)
print s['key1']
s.close()

# prints: {'int': 10, 'new_value': 'this was not here
            before', 'float': 9.5, 'string': 'data'}
```

Setting Up the Database

```
import mysearchengine
pagelist = ['http://sehir.edu.tr']
dbtables = {'urllist': 'urllist.db',
            'wordlocation': 'wordlocation.db',
            'link': 'link.db',
            'linkwords': 'linkwords.db'}
crawler = mysearchengine.crawler(dbtables)
crawler.createindextables()
```

createindextables( )

```
# Create the database tables
def createindextables(self):
    # {url:outgoing_link_count}
    self.urllist = shelve.open(self.dbtables['urllist'], writeback=True, flag='c')

    #{word:{url:[loc1, loc2, ..., locN]}}
    self.wordlocation = shelve.open(self.dbtables['wordlocation'], writeback=True, flag='c')

    #{tourl:{fromUrl:None}}
    self.link = shelve.open(self.dbtables['link'], writeback=True, flag='c')

    #{word:[(urlFrom, urlTo), (urlFrom, urlTo), ..., (urlFrom, urlTo)]}
    self.linkwords = shelve.open(self.dbtables['linkwords'], writeback=True, flag='c')
```